EURASIP Book Series on Signal Processing and Communications

Advances in Signal Transforms

Theory and Applications

Edited by: Jaakko Astola and Leonid Yaroslavsky

Advances in Signal Transforms: Theory and Applications

EURASIP Book Series on Signal Processing and Communications, Volume 7

Advances in Signal Transforms: Theory and Applications

Edited by: Jaakko Astola and Leonid Yaroslavsky

Hindawi Publishing Corporation
http://www.hindawi.com

EURASIP Book Series on Signal Processing and Communications
Editor-in-Chief: Alex Gershman

Hindawi Publishing Corporation
410 Park Avenue, 15th Floor, #287 pmb, New York, NY 10022, USA
Nasr City Free Zone, Cairo 11816, Egypt
Fax: +1-866-HINDAWI (USA Toll-Free)

ISBN 977-5945-55-0

Contents

Preface

Why yet another book on transforms? The answer to this question is simple: because transforms are the fundamental issue in digital signal, image, and video processing. Whatever we do in digital signal processing, from the very beginning to the very end, we do it in a domain of a certain signal transform. Researchers working in the field need to be constantly updated to its state of the art and progress.

Integral transforms, specifically convolution and Fourier and Laplace integral transforms, have been used in what we call now electronic and communication engineering since its very beginning (in 1920–1940). It is, apparently, impossible to give credit to numerous researchers who contributed to this process, but at least the following three names should be mentioned: Oliver Heaviside, Harry Nyquist, and Norbert Wiener. In the theory of optical imaging, E. Abbe revolutionized the theory of optical imaging even earlier when he suggested, in 1880, to treat lenses as Fourier transformers.

In 1940–1950, signal processing emerged from demands of audio communication, radar, and television. Being purely analog at the time, it was based on the same natural transforms, convolution and Fourier ones, implemented through analog lowpass, highpass, and bandpass filters and spectrum analyzers. Initially, integral transforms served only as instruments of the signal theory. With the advent of computers, signal processing became digital, which opened a completely new option of making transforms powerful instruments of applied signal processing.

It is not an exaggeration to assert that digital signal and image processing came to being with the introduction, in 1965 by Cooley and Tukey, of the fast Fourier transform [1]. This publication immediately resulted in impetuous growth of all branches of digital signal and image processing and their applications including such completely new ones as digital holography [2].

The second boom in this growth process was associated with the introduction into communication theory and signal processing, in 1970, of the Walsh transform [3] and the development of a large family of fast transforms with FFT-type algorithms [4]. Some of these transforms, such as Walsh-Hadamard and Haar transforms, already existed in mathematics, others were being invented "from scratch" to achieve better "energy compaction" while preserving the principle of fast algorithmic implementation. This development was mainly driven by the needs of data compression, though the usefulness of transform domain processing for signal and image restoration, enhancement, and feature extraction was also very quickly recognized. This period ended up with the acceptance of the discrete cosine transform (DCT) as the best choice between other available-at-the-time transforms and resulted in JPEG and MPEG standards for image, audio, and video compression.

Nowadays, audio, image, and video compression has become multibillion industries.

The next milestone in transform signal processing was the introduction, in the 1980, of a large family of new transforms that are known, due to J. P. Morlet, as wavelets [5]. There was a very rapid burst of works that followed the first publications. This development continued the line of inventing new transforms better suited for the purposes of signal processing. Specifically, the main motivation was to achieve a better local representation of signals in contrast to the "global" representation that is characteristic to Fourier, DCT, and Walsh-Hadamard transforms.

A common method in designing new transform is generating basis functions of the transform from a primary, or "mother" function by means of its certain modifications. The simplest method of such a modification is the coordinate shift. It leads to the convolution type of transforms, such as the sampling one. Yet another possible simple method is coordinate scaling. The above-mentioned fast transforms, with one exception, implement exactly this coordinate scaling method. The exception is Haar transform. Haar transform is built upon combining these two methods, coordinate shifting and scaling ones. This combination is exactly the method that gave rise to wavelets and imparted them their most attractive feature, that of multiresolution.

Since their introduction, wavelets have gained a great popularity. In late 90's, there were even claims that wavelets have made the Fourier transform obsolete. Of course, this was an over exaggeration. Undoubtedly, however, nowadays wavelets constitute a well-established and very valuable part of signal processing transform tools that has found a wide range of applications in data compression, feature extraction, and signal denoising.

What are the main directions of growth in the field of transforms for signal and image processing? We believe they are the following.

(i) Further development of "manmade" transforms, in particular, wavelets, for more efficient data representation and compression.
(ii) Perfecting numerical representation of "natural" transforms for new applications, such as, for instance, tomography and digital holography.
(iii) Research aimed at enabling local adaptivity of transform domain signal and image processing.
(iv) Exploration of the use of digital transforms in new applications and development of new practical transform domain processing methods to meet growing demands.

This volume collects some most recent developments in all the above directions in the theory and practice of the design and usage of transforms in digital signal and image processing. Of course, it does not pretend to cover all progress in this field. No other book can. The volume emerged mainly from the series of reports published by Tampere International Center for Signal Processing, Tampere University of Technology, beginning in 1998, proceedings of five international workshops on transforms and filter banks organized by the center, papers presented in special sessions of SPIE Annual Symposia on Electronic Imaging. We also invited to contribute to this volume a group of researchers from Tel Aviv University

that have recently made major advances in the theory of wavelets and sampling of multidimensional integral transforms in polar coordinates. For the volume, all contributions are appropriately updated to represent the state of the art in the field and to cover the most recent developments in different aspects of the theory and applications of transforms.

The book consists of two parts. The first part contains four chapters devoted to topical issues in the theory of signal transforms. The first two chapters of this part consider recent advances in wavelet transforms for image compression and for switching and logic design.

Chapter 1 reviews state-of-the-art trends in wavelets, introduces new families of the modern lifting-based dyadic and triadic biorthogonal wavelets and wavelet-type frames originated from continuous and discrete splines, and demonstrates their high efficiency in image compression and robustness to data losses.

Chapter 2 addresses recent developments in application of Haar transform to representation and optimization of switching functions and related decision diagrams, which can be used in fast prototyping by LUT-FPGAs and in hardware-software codesign. In addition, definitions of Haar spectral transform diagrams are extended to multi valued functions which lead to a large class of multi valued Haar functions and related transforms.

Two other chapters treat problems of discrete representation of integral transforms for digital holography and for image reconstruction in tomography.

Chapter 3 provides a comprehensive review of discrete transforms and their fast algorithms for numerical reconstruction of optical holograms and derives point spread functions of different hologram reconstruction algorithms to show how the reconstruction results and their metrological properties depend on the holographic optical set-up physical parameters, and on the reconstruction algorithm.

Chapter 4 introduces 2D pseudopolar Fourier transform that is then used to construct a 2D discrete Radon transform and the corresponding fast algorithms and to derive a 3D discrete X-ray transform that operates on 3D discrete images. The 2D discrete Radon transform together with the 3D discrete Radon transform and the 3D discrete X-ray transform provide a straightforward and complete framework for defining multidimensional Radon and X-ray transforms for sampled objects.

In the second part, advanced practical transform-based signal and image processing algorithms are considered. The first two chapters of this part describe two interrelated families of signal and image adaptive denoising and restoration methods that optimize data recovery locally in sliding window.

Chapter 5 describes, in detail and with support of extensive experimental data, 1D, 2D, and 3D sliding window recursive spectral analysis-based image restoration and enhancement methods and corresponding efficient computational algorithms that implement the principles of scalar empirical Wiener filtering in transform domain. The emphasis is done on sliding window DCT domain processing. Superiority of the methods' noise suppressing capability compared to that of wavelet

denoising methods is demonstrated and a unified treatment of both families of transform domain processing methods is suggested.

In Chapter 6, an efficient statistical method for selecting adaptive window size and shape for image local polynomial approximation in sliding window is introduced and its applications to image denoising and anisotropic differentiation are illustrated.

The last two chapters treat yet another topical issue in modern signal, image, and video processing, that of signal and image resampling and geometrical transformations. Two complementing families of signal and image resampling algorithms are presented here.

Chapter 7 is devoted to spline interpolation algorithms built upon a class of piecewise basis functions of minimal support constructed of uniform B-splines of different degree that are very susceptible for optimization and are very efficient in computational implementation.

Chapter 8 offers a comprehensive review of fast Fourier and DCT transform-based discrete sinc-interpolation algorithms, which can be regarded as a gold standard for resampling of sampled data, and illustrates their superb resampling accuracy and their various applications. As a practical solution of image resampling in irregular sampling grids, sliding window DCT domain discrete sinc-interpolation methods are introduced that, in addition to image resampling, take advantage of image restoration and enhancement capability of the filters described in Chapter 5.

Researchers with very diverse background, from pure mathematics to pure engineering, are working in the field of digital signal processing, so is the potential readership of the volume. Contributions in this volume reflect this diversity. Each chapter is self-contained and can be read separately. At the same time, the chapters are interrelated and readers may find it very instructive to compare approaches in Chapters 1 and 7, 3 and 4, 5 and 6, 7 and 8.

We address this book to graduate students and researchers in all fields of electrical and communications engineering, especially to young researchers who are just entering the field, are curious and keen on modern information technology, and are planning their own advancement. We also believe that practicing engineers and researchers in signal and image processing will as well find in the book a quite few new ideas and algorithms useful in their applications.

Jaakko Astola
Leonid Yaroslavsky

Bibliography

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[2] T. Huang, "Digital holography," *Proceedings of IEEE*, vol. 59, no. 9, pp. 1335–1346, 1971.

[3] H. Harmuth, *Transmission of Information by Orthogonal Functions*, Springer, New York, NY, USA, 1971.

[4] N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*, Springer, Berlin, Germany, 1975.

[5] I. Daubechis, "Where do wavelets come from? A personal point of view," *Proceedings of IEEE*, vol. 84, no. 4, pp. 510–513, 1996.

1 Wavelet and frame transforms originated from continuous and discrete splines

Amir Z. Averbuch and Valery A. Zheludev

New classes of diadic and triadic biorthogonal wavelets and wavelet-type frames in signal space are presented. The construction employs interpolatory filters with rational transfer functions that originate from continuous and discrete splines. These filters have linear phase. They are amenable either to fast cascading or parallel recursive implementation. The wavelets are applied to compression of still images, where they demonstrate a superb efficiency. Robust error recovery algorithms presented utilize the redundancy inherent in frame expansions. Experimental results recover images when (as much as) 60% of the expansion coefficients are either lost or corrupted. The proposed approach inflates the size of the image through framelet expansion and multilevel decomposition, thus providing redundant representation of the image.

1.1. Introduction

We present in this paper a few dyadic and triadic wavelet transforms and a family of wavelet frame transforms, which are generated by critically sampled (wavelets) or oversampled (frames) perfect reconstruction filter banks. The design of these generating filter banks is based on the following simple insertion rule: we construct a spline that interpolates a signal on a sparse grid and predict (approximate) missing samples by the values of the spline at corresponding points.

The goal of this paper is to demonstrate that this obvious idea has a diversity of implications and produces a set of tools that perfectly matches the needs of signal processing.

Most (but not all) of the results in the paper were presented in more details in our recent publications [1–10]. The outline of the results on triadic wavelet transforms is presented here for the first time. A detailed presentation is on the way.

Currently, there is no need to describe the importance of the wavelet transforms for signal processing. A wide variety of orthogonal and biorthogonal wavelets was designed since the famous Daubechies construction was presented in [18]. However, only few of them possess a combination of properties that are valuable

for signal and image processing: symmetry, interpolation, fair time-domain localization and smoothness of the synthesis waveforms, flat spectra, and any number of vanishing moments. The biorthogonal wavelets that we describe in the paper meet these requirements. Dyadic wavelets based on the discrete splines are closely related to the popular Butterworth filters. Dyadic synthesis wavelets based on the continuous splines of even order are the splines themselves. However, synthesis wavelets based on the splines of odd order form a new class of functions that are smoother than the generating splines.

These biorthogonal wavelets were tested for compression of still images. After the multiscale transform, the coefficients are coded using the SPIHT algorithm by Said and Pearlman [41] followed by arithmetic coding. The results on benchmark images demonstrated that the designed wavelet transforms are competitive with the popular B9/7 biorthogonal transform [15, 31], which is the core of the JPEG 2000 image compression standard. Note that the compression results in the paper differ from the compression results reported in our previous paper [7], where arithmetic coding was not used.

The theory of wavelet frames or framelets is an extension of wavelet analysis. Currently, it is a subject of extensive investigation by researchers working in signal processing and applied mathematics. A wavelet frame is generated by several mother wavelets and provides a redundant expansion of a function or a signal. Due to this redundancy, there is more freedom in the design and implementation of the frame transforms. The frame expansions of signals demonstrate resilience to quantization noise and to coefficients losses [23, 24, 29]. Thus, frames may serve as a tool for error correction in signals transmitted through lossy channels. Actually, the frame transforms of multimedia signals can be interpreted as joint source-channel encoding for lossy channels, which are resilient to quantization noise and erasures. This approach was developed in [21, 25, 32, 33]. Due to additional adaptation capabilities, the overcomplete representation of signals has a potential to succeed in feature extraction and identification of signals. Promising results on image reconstruction are recently reported in [1, 12, 13].

A common approach to construction of a framelet system in the function space L^2 starts from the introduction of a pair of refinable functions (or one function), which generate(s) multiresolution analysis in L^2. Then, the wavelets are derived by one or another method as linear combinations of refinable functions. Many construction schemes are based on unitary extension principle [40] for tight frames and mixed extension principle [39] for biframes. These principles reduce the construction of a framelet system to the design of a perfect reconstruction filter bank. The masks of the given refinable functions serve as lowpass filters in the filter bank.

On the other hand, the oversampled perfect reconstruction filter banks by themselves generate wavelet-type frames in signal space [16, 26]. We use filter banks as an engine to construct a new family of frames in the signal space. Under some relaxed conditions, infinite iterations of the frame filter banks result in limit functions, the so-called framelets, which generate the wavelet frames in L^2. The framelets are symmetric, interpolatory, and have flat spectra combined with

fine time-domain localization and efficient implementation of the transforms. The framelets are smooth and may have any number of vanishing moments. The redundancy rate is two.

Recently, a new oblique extension principle (OEP) was proposed [20], which essentially extends the tools for design of wavelet frames in L^2. New wavelet frames with advanced properties were constructed using OEP. However, the OEP scheme operates with filter banks that lack perfect reconstruction property. Therefore, these filter banks do not generate frames in signal space that is our prime goal.

We propose to use the wavelet frames (framelets) presented in this paper as a tool for error correction for signals transmitted through lossy channels. These frames provide minimal redundancy. The simplicity and low complexity involved in the decomposition and reconstruction of the designed frame transforms give rise to efficient joint source-channel coding and decoding. Their properties promise good error recovery capabilities. Results of our experiments with erasure recovery for multimedia images confirm this claim. It is shown by means of simulations that these framelets can effectively recover from random losses that are close to the theoretical limit.

We present also a new family of the so-called triadic biorthogonal wavelets, which, unlike the commonly used diadic wavelets, have dilation factor of 3. They originate from the insertion rule, where two missed samples are predicted by interpolatory spline. Unlike dyadic wavelet transforms, one step of the triadic transform splits the frequency domain into three subbands. Three waveforms participate in the expansion of a signal. This promises better adaptivity of the expansion to the properties of the signal. A useful property of the transforms derived from the continuous splines is that the corresponding waveforms are splines.

A similar approach can be applied to the construction of multiwavelets and multiwavelet frames, where Hermite interpolatory splines are used as a source for the design of filters. The results are presented in [6, 8].

Unlike most schemes for the construction of wavelets and wavelet frames, we use infinite impulse response (IIR) filters with rational transfer functions. Consequently, the corresponding waveforms do not have compact support. But this fact should hardly be counted as a drawback because of the exponential decay of the waveforms as the argument grows. As for the implementation, it can be carried out in a fast recursive mode. On the other hand, usage of IIR filters enables to achieve a combination of properties, which are impossible to get with finite impulse response (FIR) filters. For example, currently, only (anti)symmetric framelets with 3 vanishing moments are designed [28, 42]. But, using IIR filters, we succeeded in design of symmetric framelets with any number of vanishing moments.

Note that wavelet constructions that are based on filter banks with rational transfer functions were originally introduced in [27]. In particular, nonsymmetric wavelets, which are based on causal Butterworth filters, were presented in [27]. Petukhov [36] designed a family of symmetric wavelets with rational symbols and applied them to video compression [17].

The paper is organized as follows. In Section 1.2, we describe some known facts about filter banks and splines. In Section 1.3, we present prediction filters

that originate from continuous and discrete splines. Section 1.4 describes the lifting scheme for the design and implementation of diadic wavelet transforms and presents the corresponding filter banks. In Section 1.5, we present results of the experiments in image compression using the designed diadic wavelets. In Section 1.6, we explain the construction of interpolatory wavelet frames using spline-based filter banks, whereas in Section 1.7 these frames are applied to the erasure correction in transmitted signals. Section 1.8 is devoted to the design of triadic biorthogonal wavelet transforms. In the appendix, we outline the recursive implementation of IIR filters.

1.2. Preliminaries

In this section we describe some known facts about filter banks and splines that are used in the sequel.

1.2.1. Filter banks

We call the sequences $\mathbf{x} \triangleq \{x(n)\}$, $n \in \mathbb{Z}$, which belong to the space l_1, (and, consequently, to l_2) discrete-time signals. The z-transform of a signal $\mathbf{x}$ is defined as $X(z) \triangleq \sum_{n\in\mathbb{Z}} z^{-n}x(n)$. Throughout the paper, we assume that $z = e^{j\omega}$.

The input $x(n)$ and the output $y(n)$ of a linear discrete time shift-invariant system are linked by the discrete convolution $y(n) = \sum_{l\in\mathbb{Z}} h(n-l)x(l)$. This processing of the signal $\mathbf{x}$ is called digital filtering and the sequence $\{h(n)\}$ is called the impulse response of the filter $\mathbf{h}$. Its z-transform $H(z) \triangleq \sum_{n\in\mathbb{Z}} z^{-n}h(n)$ is called the transfer function of the filter. Usually, a filter is designated by its transfer function $H(z)$. The function $\widehat{H}(\omega) = H(e^{j\omega n})$ is called the frequency response of the digital filter.

If filtering a signal is accompanied by downsampling of upsampling, then it is called multirate filtering. For example, application to the signal $\mathbf{x}$ of the time-reversed filter $\widetilde{\mathbf{h}}$ followed by downsampling of factor N is $\widetilde{y}(l) = \sum_{n\in\mathbb{Z}} \widetilde{h}(n - Nl)x(n)$. Application to the signal $\mathbf{y}$ that is upsampled by factor N of the filter $\mathbf{h}$ is

$$\widehat{x} = \sum_{n\in\mathbb{Z}} h^k(l - Nn)y(n) \Longleftrightarrow X(z) = H(z)Y\left(z^N\right). \tag{1.1}$$

The set of filters $\{\widetilde{H}^k(z) = \sum_{n\in\mathbb{Z}} z^{-n}\widetilde{h}^k(n)\}_{k=0}^{K-1}$, which, being time-reversed and applied to the input signal $\mathbf{x}$, produces the set of decimated output signals $\{\widetilde{\mathbf{y}}^k\}_{k=0}^{K-1}$,

$$\widetilde{y}^k(l) = \sum_{n\in\mathbb{Z}} \widetilde{h}^k(n - Nl)x(n), \quad k = 0,\ldots,K-1, \tag{1.2}$$

is called the K-channel analysis filter bank. Here $N \in \mathbb{N}$ is the downsampling factor. The set of filters $\{H^k(z) = \sum_{n\in\mathbb{Z}} z^{-n}h^k(n)\}_{k=0}^{K-1}$, which, being applied to the

set of input signals $\{\mathbf{y^k}\}_{\mathbf{k=0}}^{\mathbf{K-1}}$ that are upsampled by factor N, produces the output signal $\hat{\mathbf{x}}$,

$$\hat{x}(l) = \sum_{k=0}^{K-1} \sum_{n\in\mathbb{Z}} h^k(l - Nn) y^k(n), \quad l \in \mathbb{Z}, \tag{1.3}$$

is called the K-channel synthesis filter bank. If the upsampled signals $\tilde{\mathbf{y}}^{\mathbf{k}}$, $k = 0, \ldots, K-1$, are used as an input to the synthesis filter bank and the output signal $\hat{\mathbf{x}} = \mathbf{x}$, then the pair of analysis-synthesis filter banks forms a perfect reconstruction (PR) filter bank. If the number of channels K equals the downsampling factor N, then the filter bank is said to be critically sampled. If $K > N$, then the filter bank is oversampled. Note that critically sampled PR filter banks are used in wavelet analysis, while oversampled PR filter banks serve as a source for the design of frames.

Polyphase representation provides tools to handle multirate filtering. Let $\mathbf{x} = \{x(k)\}_{k\in\mathbb{Z}}$ be a signal. The sequences $\mathbf{x}(n)^N \triangleq \{x_{kN+n}\}_{k\in\mathbb{Z}}$, $n = 0, \ldots, N-1$, are called the polyphase components of the signal $\mathbf{x}$. Their z-transforms are denoted either by $X_{n,N}(z)$ or, when it does not lead to confusion, simply by $X(n)(z)$. Similarly, we denote the polyphase components of a filter $H(z)$ either by $H_{n,N}(z)$ or by $H(n)(z)$. The z-transforms are represented through the polyphase components:

$$X(z) = \sum_{n=0}^{N-1} z^{-n} X_{n,N}(z^N). \tag{1.4}$$

Then, the filtering can be expressed in polyphase form. We apply the time-reversed filter $\tilde{\mathbf{h}}$ to the signal $\mathbf{x}$. In the z-domain, we have

$$\begin{aligned} \tilde{Y}(z) &= \sum_{m,n=0}^{N-1} z^{m-n} H_{m,N}(z^{-N}) X_{n,N}(z^N) \\ &= \sum_{r=0}^{N-1} z^{-r} \sum_{m=0}^{N-1} H_{m,N}(z^{-N}) X_{m+r,N}(z^N) \\ &= \sum_{r=0}^{N-1} z^{-r} \tilde{Y}_{r,N}(z^N). \end{aligned} \tag{1.5}$$

Thus, the polyphase components of the output are

$$\tilde{Y}_{r,N}(z) = \sum_{m=0}^{N-1} H_m\left(\frac{1}{z}\right) X_{m+r,N}(z), \quad r = 0, \ldots, N-1. \tag{1.6}$$

If filtering is followed by downsampling, then we retain only zero-polyphase component:

$$\widetilde{y}(l) = \sum_{n\in\mathbb{Z}} \widetilde{h}(n-Nl)x(n) \Longleftrightarrow \widetilde{Y}(z) = \sum_{m=0}^{N-1} H_m\left(\frac{1}{z}\right)X_m(z). \tag{1.7}$$

If filtering is applied to the upsampled signal $\mathbf{y}$ as in (1.1), then the polyphase components of the output are

$$\widehat{X}_{n,N}(z) = H_{n,N}(z)Y(z^N). \tag{1.8}$$

Equations (1.2)–(1.8) imply the following representation for the application of the analysis and synthesis filter banks:

$$\begin{pmatrix} \widetilde{Y}^0(z) \\ \widetilde{Y}^1(z) \\ \vdots \\ \widetilde{Y}^{K-1}(z) \end{pmatrix} = \widetilde{\mathbf{P}}\left(\frac{1}{z}\right)\cdot \begin{pmatrix} X_0(z) \\ X_1(z) \\ \vdots \\ X_{N-1}(z) \end{pmatrix}, \qquad \begin{pmatrix} \widehat{X}_0(z) \\ \widehat{X}_1(z) \\ \vdots \\ \widehat{X}_{N-1}(z) \end{pmatrix} = \mathbf{P}(z)\cdot \begin{pmatrix} Y^0(z) \\ Y^1(z) \\ \vdots \\ Y^{K-1}(z) \end{pmatrix}, \tag{1.9}$$

where the $K \times N$ analysis polyphase matrix is

$$\widetilde{\mathbf{P}}(z) \triangleq \begin{pmatrix} \widetilde{H}_0^0(z) & \widetilde{H}_1^0(z) & \cdots & \widetilde{H}_{N-1}^0(z) \\ \widetilde{H}_0^1(z) & \widetilde{H}_1^1(z) & \cdots & \widetilde{H}_{N-1}^1(z) \\ \vdots & \vdots & \vdots & \vdots \\ \widetilde{H}_0^{K-1}(z) & \widetilde{H}_1^{K-1}(z) & \cdots & \widetilde{H}_{N-1}^{K-1}(z) \end{pmatrix} \tag{1.10}$$

and the $N \times K$ synthesis polyphase matrix is

$$\mathbf{P}(z) \triangleq \begin{pmatrix} H_0^0(z) & H_0^1(z) & \cdots & H_0^{K-1}(z) \\ H_1^0(z) & H_1^1(z) & \cdots & H_1^{K-1}(z) \\ \vdots & \vdots & \vdots & \vdots \\ H_{N-1}^0(z) & H_{N-1}^1(z) & \cdots & H_{N-1}^{K-1}(z) \end{pmatrix}. \tag{1.11}$$

The condition for the analysis-synthesis pair of filter banks to form a PR filter bank is

$$\mathbf{P}(z)\cdot\widetilde{\mathbf{P}}(z) = \mathbf{I}_N, \tag{1.12}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix.

1.2.2. Bases and frames generated by filter banks

Oversampled PR filter banks form frames in the signal space, whereas critically sampled PR filter banks form biorthogonal bases.

Definition 1.1. A system $\widetilde{\Phi} \triangleq \{\widetilde{\phi}_j\}_{j\in\mathbb{Z}}$ of signals forms a frame of the signal space if there exist positive constants A and B such that for any signal $\mathbf{x} = \{x(l)\}_{l\in\mathbb{Z}}$,

$$A\|\mathbf{x}\|^2 \le \sum_{j\in\mathbb{Z}} \left|\langle \mathbf{x}, \widetilde{\phi}_j\rangle\right|^2 \le B\|\mathbf{x}\|^2. \tag{1.13}$$

If the frame bounds A and B are equal to each other, then the frame is said to be tight.

If the system $\widetilde{\Phi}$ is a frame, then there exists another frame $\Phi \triangleq \{\phi_i\}_{i\in\mathbb{Z}}$ of the signals space such that any signal $\mathbf{x}$ can be expanded into the sum $\mathbf{x} = \sum_{i\in\mathbb{Z}}\langle \mathbf{x}, \widetilde{\phi}_i\rangle\phi_i$. The analysis $\widetilde{\Phi}$ and synthesis Φ frames can be interchanged. Together they form the so-called biframe. If the frame is tight, then Φ can be chosen as $\Phi = c\widetilde{\Phi}$.

If the elements $\{\widetilde{\phi}_j\}$ of the analysis frame $\widetilde{\Phi}$ are linearly independent, then the synthesis frame Φ is unique, its elements $\{\phi_j\}$ are linearly independent, and the frames $\widetilde{\Phi}$ and Φ form a biorthogonal basis of the signal space.

Let the analysis $\{\widetilde{H}^k(z)\}_{k=0}^{K-1}$ and the synthesis $\{H^k(z)\}_{k=0}^{K-1}$ filter banks form a PR filter bank. Then,

$$\begin{aligned} x(l) &= \sum_{k=0}^{K-1}\sum_{n\in\mathbb{Z}} h^k(l-Nn)\widetilde{y}^{k,1}(n), \quad l\in\mathbb{Z}, \\ \widetilde{y}^{k,1}(l) &= \sum_{n\in\mathbb{Z}} \widetilde{h}^k(n-Nl)x(n), \quad k = 0,\dots,K-1. \end{aligned} \tag{1.14}$$

We denote for $k = 0,\dots,K-1$ that $\widetilde{\varphi}^{k,1} \triangleq \{\widetilde{\varphi}^{k,1}(l) = \widetilde{h}^k(l)\}_{l\in\mathbb{Z}}$ and $\varphi^{k,1} \triangleq \{\varphi^{k,1}(l) = h^k(l)\}_{l\in\mathbb{Z}}$, where $\{\widetilde{h}^k(l)\}$ and $\{h^k(l)\}$ are the impulse responses of the filters $\widetilde{H}^k(z)$ and $H^k(z)$, respectively. Then, (1.14) can be rewritten as

$$\begin{aligned} \mathbf{x} &= \sum_{k=0}^{K-1} \mathbf{x}^{k,1}, \quad \mathbf{x}^{k,1} \triangleq \sum_{n\in\mathbb{Z}} \widetilde{y}^{k,1}(n)\varphi^{k,1}(\cdot - Nn), \\ \widetilde{y}^{k,1}(n) &= \langle \mathbf{x}, \widetilde{\varphi}^{k,1}(\cdot - Nn)\rangle, \quad k = 0,\dots,K-1. \end{aligned} \tag{1.15}$$

Thus, the system $\widetilde{\Phi} \triangleq \{\widetilde{\varphi}^{k,1}(\cdot - Nn)\}_{n\in\mathbb{Z}}, k = 0,\dots,K-1$, of N-sample translations of the signals $\widetilde{\varphi}^{k,1}$ forms an analysis frame of the signal space. The system $\Phi \triangleq \{\varphi^{k,1}(\cdot - Nn)\}_{n\in\mathbb{Z}}$, $k = 0,\dots,K-1$, of N-sample translations of the signals $\varphi^{k,1}$ forms a synthesis frame of the signal space. Together $\widetilde{\Phi}$ and Φ form a biframe. In a special case when the filter banks are critically sampled, that is $K = N$, the pair $\widetilde{\Phi}$ and Φ forms a biorthogonal basis.

1.2.3. Multiscale transforms and discrete-time wavelets

As it is common in wavelet and framelet transforms, one of the filters in each filter bank is lowpass. We denote these filters by $\widetilde{H}^0(z)$ and $H^0(z)$. To expand the transform to a coarse scale, the analysis filter bank is applied to the output $\widetilde{y}^{0,1}(l) = \sum_{n\in\mathbb{Z}} \widetilde{\varphi}^0(n - Nl)x(n)$ from the lowpass filter $\widetilde{H}^0(z)$. Then, we have

$$\begin{aligned}
\widetilde{y}^{k,2}(l) &= \sum_{n\in\mathbb{Z}} \widetilde{h}^k(n - Nl)\widetilde{y}^{0,1}(n) = \sum_{n\in\mathbb{Z}} \widetilde{h}^k(n - Nl) \sum_{m\in\mathbb{Z}} \widetilde{\varphi}^{0,1}(m - Nn)x(m) \\
&= \sum_{m\in\mathbb{Z}} x(m) \sum_{n\in\mathbb{Z}} \widetilde{h}^k(n - Nl)\widetilde{\varphi}^{0,1}(m - Nn) = \sum_{m\in\mathbb{Z}} x(m)\widetilde{\varphi}^{k,2}(m - N^2 l) \\
&= \langle \mathbf{x}, \widetilde{\varphi}^{k,2}(\,\cdot\, - N^2 l)\rangle, \quad k = 0,\dots,K-1, \\
\widetilde{\varphi}^{k,2}(m) &\triangleq \sum_{n\in\mathbb{Z}} \widetilde{h}^k(n)\widetilde{\varphi}^{0,1}(n - Nm).
\end{aligned} \tag{1.16}$$

On the other hand, the array $\{\widetilde{y}^{0,1}(l)\}$ is restored as follows:

$$\widetilde{y}^{0,1}(l) = \sum_{k=0}^{K-1} \sum_{n\in\mathbb{Z}} h^k(l - Nn)\widetilde{y}^{k,2}(n), \quad l \in \mathbb{Z}, \tag{1.17}$$

and subsequently, the low-frequency component of the signal $\mathbf{x}$ is

$$\begin{aligned}
x^{0,1}(m) &= \sum_{l\in\mathbb{Z}} \widetilde{y}^{0,1}(l)\varphi^{0,1}(m - Nl) = \sum_{l\in\mathbb{Z}} \sum_{k=0}^{K-1} \sum_{n\in\mathbb{Z}} h^k(l - Nn)\widetilde{y}^{k,2}(n)\varphi^{0,1}(m - Nl) \\
&= \sum_{k=0}^{K-1} \sum_{n\in\mathbb{Z}} \widetilde{y}^{k,2}(n)\varphi^{k,2}(m - N^2 n), \\
\varphi^{k,2}(l) &\triangleq \sum_{n\in\mathbb{Z}} h^k(n)\varphi^{0,1}(n - Nl).
\end{aligned} \tag{1.18}$$

As a result, we have the following expansion of the signal $\mathbf{x}$:

$$\mathbf{x} = \sum_{k=0}^{K-1} \sum_{n\in\mathbb{Z}} \widetilde{y}^{k,2}(n)\varphi^{k,2}(\,\cdot\, - N^2 n) + \sum_{k=1}^{K-1} \sum_{n\in\mathbb{Z}} \widetilde{y}^{k,1}(n)\varphi^{k,1}(\cdot - Nn). \tag{1.19}$$

Next step consists of the application of the filter bank to the array $\{\widetilde{y}^{0,2}(l)\}$ and so on. After J iterations, the signal $\mathbf{x}$ appears expanded as follows:

$$\begin{aligned}\mathbf{x} = &\sum_{n\in\mathbb{Z}} \langle \mathbf{x}, \widetilde{\varphi}^{0,J}(\,\cdot\, - N^J n)\rangle \varphi^{0,J}(\,\cdot\, - N^J n) \\ &+ \sum_{j=1}^{J}\sum_{k=1}^{K-1}\sum_{n\in\mathbb{Z}} \langle \mathbf{x}, \widetilde{\varphi}^{k,j}(\,\cdot\, - N^j n)\rangle \varphi^{k,j}(\,\cdot\, - N^j n),\end{aligned} \tag{1.20}$$

where

$$\widetilde{\varphi}^{k,j}(m) \triangleq \sum_{n\in\mathbb{Z}} \widetilde{h}^k(n)\widetilde{\varphi}^{0,j-1}(n - Nm), \qquad \varphi^{k,j}(m) \triangleq \sum_{n\in\mathbb{Z}} h^k(n)\varphi^{0,j-1}(n - Nm). \tag{1.21}$$

We call the signals $\widetilde{\varphi}^{k,j}$ and $\varphi^{k,j}$ the analysis and synthesis discrete-time wavelets of jth scale, respectively. Equation (1.20) implies that shifts of the discrete-time wavelets form a biframe (biorthogonal basis) of the space of signals.

Scaling functions and continuous wavelets. It is well known (see [19]) that under certain conditions, the lowpass filter $H(z)$, such that $H(1) = 1$, generates a continuous scaling function $\varphi(t)$. To be specific, if the infinite product $\lim_{S\to\infty}\prod_{\nu=1}^{S} H(e^{\omega 2^{-\nu}\omega})$ converges to a function $\Phi(\omega) \in L^2(\mathbb{R})$, then the inverse Fourier transform of $\Phi(\omega)$ is the scaling function $\varphi(t) \in L^2(\mathbb{R})$, which is the solution to the refinement equation $\varphi(t) = 2\sum_{n\in\mathbb{Z}} h(n)\varphi(2t - n)$. Similarly, if the infinite product $\lim_{S\to\infty}\prod_{\nu=1}^{N} H(e^{\omega N^{-\nu}\omega})$ converges to a function $\Phi(\omega) \in L^2(\mathbb{R})$, then the inverse Fourier transform of $\Phi(\omega)$ is the scaling function $\varphi(t) \in L^2(\mathbb{R})$, which is a solution to the refinement equation $\varphi(t) = 2\sum_{k\in\mathbb{Z}} h_k\varphi(Nt - k)$. Thus, the refinement equation results in dilation with factor N for the scaling function $\varphi(t)$.

Assume that the lowpass analysis filter $\widetilde{H}^0(z)$ generates the analysis scaling function $\widetilde{\varphi}(t)$. If the impulse responses of the filters $\widetilde{H}^k(z)$, $k = 1,\ldots,K-1$, are finite or decay exponentially, then the continuous functions $\widetilde{\psi}^k(t) \triangleq \sum_{n\in\mathbb{Z}} \widetilde{h}^k(n)\widetilde{\varphi}(Nt - n)$ are called the continuous analysis wavelets with dilation factor N.

A wavelet $\widetilde{\psi}^k(t)$ has p vanishing moments if $\int_{-\infty}^{\infty} t^s\widetilde{\psi}^k(t)dt = 0, s = 0,\ldots,p-1$. The number of vanishing moments of the wavelet $\widetilde{\psi}^k(t)$ is equal to the multiplicity of zero of the filter $\widetilde{H}^k(z)$ at $z = 1$ (see [43]). The same facts hold for the synthesis filter bank $H^k(z)$, $k = 0,\ldots,K-1$.

We consider in this paper the cases $N = 2$ (diadic wavelets) and $N = 3$ (triadic wavelets).

1.3. Prediction filters originated from splines

It was mentioned in the introduction (Section 1.1) that once we constructed a spline that interpolates a signal on a sparse grid, we have to predict missing samples of the signal by the values of the spline at the intermediate points. Calculation of

these values reduces to filtering the array of interpolated samples. We derive these filters for different types of splines.

1.3.1. Filters originated from continuous splines

In this section, we use continuous splines as a source for the filter design.

1.3.1.1. B-splines

The centered B-spline of first order is the characteristic function of the interval $[-1/2, 1/2]$. The centered B-spline of order p is the convolution $M^p(x) = M^{p-1}(x) * M^1(x)$, $p \geq 2$. Note that the B-spline of order p is supported on the interval $(-p/2, p/2)$. It is positive within its support and symmetric around zero. The B-spline M^p consists of pieces of polynomials of degree $p-1$ that are linked to each other at the nodes such that $M^p \in C^{p-2}$. Nodes of B-splines of even order are located at points $\{k\}$ and of odd order at points $\{k + 1/2\}$, $k \in \mathbb{Z}$.

The Fourier transform of the B-spline of order p is

$$\widehat{M^p}(\omega) \triangleq \int_{-\infty}^{\infty} e^{-i\omega x} M^p(x)dx = \left(\frac{\sin \omega/2}{\omega/2}\right)^p. \tag{1.22}$$

The time-domain representation of the B-spline is

$$M^p(t) = \frac{1}{(p-1)!} \sum_{k=0}^{p-1} (-1)^k \binom{p}{k} \left(t + \frac{p}{2} - k\right)_+^{p-1}, \tag{1.23}$$

where $t_+ \triangleq (t + |t|)/2$.

We introduce the following sequences:

$$\mathbf{v}_0^p \triangleq \{M^p(k)\}, \qquad \mathbf{v}_{r,N}^p \triangleq \left\{M^p\left(k + \frac{r}{N}\right)\right\}, \quad r \in \mathbb{Z}. \tag{1.24}$$

Their z-transforms are, respectively, $v_0^p(z)$ and $v_{r,N}^p(z)$.

Due to the compact support of B-splines, these sequences are finite and the z-transforms are the Laurent polynomials.

Example 1.2. The Laurent polynomials for splines of second to fourth degrees are as follows.

Linear spline, $p = 2$:

$$v_0^2(z) = 1, \qquad v_{1,2}^2(z) = \frac{z+1}{2}, \qquad v_{1,3}^2(z) = v_{-1,3}^2(z^{-1}) = \frac{z+2}{3}. \tag{1.25}$$

Quadratic spline, $p = 3$:

$$v_0^3(z) = \frac{z + 6 + z^{-1}}{8}, \qquad v_{1,2}^3(z) = \frac{z+1}{2}, \qquad v_{1,3}^3(z) = v_{-1,3}^3(z^{-1}) = \frac{25z + 46 + z^{-1}}{72}. \tag{1.26}$$

Cubic spline, $p = 4$:

$$v_0^4(z) = \frac{z + 4 + z^{-1}}{6}, \qquad v_{1,2}^4(z) = \frac{z^2 + 23z + 23 + z^{-1}}{48}, \qquad v_{1,3}^4(z) = v_{-1,3}^4(z^{-1}) = \frac{z^2 + 60z + 93 + 8z^{-1}}{162}. \tag{1.27}$$

Spline of fourth degree, $p = 5$:

$$v_0^5(z) = \frac{z^2 + 76z + 230 + 76z^{-1} + z^{-2}}{384}, \qquad v_{1,2}^5(z) = \frac{z^2 + 11z + 11 + z^{-1}}{24}, \qquad v_{1,3}^5(z) = v_{-1,3}^5(z^{-1}) = \frac{625z^2 + 11516z + 16566 + 2396z^{-1} + z^{-2}}{31104}. \tag{1.28}$$

Proposition 1.3. *The Laurent polynomials $v_0^p(z)$ and $z^{-r}v_{r,2r}^p(z^{2r})$ are symmetric. The roots of $v_0^p(z)$ are all simple and negative. In addition, for all z, $|z| = 1$, $v_0^p(z) > 0$. All the Laurent polynomials become 1 at $z = 1$.*

1.3.1.2. Interpolatory continuous splines

Shifts of B-splines form a basis in the space of splines of order p on the grid $\{Nl\}_{l\in\mathbb{Z}}$. Namely, any spline S^p has the following representation:

$$S^p(x) = \sum_l q_l M^p\left(\frac{x}{N} - l\right). \tag{1.29}$$

Denote $\mathbf{q} \triangleq \{q_l\}$, $\mathbf{s}_{0,N}^p = \{s_{0,N}^p(l) \triangleq S^p(Nl)\}$, $\mathbf{s}_{r,N}^p = \{s_{r,N}^p(l) \triangleq S^p(Nl + r)\}$, $r \in \mathbb{Z}$, and let $Q(z)$, $s_{0,N}^p(z)$ and $s_{r,N}^p(z)$ be the z-transforms of these the sequences, respectively. We have

$$s_{0,N}^p(l) = \sum_n q_l M^p(l - n) \Longleftrightarrow s_{0,N}^p(z) = Q(z)v_0^p(z). \tag{1.30}$$

Thus, $s_{0,N}^p(z) = Q(z)v_0^p(z)$, where $v_0^p(z)$ is the z-transform of the sequence $\mathbf{u}^p$ defined in (1.24). From these formulas, we can derive expressions for the coefficients of the spline S_i^p which interpolates the given sequence $\mathbf{x}_0 \triangleq \{x_0(l)\} \in l^1$ at grid

points $\{Nl\}_{l\in\mathbb{Z}}$:

$$\begin{aligned} S_i^p(Nl) = x_0(l), \quad & l \in \mathbb{Z}, \\ \Longleftrightarrow\ & Q(z)v_0^p(z) = X_0(z) \\ \Longleftrightarrow\ & Q(z) = \frac{X_0(z)}{v_0^p(z)} \\ \Longleftrightarrow\ & q_l = \sum_{n=-\infty}^{\infty} \lambda_{l-n}^p x_0(n), \end{aligned} \tag{1.31}$$

where $\boldsymbol{\lambda}^p \triangleq \{\lambda_k^p\}$ is the sequence which is defined via its z-transform:

$$\lambda^p(z) = \sum_{k=-\infty}^{\infty} z^{-k}\lambda_k^p = \frac{1}{v_0^p(z)}. \tag{1.32}$$

The coefficients $\{\lambda_k^p\}$ decay exponentially as $|k| \to \infty$. Substitution of (1.31) into (1.29) results in an alternative representation of the interpolatory spline:

$$S_i^p(x) = \sum_{l=-\infty}^{\infty} x_0(l)L^p\left(\frac{x}{N} - l\right), \qquad L^p(x) \triangleq \sum_l \lambda_l^p M^p(x-l). \tag{1.33}$$

The spline $L^p(x)$, defined in (1.33), is called the fundamental spline. It interpolates the Kronecker delta sequence δ_k, that is, it vanishes at all the integer points except $t = 0$, where $L^p(0) = 1$. Due to decay of the coefficients $\{\lambda_k^p\}$, the spline $L^p(t)$ decays exponentially as $|t| \to \infty$.

The values of the fundamental spline at the intermediate points are

$$L^p\left(k + \frac{r}{N}\right) = \sum_l \lambda_l^p M^p\left(k - l + \frac{r}{N}\right), \quad r \in \mathbb{Z}. \tag{1.34}$$

Denote by $V_{r,N}^p(z)$ the z-transform of the sequence $\{L^p(k + r/N)\}$, $k \in \mathbb{Z}$. Then, we obtain from (1.34) that

$$V_{r,N}^p(z) = \frac{v_{r,N}^p(z)}{v_0^p(z)}, \qquad v_{r,N}^p(z) \triangleq \sum_{n\in\mathbb{Z}} z^{-n}M^p\left(n + \frac{r}{N}\right). \tag{1.35}$$

Hence, the values of the interpolatory spline at the intermediate points are

$$s_{r,N}^p(l) = \sum_n L^p\left(l + \frac{r}{N} - n\right)x_0(n) \Longleftrightarrow s_{r,N}^p(z) = V_{r,N}^p(z)X_0(z). \tag{1.36}$$

Switching into the signal processing terminology, we say that in order to derive the values of the interpolatory spline at the points $\{Nl+r\}$ around the points $\{Nl\}$ of interpolation, we have to filter the data $\{x_0(l)\}$ by the filters $V_{r,N}^p$ whose impulse

response $\{L^p(l + r/N)\}$, $k \in \mathbb{Z}$, is finite or decays exponentially as $|l| \to \infty$ (IIR filter).

Let $\mathbf{x} = \{x(l)\}_{l\in\mathbb{Z}}$ and $x_0(l) \triangleq x(Nl)$. Denote $x_r(l) \triangleq x(Nl + r)$. As we mentioned above, if a spline interpolates samples of a signal at $\{Nl\}$, that is, $S_i^p(Nl) = x_0(l)$, then the values of the spline at the points $\{Nl + r\}$ are used as a prediction for the corresponding samples of the signal $\mathbf{x}$: $x_r(l) \approx S_i^p(Nl + r)$.

Definition 1.4. A lowpass filter $F_{r,N}(z)$, whose impulse response $\{f_{r,N}(l)\}_{l\in\mathbb{Z}}$ is finite or decays exponentially, restores polynomials of degree p on the grid $\{Nl + r\}$ if for any polynomial $P(t)$ of degree p, the following identity $\sum_{n\in\mathbb{Z}} f_{r,N}(l - n)P(Nn) = P(Nl + r)$ for all $l \in \mathbb{Z}$ is true.

Example 1.5. The prediction filters derived from the interpolatory splines of second to fourth degrees are as follows.

Linear spline, $p = 2$:

$$V_{1,2}^2(z) = \frac{z+1}{2}, \qquad V_{1,3}^2(z) = V_{-1,3}^2\left(z^{-1}\right) = \frac{z+2}{3}. \tag{1.37}$$

Quadratic spline, $p = 3$:

$$V_{1,2}^3(z) = 4\frac{z+1}{z+6+z^{-1}}, \qquad V_{1,3}^3(z) = V_{-1,3}^3\left(z^{-1}\right) = \frac{25z+46+z^{-1}}{9\left(z+6+z^{-1}\right)}. \tag{1.38}$$

Cubic spline, $p = 4$:

$$V_{1,2}^4(z) = \frac{z^2+23z+23+z^{-1}}{8\left(z+4+z^{-1}\right)}, \qquad V_{1,3}^4(z) = V_{-1,3}^4\left(z^{-1}\right) = \frac{z^2+60z+93+8z^{-1}}{27\left(z+4+z^{-1}\right)}. \tag{1.39}$$

Spline of fourth degree, $p = 5$:

$$\begin{aligned} V_{1,2}^5(z) &= 16\frac{z^2+11z+11+z^{-1}}{z^2+76z+230+76z^{-1}+z^{-2}}, \\ V_{1,3}^5(z) &= V_{-1,3}^5\left(z^{-1}\right) = \frac{625z^2+11516z+16566+2396z^{-1}+z^{-2}}{81\left(z^2+76z+230+76z^{-1}+z^{-2}\right)}. \end{aligned} \tag{1.40}$$

Proposition 1.6 (properties of designed prediction filters). (1) *All filters $V_{r,N}^p(z)$ are lowpass, $V_{r,N}^p(1) = 1$.*

(2) *The filters $V_{r,N}^p(z)$ restore polynomials of degree $p - 1$ on the grid $\{Nl + r\}$.*

(3) *The filters $V_{r,2r}^{2q+1}(z)$, which are derived from splines of odd order (even degree), restore polynomials of degree p on the grid $\{2rl + r\}$ (superconvergence property).*

1.3.2. Filters originated from discrete splines

In this section, we use a special type of the so-called discrete splines as a source for filter design. The discrete splines are defined on the grid $\{k\}$ and they are the counterparts of the continuous polynomial splines.

The signal

$$\mathbf{b}_N^1 = \{b_N^1(l)\} \triangleq \begin{cases} 1 & \text{as } l = 0,\dots,N-1 \\ 0 & \text{otherwise} \end{cases} \Longleftrightarrow B_N^1(z) = 1 + z^{-1} + \cdots + z^{-(N-1)} \tag{1.41}$$

is called the discrete B-spline of first order with length N.

We define by recurrence the higher-order B-splines via discrete convolutions:

$$\mathbf{b}_N^p = \mathbf{b}_N^1 * \mathbf{b}_N^{p-1} \Longleftrightarrow B_N^p(z) = \left(1 + z^{-1} + \cdots + z^{-(N-1)}\right)^p. \tag{1.42}$$

If $N = 2M + 1$ is odd, then we can introduce the so-called centered B-splines as

$$\mathbf{q}_N^p = \{q_N^p(l)\}, \qquad q_N^p(l) \triangleq b_N^p(l + M^p) \Longleftrightarrow Q_N^p(z) = \left(z^M + \cdots + z^{-M}\right)^p. \tag{1.43}$$

In the case when $N = 2M$ is even, centering is possible if the order $p = 2m$ is even. Then,

$$\begin{gathered} \mathbf{q}_N^{2m} = \{q_N^{2m}(l)\}, \\ q_N^{2m}(l) \triangleq b_N^p(l + (2M-1)^m) \Longleftrightarrow Q_N^p(z) = \left(z^{2M-1} + \cdots + z^{-(2M-1)}\right)^m. \end{gathered} \tag{1.44}$$

In both cases, the centered B-splines are symmetric about the point $l = 0$ where they attain their maximal value. We assume that either $N = 2M + 1$ or $N = 2M$ and $p = 2m$.

Similar to continuous splines, the discrete spline $\mathbf{d}_N^p = \{d_N^p(l)\}_{l\in\mathbb{Z}}$ of order p on the grid $\{Nn\}$ is defined as a linear combination with real-valued coefficients of shifts of the centered B-spline:

$$d_N^p(l) \triangleq \sum_{n=-\infty}^{\infty} c_l q_N^p(l - Nn) \Longleftrightarrow D_N^p(z) = C(z^N) Q_N^p(z). \tag{1.45}$$

Let

$$q_{r,N}^p(l) \triangleq q_N^p(r + Nl), \quad r = \begin{cases} -M,\dots,M & \text{if } N = 2M+1, \\ -M+1,\dots,M & \text{if } N = 2M,\ p = 2m, \end{cases} \tag{1.46}$$

be the polyphase components of the discrete B-spline. Then, the polyphase components of the discrete spline are

$$d^p_{r,N}(l) \triangleq \sum_{n=-\infty}^{\infty} c_l q^p_{r,N}(l-N) \Longleftrightarrow D^p_{r,N}(z) = C(z)Q^p_{r,N}(z). \tag{1.47}$$

Proposition 1.7. *The component $Q^p_{0,N}(z)$ is symmetric about inversion $z \to 1/z$ and positive on the unit circle $|z| = 1$. All the components $Q^p_{r,N}(z)$ have the same value at $z = 1$:*

$$Q^p_{r,N}(1) = Q^p_0(1) \quad \forall r. \tag{1.48}$$

Our scheme to design prediction filters, which uses discrete splines, is the same as the scheme that is based on continuous splines. We construct the discrete spline $\mathbf{d}^p_N$, which interpolates even samples of the signal $\mathbf{x}$ on the sparse grid $\{Nl\}$ and predict missing samples by the corresponding values, of the constructed spline. Using (1.47), we find the z-transform of the coefficients of interpolatory spline

$$\begin{aligned} d^p_N(lN) = x(lN) &\Longleftrightarrow D^p_{0,N}(z) \\ &= C(z)Q^p_0(z) = X_{0,N}(z) \Longrightarrow C(z) = \frac{X_{0,N}(z)}{Q^p_{0,N}(z)}. \end{aligned} \tag{1.49}$$

Hence, the polyphase components $\mathbf{d}^p_{r,N}$, which are used for the prediction defined via the z-transforms, are

$$D^p_{r,N}(z) = C(z)Q^p_{r,N}(z) = U^p_{r,N}(z)X_{0,N}(z), \quad U^p_{r,N}(z) \triangleq \frac{Q^p_{r,N}(z)}{Q^p_{0,N}(z)}. \tag{1.50}$$

Thus, in order to predict the missed samples of the signal $\mathbf{x}$, we filter its polyphase component $\mathbf{x}_{0,N}$ with the filters $U^p_{r,N}(z)$. Equation (1.48) implies that $U^p_{r,N}(1) = 1$. Therefore, these filters are lowpass. We focus on filters with the downsampling factors $N = 2$ and $N = 3$.

1.3.2.1. Filters with the downsampling factor $N = 2$: relation to Butterworth filters

In the case when the downsampling factor $N = 2$, the filters $U^{2m}_{1,2}(z)$ can be presented explicitly for any even order $2m$.

We get from (1.44) that

$$Q^{2m}_2(z) = \rho^m(z), \qquad \rho(z) \triangleq z + 2 + \frac{1}{z}. \tag{1.51}$$

Hence, the polyphase components of the B-spline and the prediction filters are

$$Q_{0,2}^{2m}(z^2) = \frac{1}{2}(\rho^m(z) + \rho^m(-z)), \qquad Q_{1,2}^{2m}(z_2) = \frac{z}{2}(\rho^m(z) - \rho^m(-z)),$$
$$U_{1,2}^{2m}(z^2) = z\frac{\rho^m(z) - \rho^m(-z)}{\rho^m(z) + \rho^m(-z)}. \tag{1.52}$$

The filters $U_{1,2}^{2m}(z^2)$ are closely related to Butterworth filters [34], which are widely used in signal processing. To be specific, the functions

$$\lambda^m(z) \triangleq 1 + z^{-1}U_{1,2}^{2m}(z^2) = \frac{2\rho^m(z)}{\rho^m(z) + \rho^m(-z)},$$
$$\chi^m(z) \triangleq 1 - z^{-1}U_{1,2}^{2m}(z^2) = \lambda^m(-z) \tag{1.53}$$

are squared magnitudes of the transfer functions of half-band lowpass and high-pass Butterworth filters of order m, respectively.

1.3.2.2. Filters with downsampling factor $N = 3$

Denote $\tau(z) \triangleq z + 1 + 1/z$, $\theta \triangleq e^{2\pi j/3}$. Then,

$$Q_{0,3}^{p}(z^3) = \frac{\tau^p(z) + \tau^p(z\theta) + \tau^p(z/\theta)}{3},$$
$$Q_{1,3}^{p}(z^3) = z\frac{\tau^p(z) + \tau^p(z\theta)\theta + \tau^p(z/\theta)/\theta}{3}$$
$$\Longrightarrow U_{1,3}^{p}(z^3) = z\frac{\tau^p(z) + \tau^p(z\theta)\theta + \tau^p(z/\theta)/\theta}{\tau^p(z) + \tau^p(z\theta) + \tau^p(z/\theta)}, \tag{1.54}$$
$$Q_{-1,3}^{p}(z^3) = z\frac{\tau^p(z) + \tau^p(z\theta)/\theta + \tau^p(z/\theta)\theta}{3}$$
$$\Longrightarrow U_{1,3}^{p}(z^3) = z\frac{\tau^p(z) + \tau^p(z\theta)/\theta + \tau^p(z/\theta)\theta}{\tau^p(z) + \tau^p(z\theta) + \tau^p(z/\theta)}.$$

1.3.2.3. Examples of prediction filters

Linear discrete splines, $p = 2$:

$$U_{1,2}^{2}(z) = \frac{1+z}{2}, \qquad U_{1,3}^{2}(z) = \frac{2+z}{3}, \qquad U_{-1,3}^{2}(z) = \frac{2+1/z}{3}. \tag{1.55}$$

Quadratic discrete splines, $p = 3$:

$$U_{1,3}^{3}(z) = \frac{3(2+z)}{z+7+1/z}, \qquad U_{-1,3}^{3}(z) = \frac{3(2+1/z)}{z+7+1/z}. \tag{1.56}$$

Cubic discrete splines, $p = 4$:

$$U^4_{1,2}(z) = \frac{4(1+z)}{z+6+1/z}, \qquad U^4_{1,3}(z) = \frac{1/z+16+10z}{4z+19+4/z} = U^4_{-1,3}\left(\frac{1}{z}\right). \tag{1.57}$$

Discrete spline of sixth order, $p = 6$:

$$U^6_{1,2}(z) = \frac{(z+14+z^{-1})(1+z)}{6z^{-1}+20+6z}. \tag{1.58}$$

Discrete spline of eighth order, $r = 4$:

$$U^8_{1,2}(z) = \frac{8(1+z)(z^{-1}+6+z)}{z^{-2}+28z^{-1}+70+28z+z^2}. \tag{1.59}$$

1.4. Biorthogonal wavelet transforms generated by filter banks with downsampling factor $N = 2$ (diadic transforms)

In this section, we describe how to generate families of biorthogonal wavelet transforms and wavelet frames using spline-based prediction filters with downsampling factor $N = 2$ designed in Section 1.3. A useful tool for the design and implementation of the biorthogonal wavelet transforms when downsampling factor $N = 2$ is provided by the so-called lifting scheme introduced by Sweldens [44].

1.4.1. Lifting scheme: decomposition

Generally, the lifting mode of the wavelet transform consists of four steps: (1) split, (2) predict, (3) update or lifting, and (4) normalization.

(1) *Split*. The signal $\mathbf{x} = \{x(l)\}_{l\in\mathbb{Z}}$ is split into its polyphase components: $\mathbf{x}_r = \{x_r(l) \triangleq x(2l+r)\}_{l\in\mathbb{Z}}$, $r = 0, 1$.

(2) *Predict*. The even polyphase component is filtered by some lowpass prediction filter $\widetilde{F}(1/z)$, in order for the filtered version of $\mathbf{x}_0$ to predict the odd component $\mathbf{x}_1$. Then, the existing array $\mathbf{x}_1$ is replaced by the array $\mathbf{a}^1$, which is the difference between $\mathbf{x}_1$ and the predicted array.

In the z-domain, the operations are described as $\widetilde{A}^1(z) = X_1(z) - \widetilde{F}(1/z)X_0(z)$.

(3) *Update (lifting)*. Generally, downsampling the original signal $\mathbf{x}$ into $\mathbf{x}_0$ depletes the smoothness of the signal. To obtain a sparse signal similar to the original $\mathbf{x}$, the new odd array is filtered by a lowpass *update* filter, which we prefer to denote as $F(z)/2$. The filtered array is used to increase the smoothness of the even array $\mathbf{x}_0$:

$$\widetilde{A}^0(z) = X_0(z) + \frac{1}{2}F(z)\widetilde{A}^1(z). \tag{1.60}$$

Provided that the filter F is properly chosen, the even array $\mathbf{x}_0$ is transformed into a smoothed and downsampled replica of $\mathbf{x}$.

(4) *Normalization*. Finally, the smoothed array $\widetilde{\mathbf{y}}^0$ and the array of details $\widetilde{\mathbf{y}}^1$ are obtained by the operations $\widetilde{\mathbf{y}}^0 = \sqrt{2}\widetilde{\mathbf{a}}^0$, $\widetilde{\mathbf{y}}^1 = \widetilde{\mathbf{a}}^1/\sqrt{2}$.

1.4.2. Lifting scheme: reconstruction

One of the most attractive features of lifting schemes is that the reconstruction of the signal $\mathbf{x}$ from the arrays $\widetilde{\mathbf{y}}^{\mathbf{0}}$ and $\widetilde{\mathbf{y}}^{\mathbf{1}}$ is implemented by reverse decomposition.

Undo normalization: $\widetilde{\mathbf{a}}^0 = \widetilde{\mathbf{y}}^0/\sqrt{2}\ \widetilde{\mathbf{a}}^1 = \sqrt{2}\widetilde{\mathbf{y}}^1$.

Undo lifting: the even polyphase component $X_0(z) = \widetilde{A}^0(z) - (1/2)F(z)\widetilde{A}^1(z)$ is restored.

Undo predict: the odd polyphase component

$$X_1(z) = \widetilde{A}^1(z) + \widetilde{F}\left(\frac{1}{z}\right)X_0(z) \tag{1.61}$$

is restored.

Undo split: the last step is the standard restoration of the signal from its even and odd components. In the z-domain, it appears as $X(z) = X_1(z^2) + z^{-1}X_0(z^2)$.

1.4.3. Filter banks

Rewriting the lifting steps in a matrix form, we obtain the polyphase matrices of the wavelet transforms to be

$$\begin{pmatrix}\widetilde{Y}^0(z)\\ \widetilde{Y}^1(z)\end{pmatrix} = \begin{pmatrix}\sqrt{2} & 0\\ 0 & \dfrac{1}{\sqrt{2}}\end{pmatrix}\cdot\begin{pmatrix}1 & \dfrac{F(z)}{2}\\ 0 & 1\end{pmatrix}\cdot\begin{pmatrix}1 & 0\\ -\widetilde{F}\left(\dfrac{1}{z}\right) & 1\end{pmatrix}\cdot\begin{pmatrix}X_0(z)\\ X_1(z)\end{pmatrix}. \tag{1.62}$$

Hence, the analysis polyphase matrix is

$$\widetilde{\mathbf{P}}\left(\frac{1}{z}\right) = \begin{pmatrix}\sqrt{2}\left(1 - \dfrac{\widetilde{F}(1/z)F(z)}{2}\right) & \dfrac{F(z)}{\sqrt{2}}\\ -\dfrac{\widetilde{F}(1/z)}{\sqrt{2}} & \dfrac{1}{\sqrt{2}}\end{pmatrix}. \tag{1.63}$$

The reconstruction operations are represented by

$$\begin{pmatrix}X_0(z)\\ X_1(z)\end{pmatrix} = \begin{pmatrix}1 & 0\\ \widetilde{F}\left(\dfrac{1}{z}\right) & 1\end{pmatrix}\cdot\begin{pmatrix}1 & \dfrac{-F(z)}{2}\\ 0 & 1\end{pmatrix}\cdot\begin{pmatrix}\dfrac{1}{\sqrt{2}} & 0\\ 0 & \sqrt{2}\end{pmatrix}\cdot\begin{pmatrix}\widetilde{Y}^0(z)\\ \widetilde{Y}^1(z)\end{pmatrix}. \tag{1.64}$$

Hence, the synthesis polyphase matrix is

$$\mathbf{P}(z) = \begin{pmatrix} \dfrac{1}{\sqrt{2}} & \dfrac{-F(z)}{\sqrt{2}} \\ \dfrac{\widetilde{F}(1/z)}{\sqrt{2}} & \sqrt{2}\left(1 - \dfrac{\widetilde{F}(1/z)F(z)}{2}\right) \end{pmatrix}. \tag{1.65}$$

Obviously, $\mathbf{P}(z) = \widetilde{\mathbf{P}}(1/z)^{-1}$. Therefore, the perfect reconstruction condition is satisfied. We recall that

$$\widetilde{\mathbf{P}}(z) \triangleq \begin{pmatrix} \widetilde{H}_0^0(z) & \widetilde{H}_1^0(z) \\ \widetilde{H}_0^1(z) & \widetilde{H}_1^1(z) \end{pmatrix}, \qquad \mathbf{P}(z) \triangleq \begin{pmatrix} H_0^0(z) & H_0^1(z) \\ H_1^0(z) & H_1^1(z) \end{pmatrix}. \tag{1.66}$$

Thus, the analysis filters are

$$\begin{aligned} \widetilde{H}^0\left(\frac{1}{z}\right) &= \sqrt{2}\left(1 - \frac{\widetilde{F}(z^{-2})F(z^2) - zF(z^2)}{2}\right) = \frac{\sqrt{2}}{2}(1 + zF(z^2) + W(z^2)), \\ \widetilde{H}^1\left(\frac{1}{z}\right) &= z\frac{\sqrt{2}}{2}(1 - z^{-1}\widetilde{F}(z^{-2})), \end{aligned} \tag{1.67}$$

where

$$W(z) \triangleq 1 - \widetilde{F}(z^{-1})F(z). \tag{1.68}$$

Thus, the synthesis filters are

$$\begin{aligned} H^0(z) &= \frac{\sqrt{2}}{2}(1 + z^{-1}\widetilde{F}(z^{-2})) = z^{-1}\widetilde{H}^1\left(\frac{-1}{z}\right), \\ H^1(z) &= z\frac{\sqrt{2}}{2}(1 - z^{-1}F(z^{-2}) + W(z^2)). \end{aligned} \tag{1.69}$$

If one polyphase component of the filter $H(z)$ is a constant number, then the filter is called interpolatory. We see from (1.69) that the lowpass synthesis filter $H^0(z)$ is interpolatory. Consequently, the synthesis scaling function $\varphi(t)$, which is generated by $H^0(z)$, interpolates the Kroneker delta (up to a constant c). It means that

$$\varphi(l) = \begin{cases} c & \text{if } l = 0, \\ 0 & \text{otherwise,} \end{cases} \qquad l \in \mathbb{Z}. \tag{1.70}$$

1.4.4. Wavelets derived from spline-based filter banks

The perfect reconstruction condition for the presented filter banks is satisfied with any choice of lowpass filters $\widetilde{F}(z)$ and $F(z)$. However, once the filters $V_{1,2}^p(z)$ and

$U_{1,2}^{2m}(z)$, derived from continuous and discrete splines, respectively, are utilized, a diverse family of biorthogonal wavelets appears. Properties of these wavelets such as symmetry, interpolation, smoothness, flat spectra, good time-domain localizations, and vanishing moments fit well signal processing needs. Implementation of these transforms is highly efficient.

Note that number of vanishing moments in the *analysis* wavelets is important for the applications of signal processing while typical requirements to the *synthesis* wavelets are smoothness and good time-domain localization.

Proposition 1.8. *If any of the filters $V_{1,2}^{p}(z)$ and $U_{1,2}^{2m}(z)$ derived from splines is used as the prediction filter $\widetilde{F}(1/z)$ and any of them is used as the update filter $F(z)$, then*

(1) *the filters $\widetilde{H}^0$ and H^0 are lowpass while the filters $\widetilde{H}^1$ and H^1 are highpass;*

(2) *the filters $\widetilde{H}^0$ and H^0 generate continuous scaling functions $\widetilde{\varphi}(t)$ and $\varphi(t)$, whereas the filters $\widetilde{H}^1$ and H^1 generate continuous wavelets $\widetilde{\psi}(t)$ and $\psi(t)$ with vanishing moments;*

(3) *the filters in the analysis and synthesis filter banks are symmetric (up to a shift) about inversion $z \to 1/z$ and corresponding waveforms are symmetric in time domain and decay exponentially as $t \to \infty$.*

Update filters: in principle, any lowpass filter can serve as the update filter $F(z)$. However, in what follows, we choose the update filter $F(z)$ to be equal to the predict filter $\widetilde{F}(z)$.

1.4.5. Continuous splines

Proposition 1.9. *If the prediction filter in the lifting scheme $F(1/z) = V_{1,2}^{2m}(z)$ is derived from the continuous spline of even order $2m$, then*

(1) *the transfer function of the analysis highpass filter $\widetilde{H}^1(z)$ has zero of multiplicity $2m$ at $z = 1$, consequently, the analysis wavelet $\widetilde{\psi}(t)$ has $2m$ vanishing moments;*

(2) *the synthesis scaling function $\varphi(t)$ is equal to the fundamental spline $L^{2m}(x)$ of order $2m$ defined in* (1.33)*, consequently, it is continuous together with its derivatives up to order $2m-2$ (belongs to C^{2m-2});*

(3) *the synthesis wavelet $\psi(t)$ is a spline of order $2m$, and therefore, belongs to C^{2m-2}.*

Proposition 1.10. *If the prediction filter in the lifting scheme $F(1/z) = V_{1,2}^{2m-1}(z)$ is derived from the continuous spline of odd order $2m-1$, then*

(1) *the transfer function of the analysis highpass filter $\widetilde{H}^1(z)$ has zero of multiplicity $2m$ at $z = 1$, consequently, the analysis wavelet $\widetilde{\psi}(t)$ has $2m$ vanishing moments;*

(2) *the synthesis scaling function $\varphi(t)$ and the wavelet $\psi(t)$ are smoother than the generating spline (which belongs to C^{2m-3}).*

Example 1.11

Linear spline, $p = 2$:

$$\begin{aligned}\widetilde{H}^1(z) &= \frac{z\left(1 - z^{-1}V_{1,2}^2(z^2)\right)}{\sqrt{2}} = -\frac{(1-z)^2}{2\sqrt{2}},\\ H^0(z) &= z^{-1}\widetilde{H}^1(-z) = \frac{z^{-1}+2+z}{2\sqrt{2}}.\end{aligned} \tag{1.71}$$

The corresponding analysis wavelet $\widetilde{\psi}(t)$ has two vanishing moments. The synthesis waveforms belong to C^0.

Quadratic spline, $p = 3$:

$$\widetilde{H}^1(z) = z^{-1}\frac{(1-z)^4}{\sqrt{2}(z^2+6+z^{-2})}, \qquad H^0(z) = \frac{\left(z^{-1}+2+z\right)^2}{\sqrt{2}(z^2+6+z^{-2})}. \tag{1.72}$$

The corresponding analysis wavelet $\widetilde{\psi}(t)$ has four vanishing moments. The synthesis waveforms belong to C^2.

Cubic spline, $p = 4$:

$$\widetilde{H}^1(z) = z^{-1}\frac{(1-z)^4\left(z+4+z^{-1}\right)}{8\sqrt{2}(z^{-2}+4+z^2)}, \qquad H^0(z) = \frac{\left(z^{-1}+2+z\right)^2\left(z+4+z^{-1}\right)}{8\sqrt{2}(z^{-2}+4+z^2)}. \tag{1.73}$$

The corresponding analysis wavelet $\widetilde{\psi}(t)$ has four vanishing moments. The synthesis waveforms belong to C^2.

Interpolatory spline of fourth degree, $p = 5$:

$$\begin{aligned}\widetilde{H}^1(z) &= -z^{-2}\frac{(1-z)^6\left(z+z^{-1}-10\right)}{\sqrt{2}\left(z^4+76z^2+230+76z^{-2}+z^{-4}\right)},\\ H^0(z) &= \frac{\left(z+2+z^{-1}\right)^3\left(z+z^{-1}+10\right)}{\sqrt{2}\left(z^4+76z^2+230+76z^{-2}+z^{-4}\right)}.\end{aligned} \tag{1.74}$$

The corresponding analysis wavelet $\widetilde{\psi}(t)$ has six vanishing moments. The synthesis waveforms belong to C^4.

We display in Figures 1.1–1.4 the frequency responses of the analysis and synthesis filter banks, which are derived from continuous splines, and the corresponding scaling functions and wavelets. All the figures are organized identically. Each figure consists of four columns. The first column from left displays the analysis scaling function $\widetilde{\varphi}(t)$ (bottom) and the analysis wavelet $\widetilde{\psi}(t)$ (top). The second column from left displays the frequency responses of the analysis lowpass filter $\widetilde{H}^0(z)$ (bottom) and the analysis highpass filter $\widetilde{H}^1(z)$ (top). Next column displays

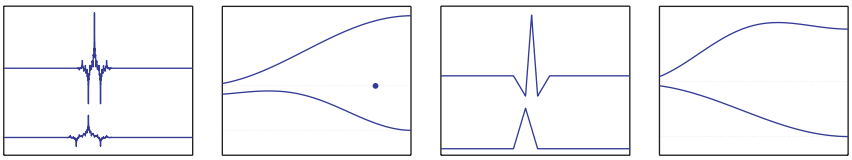


FIGURE 1.1. Filters and wavelets derived from the linear continuous spline.

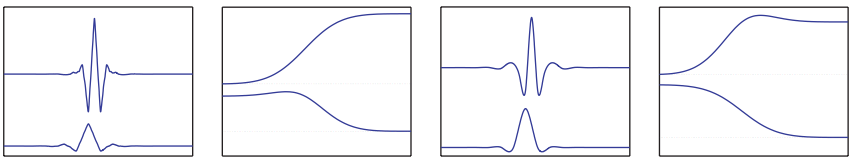


FIGURE 1.2. Filters and wavelets derived from the quadratic continuous spline.

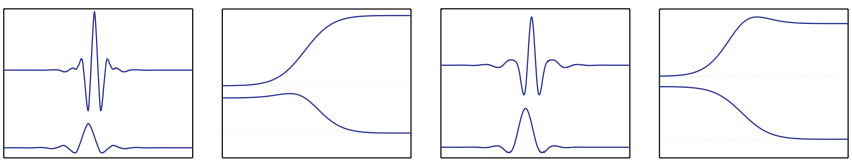


FIGURE 1.3. Filters and wavelets derived from the cubic continuous spline.

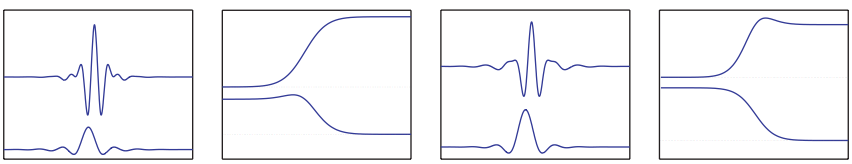


FIGURE 1.4. Filters and wavelets derived from the continuous spline of fourth degree.

the synthesis scaling function $\varphi(t)$ (bottom) and the synthesis wavelet $\psi(t)$ (top). Last column displays the frequency responses of the synthesis lowpass filter $H^0(z)$ (bottom) and the synthesis highpass filter $H^1(z)$ (top).

1.4.6. Discrete splines

Proposition 1.12. *If the prediction filter in the lifting scheme* $F(1/z) = U_{1,2}^{2m}(z)$ *is derived from the discrete spline of even order* $2m$*, then*

(1) *the transfer function of the analysis highpass filter* $\widetilde{H}^1(z)$ *has zero of multiplicity* $2m$ *at* $z = 1$*, consequently, the analysis wavelet* $\widetilde{\psi}(t)$ *has* $2m$ *vanishing moments;*

(2) *the transfer function of the synthesis lowpass filter* $H^0(z)$ *and the analysis highpass filter* $z^{-1}\widetilde{H}^1(z)$ *are the squared magnitudes of the transfer functions of half-band lowpass and highpass Butterworth filters of order* m*, respectively.*

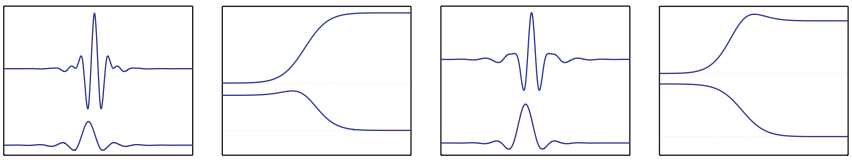


FIGURE 1.5. Filters and wavelets derived from the discrete spline of fifth degree.

Example 1.13

Linear spline, $m = 1$: the filter bank coincides with the filter bank derived from the continuous linear spline.

Cubic spline, $m = 2$: the filter bank coincides with the filter bank derived from the continuous quadratic spline.

Discrete spline of fifth degree, $m = 3$:

$$\widetilde{H}^1(z) = z^{-2}\frac{(1-z)^6}{\sqrt{2}\left(6z^2+20+6z^{-2}\right)}, \qquad H^0(z) = -\frac{\left(z^{-1}-2+z\right)^3}{\sqrt{2}\left(6z^2+20+6z^{-2}\right)}. \tag{1.75}$$

The corresponding analysis wavelet $\widetilde{\psi}(t)$ has six vanishing moments. The synthesis waveforms belong to C^4.

Discrete spline of seventh degree, $m = 4$:

$$\widetilde{H}^1(z) = z^{-3}\frac{(1-z)^8}{\sqrt{2}\left(z^{-4}+28z^{-2}+70+28z^2+z^4\right)}, \qquad H^0(z) = \frac{\left(z^{-1}-2+z\right)^4}{\sqrt{2}\left(z^{-4}+28z^{-2}+70+28z^2+z^4\right)}. \tag{1.76}$$

The corresponding analysis wavelet $\widetilde{\psi}(t)$ has eight vanishing moments. The synthesis waveforms belong to C^5.

We display in Figures 1.5 and 1.6 the frequency responses of the analysis and synthesis filter banks, which are derived from discrete splines, and the corresponding scaling functions and wavelets. The figures are organized similarly to Figures 1.1–1.4.

Remarks 1.14. (1) From the above figures, we see that the designed waveforms are smooth (except for the analysis waveforms derived from the linear spline), well localized in time domain, and symmetric.

(2) The frequency responses of filters are maximally flat (has no ripples) in the passband, and rolls towards zero in the stopband. As the order of the generating spline is high, the decline is steeper in the stopband.

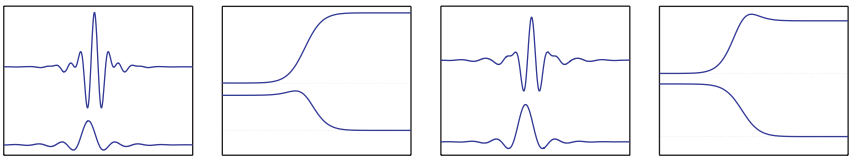


FIGURE 1.6. Filters and wavelets derived from the discrete spline of seventh degree.

(3) The frequency responses of the filters $H^0(z)$ and $\widetilde{H}^1(z)$ are monotonous, whereas the responses of $\widetilde{H}^0(z)$ and $H^1(z)$ have a bump near the cutoff. This bump stems from the presence of the term $W(z)$ in the transfer functions of these filters (see (1.67)–(1.69)). Also, from this reason, the analysis waveforms are less regular than the synthesis counterparts.

(4) The transfer functions of all the designed filters are rational functions. Therefore, the filters (except the filters derived from the linear spline) have infinite impulse response (IIR). But they can be efficiently implemented via causal-anticausal recursive filtering (see the appendix). The computational cost is even lower than the cost of implementing finite impulse response (FIR) filters with similar properties.

Properties of the designed wavelets and filters make them useful for signal processing applications. We describe one application in Section 1.5.

1.5. Application of spline-based wavelet transforms to image compression

The above transforms were applied to multimedia still images to achieve high-quality compression. In most experiments, they outperform the popular B9/7 biorthogonal transform [15, 31], which is the core of JPEG 2000 still image compression standard. In this section, we present the results after compression and decompression of four images in Figures 1.7 and 1.8. These are 512×512 8-bit per pixel (8 bpp) images.

The following experiments were done.

(1) The image was decomposed up to 6 scales by the wavelet transform using the B9/7 transforms and the transforms designed in Section 1.4.

(2) The transform coefficients were coded using the SPIHT algorithm (see [41]) followed by arithmetic coding. This algorithm enables to achieve exact predetermined compression rate. We coded the coefficients with different compression ratios (CR) 1 : 10 (0.8 bpp), 1 : 20 (0.4 bpp), 1 : 30 (4/15 bpp), 1 : 40 (0.2 bpp), and 1 : 50 (4/25 bpp).

(3) The reconstructed image was compared with the original image and the achieved peak signal-to-noise ratio (PSNR) in decibels was computed:

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{N 255^2}{\sum_{k=1}^{N} \left(x(k) - \widetilde{x}(k) \right)^2} \right) \mathrm{dB}. \tag{1.77}$$

(a) (b)

FIGURE 1.7. (a) Lena, (b) Barbara.

(a) (b)

FIGURE 1.8. (a) Canaletto, (b) fabrics.

We denote the transforms as follows.

(i) B9/7 denotes the B9/7 biorthogonal transform.
(ii) CS3 denotes the transform generated by quadratic continuous spline (1.72).
(iii) CS4 denotes the transform generated by cubic continuous spline (1.73).
(iv) CS5 denotes the transform generated by continuous spline of fourth degree (1.74).
(v) DS3 denotes the transform generated by discrete spline of sixth degree (1.75).
(vi) DS4 denotes the transform generated by discrete spline of eighth degree (1.76).

The results are summarized in Tables 1.1–1.4.

Lena: the PSNR values of "Lena" are presented in Table 1.1.

All the spline wavelet transforms outperform the B9/7 filter in any compression rate (with one exception for CS3 at CR = 10). In Figure 1.9, we display the

TABLE 1.1. PSNR of "Lena" after the application of SPIHT where the decomposition of the wavelet transform was 6 scales.

CR	B9/7	CS3	CS4	CS5	DS3	DS4
10	39.12	39.09	39.14	**39.15**	**39.15**	39.14
20	36.08	36.14	36.17	36.17	**36.18**	36.12
30	34.12	34.23	34.27	34.29	**34.30**	34.23
40	32.99	33.05	33.08	33.07	**33.09**	33.08
50	31.90	31.91	**31.98**	31.96	**31.98**	31.92

TABLE 1.2. PSNR of "Barbara" after the application of SPIHT where the decomposition of the wavelet transform was 6 scales.

CR	B9/7	CS3	CS4	CS5	DS3	DS4
10	35.36	35.64	35.99	36.27	36.21	**36.39**
20	30.63	30.87	31.05	31.19	31.17	**31.27**
30	28.27	28.09	28.27	28.46	28.42	**28.59**
40	26.91	26.78	26.93	27.13	27.08	**27.22**
50	26.17	25.72	25.89	26.04	26.02	**26.18**

TABLE 1.3. PSNR of "Canaletto" after the application of SPIHT where the decomposition of the wavelet transform was 6 scales.

CR	B9/7	CS3	CS4	CS5	DS3	DS4
10	35.10	35.16	35.20	**35.21**	**35.21**	**35.21**
20	31.48	31.59	31.66	**31.68**	**31.68**	31.67
30	29.54	29.66	29.71	**29.74**	**29.74**	29.72
40	28.38	28.50	28.54	28.54	**28.55**	28.52
50	27.55	27.63	**27.63**	**27.63**	**27.63**	27.62

TABLE 1.4. PSNR of "fabrics" after the application of SPIHT where the decomposition of the wavelet transform was 6 scales.

CR	B9/7	CS3	CS4	CS5	DS3	DS4
10	**36.30**	36.21	36.24	36.24	36.24	36.20
20	**32.58**	32.42	32.45	32.45	32.45	32.43
30	**31.14**	31.02	31.07	31.04	31.06	30.99
40	**30.08**	30.04	30.03	30.03	30.03	29.97
50	**29.38**	29.36	29.37	29.36	29.37	29.31

closed-up fragments of "Lena" reconstructed from 1 : 50 compression files of wavelet coefficients of the B9/7 and DS3 transforms.

Barbara: the PSNR values of "Barbara" are presented in Table 1.2.

All the spline-wavelet transforms outperform the B9/7 filter in any compression rate, especially at high bit rate. Most efficient is the DS4, where the wavelets have eight vanishing moments and the synthesis waveforms belong to C^4. In Figure 1.10, we display closed-up fragments of "Barbara" reconstructed from 1 : 40 compression files of wavelet coefficients of B9/7 and DS4 transforms.

Canaletto: the PSNR values of "Canaletto" are presented in Table 1.3.

(a) (b)

FIGURE 1.9. Reconstruction of "Lena" from 1 : 50 compression files of wavelet coefficients of B9/7 (a) and DS3 (b) transforms.

(a) (b)

FIGURE 1.10. Reconstruction of "Barbara" from 1 : 50 compression files of wavelet coefficients of B9/7 (a) and DS4 (b) transforms.

All the splinetransforms slightly outperform the B9/7 filter in any compression rate. Most efficient are the DS3 and DS4, where the wavelets have six and eight vanishing moments, respectively. In Figure 1.11, we display closed-up fragments of "Canaletto" reconstructed from 1 : 40 compression files of wavelet coefficients of B9/7 and DS3 transforms.

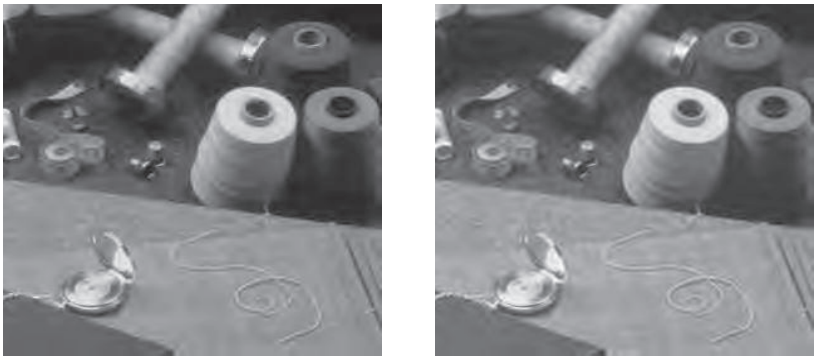
Fabrics: the PSNR values of "fabrics" are presented in Table 1.4.

The B9/7 filter demonstrates a small advantage of the PSNR over all splinetransforms on this image. In Figure 1.12, we display closed-up fragments of "fabrics" reconstructed from 1 : 50 compression files of wavelet coefficients of B9/7 and DS3 transforms.

The examples presented above demonstrate that the performance of the designed spline-based wavelet transforms for image compression is competitive to the performance of the popular B9/7 transform, which is used in JPEG 2000. Once

(a) (b)

FIGURE 1.11. Reconstruction of "Canaletto" from 1 : 50 compression files of wavelet coefficients of B9/7 (a) and DS3 (b) transforms.



(a) (b)

FIGURE 1.12. Reconstruction of "fabrics" from 1 : 50 compression files of wavelet coefficients of B9/7 (a) and DS3 (b) transforms.

the recursive implementation of the spline filters is conducted, its computational cost is comparable with the cost of the implementation of B9/7 transform. For example, the number of additions (A) and multiplications (M) per pixel for the B9/7 transform is 8A + 4M, whereas for the transform CS3, it is 8A + 6M operations. The efficient DS3 transform requires 12A + 8M operations.

1.6. Wavelet frames (framelets) generated by 3-channel filter banks with downsampling factor $N = 2$

As we mentioned in the end of Section 1.4, the analysis wavelets are less regular than the synthesis ones. The reason is that the structure of the lowpass analysis filters is more complicated than the structure of their synthesis counterparts. The latter is very simple and has the interpolation property. If the prediction filter originates from the discrete spline, then the lowpass synthesis filter coincides with the

squared magnitude of the transfer function of half-band lowpass Butterworth filter. Also we mentioned the bumps that disturbed the flatness of the frequency responses of the lowpass analysis filters and the highpass synthesis ones. We can essentially improve the properties of the filters and waveforms by introducing additional channel into the filter bank retaining the perfect reconstruction property. Thus, the representation of a signal becomes redundant and produces a frame expansions of signals. The redundancy should hardly be regarded as a drawback. The redundant representations have more adaptation abilities compared to basis expansions of signals. Therefore, they are feasible for signal denoising and features extraction. This redundancy enables to exploit frame expansion as a tool for recovery of erasures, which may occur while a multimedia signal is transmitted through a lossy channel.

1.6.1. Interpolatory frames

In this section, we describe how to construct frames in signal space starting from either any pair of lowpass interpolatory filters or from a single filter. The problem reduces to the design of a perfect reconstruction filter bank with desired properties. The key point in this design is the factorization scheme of a polyphase matrix.

We introduce an analysis-synthesis pair of lowpass interpolatory filters

$$\widetilde{G}^0(z) \triangleq \frac{\sqrt{2}}{2}\left(1 + z^{-1}\widetilde{F}(z^2)\right), \qquad G^0(z) \triangleq \frac{\sqrt{2}}{2}\left(1 + z^{-1}F(z^2)\right), \tag{1.78}$$

that are similar to the lowpass interpolatory synthesis filter $H^0(z)$ derived from the lifting scheme (see (1.69)). As before, denote $W(z) \triangleq 1 - F(z)\widetilde{F}(z^{-1})$.

The polyphase matrices for the filter banks $\widetilde{G}^0$, $\widetilde{G}^1$, $\widetilde{G}^2$ and G^0, G^1, G^2 comprising the interpolatory lowpass filters $G^0(z)$ and $\widetilde{G}^0(z)$ are

$$\widetilde{\mathbf{P}}(z) \triangleq \begin{pmatrix} \dfrac{1}{\sqrt{2}} & \dfrac{\widetilde{F}(z)}{\sqrt{2}} \\ \widetilde{G}_0^1(z) & \widetilde{G}_1^1(z) \\ \widetilde{G}_0^2(z) & \widetilde{G}_1^2(z) \end{pmatrix}, \qquad \mathbf{P}(z) \triangleq \begin{pmatrix} \dfrac{1}{\sqrt{2}} & G_0^1(z) & G_0^2(z) \\ \dfrac{F(z)}{\sqrt{2}} & G_1^1(z) & G_1^2(z) \end{pmatrix}. \tag{1.79}$$

Then, the perfect reconstruction condition (1.12) leads to

$$\mathbf{P}_g(z) \cdot \widetilde{\mathbf{P}}_g\left(\frac{1}{z}\right) = \frac{1}{2}\mathbf{Q}(z), \tag{1.80}$$

where

$$\widetilde{\mathbf{P}}_g(z) \triangleq \begin{pmatrix} \widetilde{G}_0^1(z) & \widetilde{G}_1^1(z) \\ \widetilde{G}_0^2(z) & \widetilde{G}_1^2(z) \end{pmatrix}, \qquad \mathbf{P}_g(z) \triangleq \begin{pmatrix} G_0^1(z) & G_0^2(z) \\ G_1^1(z) & G_1^2(z) \end{pmatrix},$$
$$\mathbf{Q}(z) \triangleq \begin{pmatrix} 1 & -\widetilde{F}(z^{-1}) \\ -F(z) & 2 - F(z)\widetilde{F}(z^{-1}) \end{pmatrix}. \tag{1.81}$$

Therefore, the construction of a frame with the interpolatory lowpass filters $\widetilde{G}^0(z)$ and $G^0(z)$ reduces to factorization of the matrix $\mathbf{Q}(z)$ as in (1.80). There are many options for this factorization of $\mathbf{Q}(z)$. We describe implications of the simplest triangular factorization:

$$\widetilde{\mathbf{P}}_g(z) = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & \widetilde{w}(z) \\ 1 & -\widetilde{F}(z) \end{pmatrix}, \quad \mathbf{P}_g(z) = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 \\ w(z) & -F(z) \end{pmatrix}, \quad w(z)\widetilde{w}(z^{-1}) = W(z). \tag{1.82}$$

Thus, to complete the construction, we have to factorize the function $W(z)$. As soon as it is done, we obtain the perfect reconstruction filter bank,

$$\begin{aligned} \widetilde{G}^0(z) &\triangleq \frac{\sqrt{2}}{2}(1 + z^{-1}\widetilde{F}(z^2)), & G^0(z) &\triangleq \frac{\sqrt{2}}{2}(1 + z^{-1}F(z^2)), \\ \widetilde{G}^1(z) &= \frac{z^{-1}\widetilde{w}(z^2)}{\sqrt{2}}, & G^1(z) &= \frac{z^{-1}w(z^2)}{\sqrt{2}}, \\ \widetilde{G}^2(z) &= \frac{1 - z^{-1}\widetilde{F}(z^2)}{\sqrt{2}} = \widetilde{H}(-z), & G^2(z) &= \frac{1 - z^{-1}F(z^2)}{\sqrt{2}} = H(-z). \end{aligned} \tag{1.83}$$

Note that in this case, the filters $G^2(z)$ and $\widetilde{G}^2(z)$ are interpolatory. If $F(1) = \widetilde{F}(1)$, then these filters are highpass. The filters $\widetilde{G}^1(z)$ and $G^1(z)$ have no even polyphase component.

This filter bank generates a biframe in signal space.

We use, as the prediction filters $\widetilde{F}(z)$ and $F(z)$, the spline-based filters $V_{1,2}^p(z)$ and $U_{1,2}^{2m}(z)$, which are designed in Section 1.3. All these filters possess the symmetry property: $z^{-1}F(z^2) = zF(z^{-2})$. Thus, the filters $\widetilde{G}^0(z)$, $\widetilde{G}^2(z)$, $G^0(z)$, and $G^2(z)$ are symmetric about inversion $z \to 1/z$. The rational function $W(z^2)$ can be written as $W(z^2) = (1 - z^{-1}F(z^2) \cdot z\widetilde{F}(z^{-2}))/2 = W(z^{-2})$. Thus, a rational symmetric or antisymmetric factorization is possible. The trivial rational symmetric factorizations are $\nu(z) = 1$, $\widetilde{\nu}(z) = W(z)$ or $\widetilde{\nu}(z) = 1$, $\nu(z) = W(z)$. Since $W(z^2) = 0$ if $z = \pm 1$, at least one of the filters $G^2(z)$ and $\widetilde{G}^2(z)$ is bandpass and the corresponding framelet has vanishing moments.

1.6.2. Tight and semitight frames

If $F(z) = \widetilde{F}(z)$, then we get $G^0(z) = \widetilde{G}^0(z)$, $G^2(z) = \widetilde{G}^2(z)$, and

$$W(z) = \frac{1 - |F(z)|^2}{2}, \qquad W(z^2) = 2G^0(z)G^0(-z). \tag{1.84}$$

If the inequality

$$|F(z)| \le 1, \quad \text{as } |z| = 1, \tag{1.85}$$

holds, then the function $W(z)$ can be factorized as $W(z) = w(z)w(1/z)$. This factorization is not unique. Due to Riesz's lemma [19], a rational factorization is possible. Then, we have $G^1(z) = \widetilde{G}^1(z)$. Thus, the synthesis filter bank coincides with the analysis filter bank and generates a tight frame. Due to (1.84), the (anti)symmetric rational factorization is possible if and only if all roots and poles of the function $G^0(z)$ have even multiplicity. If $G^0(z)$ has a root of multiplicity $2m$ at $z = 1$, then the filter $G^1(z)$ has roots of multiplicity m at $z = 1$ and $z = -1$. The corresponding framelet $\psi^1(t)$ has m vanishing moments. A similar construction for the tight frame based on a family of interpolatory symmetric FIR filters was presented in [14]. However, the filter $G^1(z)$ in [14] lacks symmetry.

If the condition (1.85) is not satisfied, we are still able to generate frames, which are very close to a tight frame. Namely,

$$\begin{aligned} G^0(z)(z) = \widetilde{G}^0(z)(z) &= \frac{1 + z^{-1}F(z^2)}{\sqrt{2}}, \\ G^2(z) = \widetilde{G}^2(z) &= \frac{1 - z^{-1}F(z^2)}{\sqrt{2}}, \\ G^1(z) = \frac{z^{-1}w(z^2)}{\sqrt{2}}, \qquad \widetilde{G}^1(z) &= \frac{z^{-1}\widetilde{w}(z^2)}{\sqrt{2}}, \\ w(z)\widetilde{w}\left(\frac{1}{z}\right) = W(z) &= \left(1 - |F(z)|^2\right). \end{aligned} \tag{1.86}$$

It is natural to refer to such a frame as a *semitight* frame. Due to the symmetry of $W(z)$, an (anti)symmetric factorization of type (1.86) is always possible. Therefore, even when (1.85) holds, sometimes it is preferable to construct a semitight rather than a tight frame. For example, it was proved in [38] that a compactly supported interpolatory symmetric tight frame with two framelets is possible only with the lowpass filter $G^0(z) = (1 + (z + 1/z)/2)/\sqrt{2}$. In this case, the scaling function and the framelets are piecewise linear. The framelets $\psi^2(t)$ and $\psi^1(t)$ have two and one vanishing moments, respectively. However, it is possible to construct a variety of compactly supported interpolatory symmetric semitight frames with smooth framelets. The construction of compactly supported interpolatory symmetric tight frame with three framelets is always possible [14]. On the other hand, in the semitight frames, we can swap vanishing moments between the analysis and

synthesis framelets. Typically, it is advisable to have more vanishing moments in the analysis framelets.

Note that smoothness of all waveforms in a tight or semitight frame is the same.

1.6.3. Tight and semitight frames generated by spline-based filter banks

1.6.3.1. Butterworth frames

If we use as the prediction filters $F(z)$, the filters $U_{1,2}^{2m}(z)$, which are based on the discrete splines, then we can explicitly design tight and semitight frames, where the waveforms are (anti)symmetric and have arbitrary smoothness. The framelets may have any number of vanishing moments. Since there is a relation between the filters and the Butterworth filters, we call the corresponding frames the Butterworth frames. Let us denote $\rho(z) \triangleq z + 2 + z^{-1}$.

Let $F(z) = U_{1,2}^{2m}(z)$. Then,

$$\begin{aligned} G^0(z) = \widetilde{G}^0(z) &= \frac{1 + z^{-1} U_{1,2}^{2m}(z^2)}{\sqrt{2}} = \frac{\sqrt{2}\rho^r(z)}{\rho^r(z) + \rho^r(-z)}, \\ G^2(z) = \widetilde{G}^2(z) &= \frac{1 - z^{-1} U_{1,2}^{2m}(z^2)}{\sqrt{2}} = \frac{\sqrt{2}\rho^r(-z)}{\rho^r(z) + \rho^r(-z)}. \end{aligned} \tag{1.87}$$

We get a tight frame when we factorize $W(z)$ to be

$$W(z) = 1 - \left| U_{1,2}^{2m}(z) \right|^2 = w(z) w\left(\frac{1}{z}\right). \tag{1.88}$$

From (1.84), we have

$$\begin{aligned} W(z^2) = 2G^0(z)G^0(-z) &= \frac{4(-1)^m z^{-2m} (1 - z^2)^{2m}}{\left(\rho^m(z) + \rho^m(-z)\right)^2} = w(z^2) w(z^{-2}), \\ w(z^2) &\triangleq \frac{2(1 - z^2)^m}{\rho^m(z) + \rho^m(-z)}. \end{aligned} \tag{1.89}$$

If $m = 2n$, then we can define $w(z^2)$ differently:

$$w(z^2) \triangleq \frac{2(z - z^{-1})^{2n}}{\rho^{2n}(z) + \rho^{2n}(-z)}. \tag{1.90}$$

Hence, the three filters $G^0(z)$, defined in (1.87), $G^2(z) = H(-z)$, and $G^1(z) \triangleq z^{-1} w(z^2)/\sqrt{2}$, where $w(z^2)$ is defined in (1.89), generate a tight frame in the signal space. The scaling function $\varphi(t)$ and the framelet $\psi^2(t)$ are symmetric, whereas the framelet $\psi^1(t)$ is symmetric when m is even and antisymmetric when m is odd. The framelet $\psi^2(t)$ has $2m$ vanishing moments and the framelet $\psi^1(t)$ has m

vanishing moments. The frequency response of the filter $G^0(z)$ is maximally flat. The frequency response of the filter $G^2(z)$ is a mirrored version of $G^0(z)$. The frequency response of the filter $G^1(z)$ is symmetric about $\omega = \pi/2$ and it vanishes at the points $\omega = 0 (z = 1)$ and $\omega = \pi (z = -1)$.

We have more freedom in the design of the filters $\widetilde{G}^1(z)$ and $G^1(z)$ if we drop the requirement $w(z) = \widetilde{w}(z)$ in the factorization of $W(z)$. Doing so, we arrive at the semitight case. For example, a symmetric factorization of $W(z)$ is possible for either of even and odd values of m:

$$w(z^2) \triangleq \frac{2\left(2 - z^{-2} - z^2\right)^p}{\left(\rho^m(z) + \rho^m(-z)\right)^s}, \quad \widetilde{w}(z^2) \triangleq \frac{\sqrt{2}\left(2 - z^{-2} - z^2\right)^{m-p}}{\left(\rho^m(z) + \rho^m(-z)\right)^{2-s}}, \quad s \in \mathbb{Z}. \tag{1.91}$$

We can get an antisymmetric factorization by choosing an odd p:

$$w(z^2) \triangleq -\frac{2\left(-z^2\right)^{-m}\left(1 - z^2\right)^p}{\left(\rho^m(z) + \rho^m(-z)\right)^s}, \quad \widetilde{w}(z^2) \triangleq \frac{2\left(-z^2\right)^{p-2m}\left(1 - z^2\right)^{2m-p}}{\left(\rho^m(z) + \rho^m(-z)\right)^{2-s}}, \quad s \in \mathbb{Z}. \tag{1.92}$$

With this factorization, we can change the number of vanishing moments in the framelets $\psi^1(t)$ and $\widetilde{\psi}^1(t)$. One option is that one of the filters $G^1(z) = z^{-1}w(z^2)$ or $\widetilde{G}^1(z) = z^{-1}\widetilde{w}(z^2)$ has a finite impulse response. This is achieved if $s \leq 0$ or $s \geq 2$.

1.6.3.2. Frames based on continuous splines

When we use as the prediction filters $F(z)$ the filters $V^p_{1,2}(z)$, which are based on the continuous splines, then the design of tight frames with (anti)symmetric filters $G^1(z)$ is possible only for the cases $p = 2, 3$ (linear and quadratic splines). For the higher orders, the semitight construction is recommended in order to retain symmetry.

We provide examples of filter banks that generate tight and semitight frames.

Example 1.15

Linear spline, $p = 2$:

$$G^0(z) = \frac{z^{-1} + 2 + z}{2\sqrt{2}}, \qquad G^2(z) = \frac{-z^{-1} + 2 - z}{2\sqrt{2}}, \qquad G^1(z) = \frac{\left(z^{-1} - z\right)}{2}. \tag{1.93}$$

The filters are FIR, and therefore, the scaling function $\varphi(t)$ and the framelets $\psi^1(t)$ and $\psi^2(t)$ are compactly supported. The framelet $\psi^2(t)$ has two vanishing moments. The framelet $\psi^1(t)$ is antisymmetric and has one vanishing moment. The scaling function $\varphi(t)$ is the fundamental linear spline.

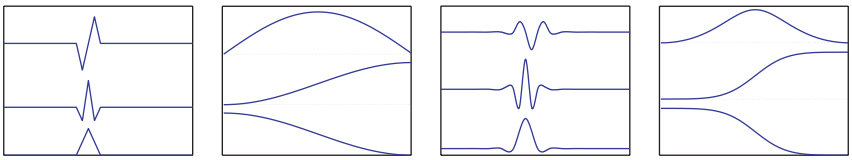


FIGURE 1.13. Filters and framelets for tight frames resulting from the linear spline (two leftmost columns) and continuous quadratic spline (two rightmost columns).

Quadratic continuous spline, $p = 3$ (cubic discrete spline, $m = 2$):

$$G^0(z) = \frac{(z+2+z^{-1})^2}{\sqrt{2}(z^{-2}+6+z^2)}, \qquad G^1(z) = \frac{(z-2+z^{-1})^2}{\sqrt{2}(z^{-2}+6+z^2)}, \tag{1.94}$$
$$G^2(z) = \frac{2z^{-1}(z-z^{-1})^2}{z^{-2}+6+z^2}.$$

The framelet $\psi^2(t)$ has four vanishing moments. The framelet $\psi^1(t)$ is symmetric and has two vanishing moments. All waveforms belong to C^2.

We display in Figure 1.13 filters and framelets for tight frames resulting from linear spline and the continuous quadratic spline. The four rows on the bottom depict the scaling functions $\varphi(t)$ and the frequency response of the lowpass filters $G^0(z)$, the central four rows display the highpass filters $G^2(z)$ and the framelets $\psi^2(t)$, and the upper four rows depict the bandpass filters $G^1(z)$ and the corresponding framelets $\psi^1(t)$.

Discrete spline of sixth order, $m = 3$:

$$G^0(z) = \frac{(z^{-1}+2+z)^3}{\sqrt{2}(6z^2+20+6z^{-2})}, \qquad G^2(z) = G^0(-z), \qquad G^1(z) = \frac{2z^{-1}(1-z^2)^3}{6z^2+20+6z^{-2}}. \tag{1.95}$$

The framelet $\psi^2(t)$ has six vanishing moments. The framelet $\psi^1(t)$ is antisymmetric and has three vanishing moments. All waveforms belong to C^3.

Discrete spline of eighth order, $m = 4$:

$$G^0(z)\frac{(z^{-1}+2+z)^4}{\sqrt{2}(z^{-4}+28z^{-2}+70+28z^2+z^4)}, \qquad G^2(z) = G^0(-z), \tag{1.96}$$
$$G^1(z) = \frac{2z^{-1}(z-z^{-1})^4}{z^{-4}+28z^{-2}+70+28z^2+z^4}.$$

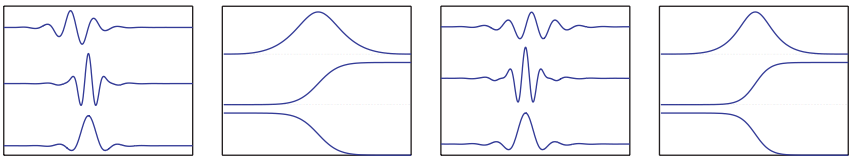


FIGURE 1.14. Filters and framelets for tight frames resulting from the discrete spline of sixth order (two leftmost columns) and the discrete spline of eighth order (two rightmost columns).

The framelet $\psi^2(t)$ has eight vanishing moments. The framelet $\psi^1(t)$ is symmetric and has four vanishing moments. All waveforms belong to C^3.

We display in Figure 1.14 filters and framelets for tight frames resulting from discrete splines of sixth and eighth orders. The figure is organized similarly to Figure 1.13.

Continuous cubic spline, $p = 4$:

$$G^0(z) = \frac{(z^{-1} + 2 + z)^2 (z + 4 + z^{-1})}{8\sqrt{2}(z^{-2} + 4 + z^2)}, \qquad G^2(z) = G^0(-z). \tag{1.97}$$

To obtain the filter $G^0(z)$, the function

$$W(z^2) = 2G^0(z)G^0(-z) = \frac{(z^{-1} - z)^4 (z^2 - 14 + z^{-2})}{8(z^{-2} + 4 + z^2)^2} \tag{1.98}$$

has to be factorized. The factorization $W(z) = w(z)w(1/z)$, which leads to tight frame, results in the filter

$$G^1(z) = z^{-1} \frac{(z^{-1} - z)^2 (1 - qz^2)}{8q\sqrt{2}(z^{-2} + 4 + z^2)}, \tag{1.99}$$

where $q = 7 - 4\sqrt{3} \approx 0.0718$. The filter is not symmetric about inversion $z \to 1/z$, and consequently, the framelet $\psi^1(t)$ is slightly asymmetric. It has two vanishing moments, whereas the framelet $\psi^2(t)$ has four vanishing moments.

We display in Figure 1.15 filters and framelets for tight frame resulting from the continuous cubic spline. The first from the left frame displays the waveforms, the second frame displays the frequency response of the filter bank, and the last frame provides the closed-up view to framelet $\psi^1(t)$ in order to highlight its asymmetry.

The factorization $W(z) = w(z)\widetilde{w}(1/z)$ with unequal factors $w(z)$ and $\widetilde{w}(z)$ leads to a semitight frame. We present the filters $\widetilde{G}^1(z)$ and $G^1(z)$ that result from

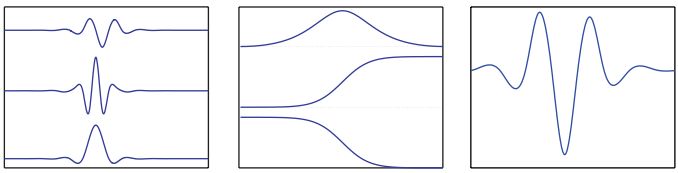


FIGURE 1.15. Filters and framelets for tight frames resulting from the continuous cubic spline (two leftmost columns) and the closed-up view of the framelet $\psi^1(t)$ (rightmost column).

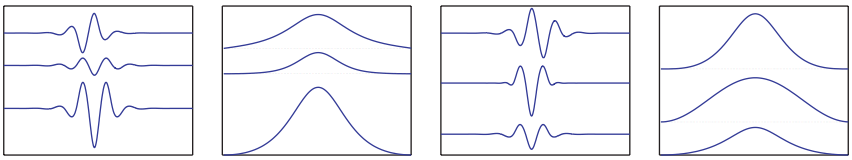


FIGURE 1.16. Filters and framelets for semitight frames resulting from the continuous cubic spline, that correspond to (1.100) (bottom row), (1.101) (central row) and (1.102) (top row).

three different factorization modes:

(1) symmetric factorization with equal number (two) of vanishing moments in the analysis $\widetilde{\psi}^1(t)$ and synthesis $\psi^1(t)$ framelets:

$$\widetilde{G}^1(z) = z^{-1}\frac{(z^{-1}-z)^2}{8\sqrt{2}(z^{-2}+4+z^2)}, \qquad G^1(z) = z^{-1}\frac{(z^{-1}-z)^2(-z^2+14-z^{-2})}{8\sqrt{2}(z^{-2}+4+z^2)}; \tag{1.100}$$

(2) symmetric factorization with equal number (two) of vanishing moments in the analysis $\widetilde{\psi}^1(t)$ and synthesis $\psi^1(t)$ framelets. The analysis filter $\widetilde{G}^1(z)$ is FIR, and consequently, the framelet $\widetilde{\psi}^1(t)$ is compactly supported:

$$\widetilde{G}^1(z) = z^{-1}\frac{(z^{-1}-z)^2}{8\sqrt{2}}, \qquad G^1(z) = z^{-1}\frac{(z^{-1}-z)^2(-z^2+14-z^{-2})}{8\sqrt{2}(z^{-2}+4+z^2)^2}; \tag{1.101}$$

(3) antisymmetric factorization with three vanishing moments in the analysis $\widetilde{\psi}^1(t)$ and one vanishing moment in synthesis $\psi^1(t)$ framelets:

$$\widetilde{G}^1(z) = \frac{(z^{-1}-z)^3}{8\sqrt{2}(z^{-2}+4+z^2)}, \qquad G^1(z) = \frac{(z^{-1}-z)(-z^2+14-z^{-2})}{8\sqrt{2}(z^{-2}+4+z^2)}. \tag{1.102}$$

We recall that the scaling function $\varphi(t)$ is the fundamental cubic spline and all the waveforms in the tight and semitight frames are cubic splines.

We display in Figure 1.16 the filters $\widetilde{G}^1(z)$ and $G^1(z)$ and framelets $\widetilde{\psi}^1(t)$ and $\psi^1(t)$ that result from the above modes of factorization. The four plots on the

bottom depict the analysis framelets $\widetilde{\psi}^1(t)$, the frequency response of the analysis filter $\widetilde{G}^1(z)$, the synthesis framelets $\psi^1(t)$, the frequency response of the synthesis filter $G^1(z)$, respectively, where the filters are given in (1.100). The central four plots do the same for the case (1.101) and the upper four correspond to (1.102).

Remarks 1.16. (1) From the above figures, we see that the designed waveforms are smooth, well localized in time domain, and symmetric. They are as smooth as the synthesis waveforms in the biorthogonal wavelet transforms originating from the same prediction filter and are smoother than the analysis waveforms.

(2) The frequency responses of the filters are maximally flat (has no ripples) in the passband, and roll towards zero in the stopband. As the order of the generating spline is high, the decline is steeper in the stopband.

(3) The frequency responses of the lowpass filter $G^0(z)$ and the highpass filter $G^2(z)$ are monotonous and are the mirrored versions of each other. The bumps, which are present in the filters $\widetilde{H}^0(z)$ and $H^1(z)$ of the wavelet filter bank, are compensated by the bandpass filter $G^1(z)$.

(4) The transfer functions of all designed filters are rational functions. Therefore, the filters (except for the filters derived from the linear spline) have infinite impulse response (IIR). But they can be efficiently implemented via causal-anticausal recursive filtering (see the appendix).

1.7. Erasure recovery properties of the designed wavelet frames

In this section, we outline a scheme that applies the designed wavelet frames to recovery of erasures, which occur when a multimedia signal is transmitted through a lossy channel. This is described in [1].

Conventional methods for protecting data are well developed both in theory and practice. Block and convolutional codes are considered to be very efficient classes as channel codes. They are widely used in wireless and wireline channels such as the Internet. However, these codes, and other conventional methods, do not generally take into account the inner structure of the transmitted (multimedia) signal. Rather, it is assumed that every bit is equally significant, and hence it has to be equally protected. Multimedia information usually undergoes some transform (e.g., DCT, FFT, or wavelet) followed by quantization and lossless compression (entropy coding). The last two operations constitute source coding. The resulting binary sequence (which assumed to be white noise) typically contains bits with unequal significance, which must be protected accordingly. Due to this inhomogeneity, direct application of channel coding methods to audio-visual information is not optimal; it may significantly increase the transmission length if the (equal) error correction code is chosen according to the most vulnerable data. Hence, it is desired to design error correction codes that dynamically allocate more bits to more important information. Such codes are known as unequal error protection (UEP) codes. Due to the growing importance in rich multimedia data transmission, unequal error protection methods have attracted research efforts, see, for example, [30] and the references therein. The designed framelet transform can be

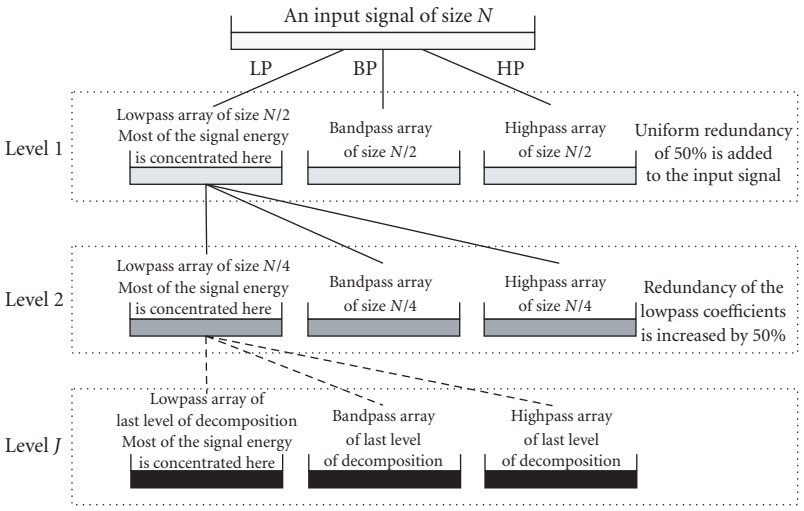


FIGURE 1.17. Diagram of multiscale frame transform with 3-channel filter bank and downsampling factor of 2.

effectively employed as a true combined source-channel coding scheme—there is no separate source coding followed by channel coding. In fact, no explicit channel coding is used. The proposed approach makes use of naturally occurring redundancy within multiscale decomposition of framelet transforms to provide unequal error protection (UEP).

1.7.1. Remarks on unequal error protection

The multiscale frame transform, which is described in Section 1.2.3, is demonstrated schematically in Figure 1.17.

Assume that there are four scales of decomposition. Figure 1.18 displays the spectra of the discrete-time framelets $\psi^{r,1}$ $\psi^{r,2}$, $r = 1, 2, 3, 4$, and φ^4 that originate from the filter bank (1.94). The shifts of these framelets provide a four-scale tight frame expansion of the signal. First scale of decomposition produces three blocks of coefficients: lowpass, bandpass and highpass. As it was explained in Section 1.2.3, these are the coefficients of the orthogonal projections of the signal onto the subspaces spanned by two-sample shifts of the discrete-time framelets $\varphi^1(k)$, $\psi^{1,2}(k)$, and $\psi^{1,1}(k)$, respectively. The spectra of the framelets $\psi^{1,2}(k)$ and $\psi^{1,1}(k)$ are displayed in the top row of Figure 1.18. The second step of the decomposition transforms the lowpass block into three blocks of coefficients, which are the coefficients of the orthogonal projections of the signal onto the subspaces spanned by four-sample shifts of the framelets $\varphi^2(k)$, $\psi^{2,2}(k)$, and $\psi^{2,1}(k)$. The spectra of the framelets $\psi^{2,2}(k)$ and $\psi^{2,1}(k)$ are displayed in the second from the top row of the figure. The last fourth step of the decomposition transforms the lowpass block of the third scale into three blocks of coefficients, which are the coefficients of the orthogonal projections of the signal onto the subspaces spanned

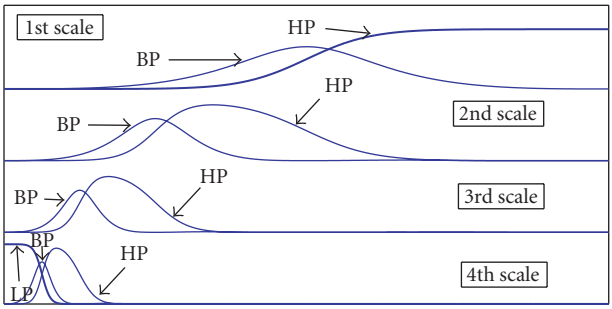


FIGURE 1.18. Spectra of the discrete-time framelets $\psi^{r,1}$, $\psi^{r,2}$, $r = 1, 2, 3, 4$, and φ^4 that originate from the filter bank (1.94). The abbreviation LP means lowpass, it is related to φ^4, HP means highpass, it is related to $\psi^{r,1}$, BP means bandpass, it is related to $\psi^{r,2}$.

by sixteen-sample shifts of the framelets $\varphi^4(k)$, $\psi^{4,2}(k)$, and $\psi^{4,1}(k)$. The spectra of these framelets are displayed in the bottom row of the figure. The reconstruction consists of the synthesis of the original signal from the above set of projections.

We see that the spectra displayed in Figure 1.18 form at least twofold cover of the frequency domain of the signal except for the frequency bands occupied by the spectra of the low-frequency framelet φ^4 and the high-frequency framelet $\psi^{1,1}$. They are highlighted by boldface lines in Figure 1.18. It means that once a projection (except for the projections on φ^4 and $\psi^{1,1}$) is lost, it can be restored from the remaining projections. Also two or more projections, whose spectra do not overlap, can be restored. In other words, erasure of a number of coefficients from a block or even the whole block (except for the blocks related to φ^4 and $\psi^{1,1}$) can be compensated by the coefficients from the remaining blocks.

Two exclusive blocks of coefficients related to φ^4 and $\psi^{1,1}$ must also be protected. The lowpass block is the most significant. Erasure of even one coefficient can essentially distort the signal. But for the four-scale transform, it comprises only $N/16$ coefficients, where N is the length of the signal. If we expand the transform to scale J, then the last lowpass block comprises only $N/2^J$ coefficients. This relatively small number of coefficients can also be protected at a low computational cost.

The highpass block related to $\psi^{1,1}$ is most populated ($N/2$ coefficients). Due to the vanishing moments of the framelets $\psi^{1,1}$, this block contains relatively small number of significant coefficients, which correspond to sharp transients in the signal (edges in the image). Only these significant coefficients deserve an additional protection.

1.7.2. Outline of the recovery algorithm

Assume that the original 2D image is arranged into a 1D array $X \in \mathbb{R}^N$ and the coefficients of its 2D framelet transform are arranged into a 1D array $Y \in \mathbb{H} \subset \mathbb{R}^K$ of length $K > N$. Let $\mathcal{J} \triangleq \{C_k\}_1^{nm}$ be the set of coordinates of this array, and let $E \subset \mathcal{J}$ be the set of coordinates of the erased coefficients. The subspace $\mathbb{H} \subset \mathbb{R}^K$ is called

Initialize $y^{(0)} = \widetilde{Y}$;
for $k = 0$ to $K - 1$
$\hat{x}^{(k)} = Fy^{(k)}$; *fit out-of-interval values into* $[L_0, L_{255}]$;
$\hat{y}^{(k)} = \widetilde{F} * \hat{x}^{(k)}$;
$y^{(k+1)} = \hat{y}^{(k)}$ *on the coordinates of* E;
$y^{(k+1)} = \widetilde{Y}$ *on the coordinates of* $\overline{E}$;
end.

Algorithm 1.1

the space of codewords. Define $\overline{E} \triangleq \mathcal{S} \setminus E$ and $\widetilde{Y}$ is obtained from Y by erasing all coefficients that correspond to E. Let $\widetilde{F}$ be the analysis operator $\widetilde{F} : \mathbb{R}^N \mapsto H \subset \mathbb{R}^K$, $K > N$ associated with the framelet transform: $Y = \widetilde{F}X$. Obviously, $\operatorname{rank}(\widetilde{F}) = N$. Let F be the inverse operator (i.e., the synthesis operator) of $\widetilde{F}$. We denote by $\widetilde{\widetilde{F}}$ the matrix $\widetilde{F}$ with erased rows determined by the coordinates of E. Assume that $\widetilde{Y}$ contains zeros instead of all the erased coefficients in $\widetilde{Y}$. If $\operatorname{rank}(\widetilde{\widetilde{F}}) = N$, then $\widetilde{Y}$ contains sufficient information to recover the original data X.

Each coefficient of the transformed image is presented by few bits. If one or more bits associated with the same coefficient are lost in transit, the whole coefficient may be treated as an erasure, or alternatively, as being in error. It is well known that, in general, recovering from erasures is easier than recovering from errors. Hence the motivation for the algorithm stems. This algorithm is a slightly modified version of the well-known Gerchberg [22] and Papoulis [35] algorithm. The Gerchberg-Papoulis algorithm was applied, in particular, to interpolation of data given on an irregular grid. The application of the mentioned algorithm to erasure recovery was reported in [37].

It utilizes the redundancy inherent in frame transforms to recover from erasures of whole coefficients that occur during transmission. As before, $\widetilde{Y}$ denotes the received set of coefficients with the convention that erased coefficients are substituted by zeros. Let y^k denote the set of (received+recovered) framelet coefficients at iteration k of the recovery algorithm. Assume the image intensities belong to the interval $[L_0, L_{255}]$, where $L_0 < L_{255}$.

This framelet-based algorithm Algorithm 1.1 iteratively recovers an image from its transformed version $\widetilde{Y}$ that contains erasures. The recovered image at each iteration is given by $\hat{x}^{(k)}$.

1.7.3. Experimental results

We conducted a series of experiments on image recovery from erasures of the transform coefficients. This can be regarded as a simulation of channels with erasures. To be specific, we applied the framelet decomposition up to the fourth level. The redundancy factor of this decomposition is 2.66. Then $\alpha \cdot 100\%$ of the transform coefficients, whose locations were randomly chosen, were put to zero. We restored the images using the iterative algorithm described in Section 1.7.2. We

Table 1.5. Averaged PSNR values of the four reconstructed images using symmetric tight frame (1.94), antisymmetric tight frame (1.95), and semitight frame (1.97) and (1.102).

Erasure	10%	20%	30%	40%	50%	60%	70%
PSNR/S-TF	51.8418	50.7470	49.0475	46.3734	40.7849	32.3740	19.2204
PSNR/symm. TF	52.0012	51.3969	50.0345	47.9709	43.6514	32.9655	19.7563
PSNR/antisymm. TF	52.2622	51.3204	50.2554	48.2412	43.1816	32.8288	19.5409

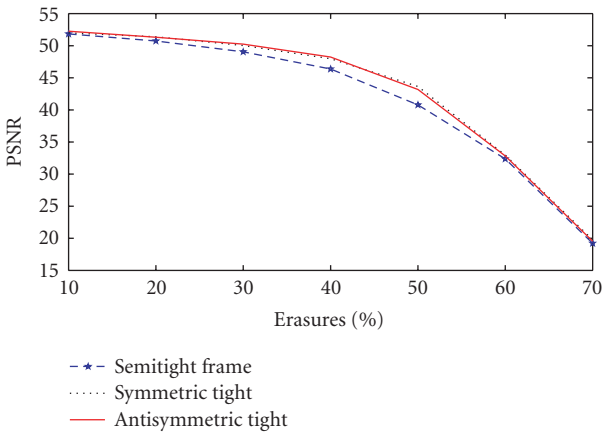
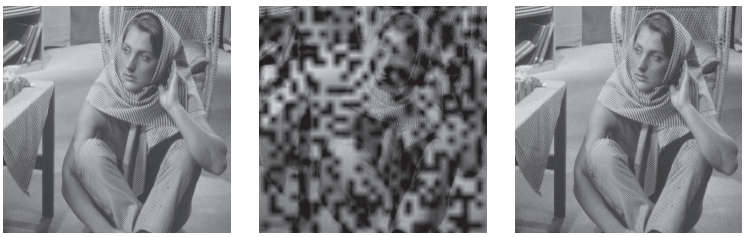


Figure 1.19. Averaged PSNR of the reconstructed images versus coefficient erasure probability.

tested two benchmark images *Barbara* and *boats* and two medical images taken by MRI scanner while using $\alpha = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7$. Three different types of framelets were tested: the symmetric tight framelets originating from quadratic spline (1.94), tight frame originating from discrete spline of sixth order, where $\psi^1(t)$ is antisymmetric, (1.95), and the semitight frame originating from cubic spline (1.97) and (1.102), where $\widetilde{\psi}^1(t)$ and $\psi^1(t)$ are antisymmetric. The distance between the original and the restored images was evaluated via PSNR (see (1.77)).

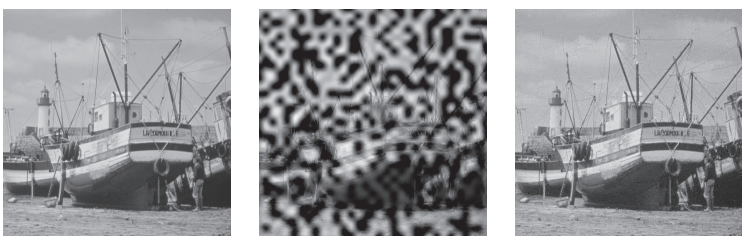
The experimental results are summarized in Table 1.5 and are illustrated by Figure 1.19. The results for all the tested images are similar to each other, therefore, for brevity, we present PSNR values that are averaged over the four images. The results demonstrate a graceful degradation in performance when the erasure probability of the coefficients increases to 0.7. The performance of the symmetric and antisymmetric tight frames is almost identical, while the biframe produces images with a slightly lower PSNR.

In addition, we display the four restored images in Figures 1.20–1.23. All the figures are similarly organized. Each of them comprises three columns. The left column displays the original image, the middle column is the corrupted image, the right column displays the reconstructed image from the corrupted transform coefficients. We observe from the images that the restoration scheme, that is based on the wavelet frames, produces satisfactory output even for 60 percent of randomly



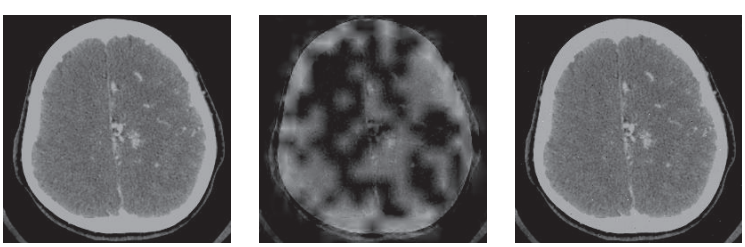
(a) (b) (c)

FIGURE 1.20. Results from the application of the antisymmetric tight framelet transform. (a) Original image, (b) the corrupted image with 60% erased coefficients, (c) the recovered image. PSNR = 32.24.



(a) (b) (c)

FIGURE 1.21. Results from the application of the symmetric tight framelet transform. (a) The source image, (b) the corrupted image with 60% erased coefficients, (c) the recovered image. PSNR = 31.98.



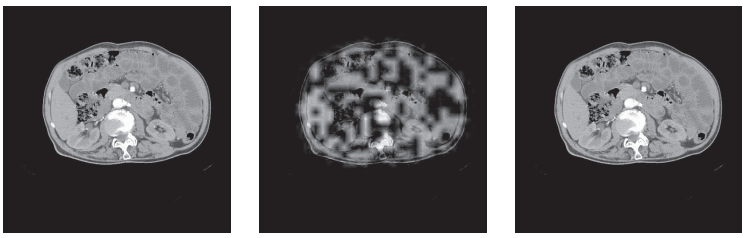
(a) (b) (c)

FIGURE 1.22. Results from the application of symmetric semitight framelet transform. (a) The source image, (b) the corrupted image with 50% erased coefficients, (c) the recovered image. PSNR = 43.354.

erased coefficients. For 50 and, especially, for 40 percents of erased coefficients, the restored images hardly can be distinguished from the original source images.

1.8. Biorthogonal wavelet transforms generated by filter banks with downsampling factor $N = 3$ (triadic transforms)

In this section, we expand the method that constructs biorthogonal wavelet transforms, which was described in Section 1.4, to the case when the downsampling factor of the filter banks is $N = 3$. As in Section 1.4, the construction is carried out via the lifting scheme.



(a) (b) (c)

FIGURE 1.23. Results from the application of the semitight framelet transform. (a) The source image, (b) the corrupted image with 40% erased coefficients, (c) the recovered image. PSNR = 52.461.

1.8.1. Decomposition

1.8.1.1. Lifting steps

Split: we split the signal $\mathbf{x}$ into three subarrays:

$$\mathbf{x}_{-1} \triangleq \{x_{3l-1}\}_{l\in\mathbb{Z}}, \qquad \mathbf{x}_0 \triangleq \{x_{3l}\}_{l\in\mathbb{Z}}, \qquad \mathbf{x}_1 \triangleq \{x_{3l+1}\}_{l\in\mathbb{Z}}. \tag{1.103}$$

Let $X_{-1}(z)$, $X_0(z)$, and $X_1(z)$ denote the z-transforms of these subarrays.

Predict: we predict

$$\check{D}_{-1}(z) = X_{-1}(z) - \widetilde{F}_{-1}(z^{-1})X_0(z), \qquad \check{D}_1(z) = X_1(z) - \widetilde{F}_1(z^{-1})X_0(z). \tag{1.104}$$

Here, $\widetilde{F}_{-1}$ and $\widetilde{F}_1$ are some filters that are applied to the subarray $\mathbf{x}_0$ in order to predict the subarrays $\mathbf{x}_{-1}$ and $\mathbf{x}_1$, respectively. Then, these latter subarrays are replaced by the differences $\check{\mathbf{d}}_{-1}$ and $\check{\mathbf{d}}_1$ between initial and predicted values.

Update: the array $\mathbf{x}_0$ is smoothed using the arrays $\check{\mathbf{d}}_{-1}$ and $\check{\mathbf{d}}_1$, which are processed by the update filters F_{-1} and F_1, respectively,

$$D_0(z) = X_0(z) + \frac{1}{3}F_{-1}(z)\check{D}_{-1}(z) + \frac{1}{3}F_1(z)\check{D}_1(z). \tag{1.105}$$

Symmetrization: the filters $V^p_{\mp 1,3}$ and $U^p_{\mp 1,3}$, derived from the splines in Section 1.3, are used as prediction and update filters. These filters are not symmetric, unlike their sums and differences. Therefore, we apply two additional lifting steps:

$$D_1(z) = \frac{\check{D}_1(z) - \check{D}_{-1}(z)}{2}, \qquad D_{-1}(z) = \check{D}_{-1}(z) + D_1(z) = \frac{\check{D}_1(z) + \check{D}_{-1}(z)}{2}. \tag{1.106}$$

1.8.1.2. Polyphase matrix

The above lifting steps can be represented via the matrix-vector multiplication

$$\begin{pmatrix} D_1(z) \\ D_0(z) \\ D_{-1}(z) \end{pmatrix} = \widetilde{\mathbf{P}}(z^{-1}) \cdot \begin{pmatrix} X_1(z) \\ X_0(z) \\ X_{-1}(z) \end{pmatrix}, \tag{1.107}$$

where

$$\begin{aligned} \widetilde{\mathbf{P}}(z^{-1}) &= \begin{pmatrix} \frac{1}{2} & 0 & \frac{-1}{2} \\ 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ \frac{F_1(z)}{3} & 1 & \frac{F_{-1}(z)}{3} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -\widetilde{F}_1(z^{-1}) & 0 \\ 0 & 1 & 0 \\ 0 & -\widetilde{F}_{-1}(z^{-1}) & 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{2} & \frac{-\widetilde{F}_1(z^{-1})}{2} + \frac{\widetilde{F}_{-1}(z^{-1})}{2} & \frac{-1}{2} \\ \frac{F_1(z)}{3} & 1 - \frac{F_1(z)\widetilde{F}_1(z^{-1})}{3} - \frac{F_{-1}(z)\widetilde{F}_{-1}(z^{-1})}{3} & \frac{F_{-1}(z)}{3} \\ \frac{1}{2} & \frac{-\widetilde{F}_1(z^{-1})}{2} - \frac{\widetilde{F}_{-1}(z^{-1})}{2} & \frac{1}{2} \end{pmatrix}. \end{aligned} \tag{1.108}$$

1.8.1.3. Analysis filter bank

The above operations are equivalent to the application to the signal $\mathbf{x}$ of the time-reversed filter bank

$$\begin{aligned} \widetilde{H}^1(z) &\triangleq \frac{z^{-1}}{2} + \frac{\widetilde{F}_{-1}(z^3) - \widetilde{F}_1(z^3)}{2} - \frac{z}{2}, \\ \widetilde{H}^0(z) &= \frac{z^{-1}}{3} F_1(z^{-3}) + 1 - \frac{F_1(z^{-3})\widetilde{F}_1(z^3) + F_{-1}(z^{-3})\widetilde{F}_{-1}(z^3)}{3} + \frac{z}{3} F_{-1}(z^{-3}), \\ \widetilde{H}^{-1}(z) &= \frac{z^{-1}}{2} - \frac{\widetilde{F}_{-1}(z^3) + \widetilde{F}_1(z^3)}{2} + \frac{z}{2} \end{aligned} \tag{1.109}$$

followed by downsampling factor 3.

1.8.2. Reconstruction

1.8.2.1. Lifting steps

Undo symmetrization:

$$\begin{aligned} \check{D}_{-1}(z) &= D_{-1}(z) - D_1(z), \\ \check{D}_1(z) &= 2D_1(z) + \check{D}_1(z) = D_{-1}(z) + D_1(z). \end{aligned} \tag{1.110}$$

Undo update:

$$X_0(z) = D_0(z) - \frac{1}{3}F_{-1}(z)\check{D}_{-1}(z) - \frac{1}{3}F_1(z)\check{D}_1(z). \tag{1.111}$$

Undo predict:

$$X_{-1}(z) = \check{D}_{-1}(z) + \widetilde{F}_{-1}(z^{-1})X_0(z), \qquad X_1(z) = \check{D}_1(z) + \widetilde{F}_1(z^{-1})X_0(z). \tag{1.112}$$

Undo split:

$$X(z) = zX_{-1}(z^3) + X_0(z^3) + z^{-1}X_1(z^3). \tag{1.113}$$

1.8.2.2. Polyphase matrix

The above lifting steps can be represented via the matrix-vector multiplication,

$$\begin{pmatrix} X_1(z) \\ X_0(z) \\ X_{-1}(z) \end{pmatrix} = \mathbf{P}(z) \cdot \begin{pmatrix} D_1(z) \\ D_0(z) \\ D_{-1}(z) \end{pmatrix}, \tag{1.114}$$

where

$$\begin{aligned} \mathbf{P}(z) &= \begin{pmatrix} 1 & \widetilde{F}_1(z^{-1}) & 0 \\ 0 & 1 & 0 \\ 0 & \widetilde{F}_{-1}(z^{-1}) & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ \dfrac{-F_1(z)}{3} & 1 & \dfrac{-F_{-1}(z)}{3} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1+\widetilde{F}_1(z^{-1})\dfrac{F_{-1}(z)-F_1(z)}{3} & \widetilde{F}_1(z^{-1}) & 1-\widetilde{F}_1(z^{-1})\dfrac{F_{-1}(z)+F_1(z)}{3} \\ \dfrac{F_{-1}(z)-F_1(z)}{3} & 1 & \dfrac{-(F_{-1}(z)+F_1(z))}{3} \\ -1+\widetilde{F}_{-1}(z^{-1})\dfrac{F_{-1}(z)-F_1(z)}{3} & \widetilde{F}_{-1}(z^{-1}) & 1-\widetilde{F}_{-1}(z^{-1})\dfrac{F_{-1}(z)+F_1(z)}{3} \end{pmatrix}. \end{aligned} \tag{1.115}$$

1.8.2.3. Synthesis filter bank

The above operations are equivalent to the application to the upsampled sets of the transform coefficients $\check{D}_{-1}$, $\check{D}_0$, and $\check{D}_1$ of the filter bank

$$H_1(z) \triangleq z^{-1} - z + \frac{(F_{-1}(z^3) - F_1(z^3))(z^{-1}\widetilde{F}_1(z^{-3}) + 1 + z\widetilde{F}_{-1}(z^{-3}))}{3},$$

$$H_0(z^{-1}) \triangleq z^{-1}\widetilde{F}_1(z^{-3}) + 1 + z\widetilde{F}_{-1}(z^{-3}),$$

$$H_{-1}(z^{-1}) = H_1(z) \triangleq z^{-1} + z - \frac{(F_{-1}(z^3) + F_1(z^3))(z^{-1}\widetilde{F}_1(z^{-3}) + 1 + z\widetilde{F}_{-1}(z^{-3}))}{3}. \tag{1.116}$$

Remark 1.17. All the designed filters are linear phase. The scaling functions $\varphi(t)$ and $\widetilde{\varphi}(t)$ and wavelets $\psi^{-1}(t)$ and $\widetilde{\psi}^{-1}(t)$ are symmetric, whereas the wavelets $\psi^1(t)$ and $\widetilde{\psi}^1(t)$ are antisymmetric.

1.8.3. Filters and wavelets originating from splines

In this section, we exploit the filters $V^p_{\mp 1,3}$ and $U^p_{\mp 1,3}$ derived from the splines in Section 1.3 as the prediction and update filters.

Proposition 1.18. *If the prediction and update filters in the lifting scheme are either derived from a continuous spline of* $F_{\pm 1}(1/z) = V^p_{\pm 1,3}(z) = F_{\pm 1}(z)$ *or from a discrete spline of* $F_{\pm 1}(1/z) = U^p_{\pm 1,3}(z) = F_{\pm 1}(z)$*, then the analysis filters* $\widetilde{H}^1(z)$ *and the synthesis filters* $H^1(z)$ *are bandpass, filters* $\widetilde{H}^1(z)$ *and the synthesis filters* $H^1(z)$ *are bandpass, whereas the filters* $\widetilde{H}^{-1}(z)$ *and* $H^{-1}(z)$ *are highpass and the filters* $\widetilde{H}^0(z)$ *and* $H^0(z)$ *are lowpass.*

1.8.3.1. Continuous splines

Proposition 1.19. *If the prediction and update filters in the lifting scheme* $F_{\pm 1}(1/z) = V^p_{\pm 1,3}(z) = F_{\pm 1}(z)$ *are derived from the continuous spline of order* p*, then*

(1) *the analysis filters* $\widetilde{H}^{\pm 1}(z)$ *and synthesis filters* $H^{\pm 1}(z)$ *have zero of multiplicity not less than* p *at* $z = 1$*;*
(2) *the analysis wavelets* $\widetilde{\psi}^{\pm 1}(t)$ *and synthesis wavelets* $\psi^{\pm 1}(t)$ *have not less than* p *vanishing moments;*
(3) *the filter* $\widetilde{H}^0(z) = 1 + \widetilde{\chi}(z)$ *and the filter* $H^0(z) = 3 + \chi(z)$*, where the functions* $\widetilde{\chi}(z)$ *and* $\chi(z)$ *have zero of multiplicity not less than* p *at* $z = 1$*;*
(4) *if* $p = 2m+1$*, then* $\widetilde{H}^{-1}(z)$, $H^{-1}(z)$, $\widetilde{\chi}(z)$*, and* $\chi(z)$ *have zero of multiplicity* $p + 1$ *at* $z = 1$*;*
(5) *if* $p = 2m + 1$*, then* $\widetilde{\psi}^{-1}(t)$ *and* $\psi^{-1}(t)$ *have* $p + 1$ *vanishing moments;*
(6) *if* $p = 2m$*, then* $\widetilde{H}^1(z)$ *and* $H^1(z)$ *have zero of multiplicity* $p + 1$ *at* $z = 1$*;*
(7) *if* $p = 2m$*, then the wavelets* $\widetilde{\psi}^1(t)$ *and* $\psi^1(t)$ *have* $p + 1$ *vanishing moments;*

(8) *the synthesis scaling function $\varphi(t)$ is the fundamental spline of order p defined in* (1.33). *All the synthesis waveforms are splines of order p.*

Example 1.20

Linear spline: from (1.37), the analysis filter bank is

$$\begin{aligned}
\widetilde{H}^1(z) &= \frac{\left(z^{-1}-z\right)^3}{6},\\
\widetilde{H}^0(z) &= 1-\frac{\left(z-2+z^{-1}\right)\left(4z^2+5z+5z^{-1}+4z^{-2}\right)}{27},\\
\widetilde{H}^{-1}(z) &= -\frac{\left(-z^{-1}+2-z\right)\left(z^2+2z+2z^{-1}+z^{-2}\right)}{6}.
\end{aligned} \tag{1.117}$$

The synthesis filter bank is

$$\begin{aligned}
H^1(z) &= \frac{\left(z-z^{-1}\right)\left(z-2+z^{-1}\right)\left(22+11\left(z+z^{-1}\right)+4\left(z^2+z^{-2}\right)+z^3+z^{-3}\right)}{27},\\
H^0(z) &= \frac{\left(z+1+z^{-1}\right)^2}{3} = 3+\frac{(z+4+1/z)(z-1/z)^2}{3},\\
H^{-1}(z) &= \frac{\left(-z^{-1}+2-z\right)\left(16+22\left(z+z^{-1}\right)+10\left(z^2+z^{-2}\right)+4\left(z^3+z^{-3}\right)+z^4+z^{-4}\right)}{27}.
\end{aligned} \tag{1.118}$$

The transfer functions $\widetilde{H}^1(z)$ and $H^1(z)$ have zero of multiplicity 3 as $z=1$. Consequently, the wavelets $\widetilde{\psi}^1(t)$ and $\psi^1(t)$ have three vanishing moments. The wavelets $\widetilde{\psi}^{-1}(t)$ and $\psi^{-1}(t)$ have two vanishing moments. The synthesis scaling function $\varphi(t)$ is the linear fundamental spline. The analysis waveforms have a fractal shape.

We display in Figure 1.24 the frequency responses of the analysis and synthesis filter banks, which originate from the linear spline and corresponding waveforms. Figures 1.24–1.28 are organized identically. Each figure consists of four columns. The first column from the left displays the analysis scaling function $\widetilde{\varphi}(t)$ and the analysis wavelets $\widetilde{\psi}_1(t)$ (top) and $\widetilde{\psi}_{-1}(t)$ (bottom). The second column from the left displays the frequency responses of the analysis lowpass filter $\widetilde{H}^0(z)$ (center) and of the analysis highpass filters $\widetilde{H}^1(z)$ (top) and $\widetilde{H}^{-1}(z)$ (bottom). Next column displays the synthesis scaling function $\varphi(t)$ (center) and the synthesis wavelets $\psi_1(t)$ (top) and $\psi_{-1}(t)$ (bottom). Last column (rightmost) displays the frequency responses of the synthesis lowpass filter $H^0(z)$ (bottom) and the synthesis highpass filters $H^1(z)$ (top) and $H^{-1}(z)$ (center).

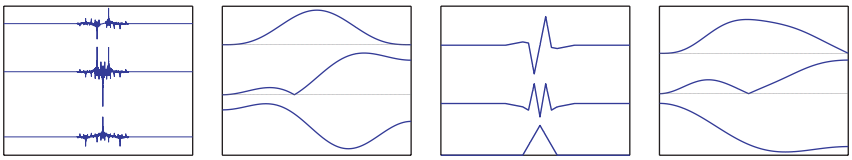


FIGURE 1.24. Filters and wavelets derived from the linear continuous spline.

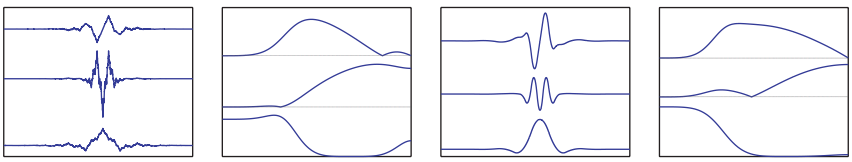


FIGURE 1.25. Filters and wavelets derived from the quadratic continuous spline.

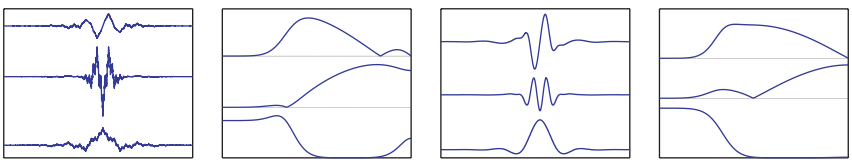


FIGURE 1.26. Filters and wavelets derived from the cubic continuous spline.

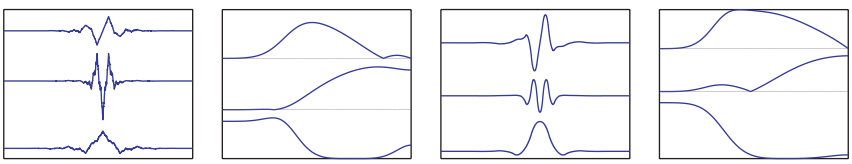


FIGURE 1.27. Filters and wavelets derived from the quadratic discrete spline.

Quadratic spline: from (1.38), we derive the analysis filter bank

$$\widetilde{H}^1(z) = -\frac{(z^{-1}-2+z)(z-z^{-1})(3z^2-2z-7-2z^{-1}+3z^{-2})}{6(z^3+7+z^{-3})},$$

$$\widetilde{H}^0(z) = 1+\frac{(z^{-1}-2+z)^2}{(z^3+7+z^{-3})^2}\sum_{l=0}^{5}\widetilde{c}_l^{\,0}(z^l+z^{-l}), \tag{1.119}$$

$$\widetilde{H}^{-1}(z) = \frac{(z^{-1}-2+z)^2(9z^2+10z-5+10z^{-1}+9z^{-2})}{18(z^3+7+z^{-3})}.$$

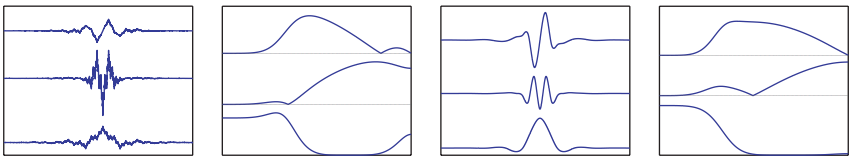


FIGURE 1.28. Filters and wavelets derived from the cubic discrete spline.

The synthesis filter bank is

$$H^1(z) = -\frac{(z^{-1}-2+z)(z-z^{-1})}{(z^3+6+z^{-3})^2}\sum_{l=0}^{5} c_l^1 (z^l + z^{-l}),$$

$$H^0(z) = \frac{(z+6+z^{-1})(z+1+z^{-1})^3}{9(z^3+6+z^{-3})} = 3+\frac{(z-z^{-1})^4(z^2-14z-37-14z^{-1}+z^{-2})}{9(z^3+6+z^{-3})},$$

$$H^{-1}(z) = \frac{2(z^{-1}-2+z)^2}{(z^3+6+z^{-3})^2}\sum_{l=0}^{5} c_l^{-1}(z^l + z^{-l}). \tag{1.120}$$

The transfer functions $\widetilde{H}^1(z)$ and $H^1(z)$ have zero of multiplicity 3 as $z = 1$. Consequently, the wavelets $\widetilde{\psi}^1(t)$ and $\psi^1(t)$ have three vanishing moments. The transfer functions $\widetilde{H}^1(z)$ and $H^1(z)$ have zero of multiplicity 4 at $z = 1$. The wavelets $\widetilde{\psi}^{-1}(t)$ and $\psi^{-1}(t)$ have four vanishing moments. The synthesis scaling function $\varphi(t)$ is the quadratic fundamental spline and the synthesis wavelets are quadratic splines. The analysis waveforms are continuous (C^0) but do not have continuous derivative.

We display in Figure 1.25 the frequency responses of the analysis and synthesis filter banks, which originate from the quadratic continuous spline and corresponding waveforms. This figure is organized similarly to Figure 1.24.

Cubic spline: from (1.39), we derive the analysis filter bank

$$\widetilde{H}^1(z) = \frac{(z^{-1}-2+z)^2(z-z^{-1})(z^3+4z^2-16z-32-16z^{-1}+4z^{-2}+z^{-3})}{54(z^3+4+z^{-3})},$$

$$\widetilde{H}^0(z) = 1-\frac{(z^{-1}-2+z)^2}{2187(z^3+4+z^{-3})^2}\sum_{l=0}^{7}\widetilde{c}_l^0(z^l+z^{-l}),$$

$$\widetilde{H}^{-1}(z) = -\frac{(z^{-1}-2+z)^2(z^4+4z^3-17z^2-20z+10-20z^{-1}-17z^{-2}+4z^{-3}+z^{-4})}{54(z^3+4+z^{-3})}. \tag{1.121}$$

The synthesis filter bank is

$$
\begin{aligned}
H^1(z) &= \frac{\left(z^{-1}-2+z\right)^2\left(z-z^{-1}\right)}{2187\left(z^3+4+z^{-3}\right)^2}\sum_{l=0}^{8} c_l^1\left(z^l+z^{-l}\right),\\
H^0(z) &= \frac{\left(z+4+z^{-1}\right)\left(z+1+z^{-1}\right)^4}{27\left(z^3+4+z^{-3}\right)}\\
&= 3+\frac{\left(z-z^{-1}\right)^4\left(z^3+12z^2-12z-56-12z^{-1}+12z^{-2}+z^{-3}\right)}{27\left(z^3+4+z^{-3}\right)},\\
H^{-1}(z) &= -\frac{\left(z^{-1}-2+z\right)^2}{2187\left(z^3+4+z^{-3}\right)^2}\sum_{l=0}^{9} c_l^{-1}\left(z^l+z^{-l}\right).
\end{aligned}
\tag{1.122}
$$

The transfer functions $\widetilde{H}^1(z)$ and $H^1(z)$ have zero of multiplicity 5 at $z=1$. Consequently, the wavelets $\widetilde{\psi}^1(t)$ and $\psi^1(t)$ have five vanishing moments. The transfer functions $\widetilde{H}^1(z)$ and $H^1(z)$ have zero of multiplicity 4 at $z=1$. The wavelets $\widetilde{\psi}^{-1}(t)$ and $\psi^{-1}(t)$ have four vanishing moments. The synthesis scaling function $\varphi(t)$ is the cubic fundamental spline and the synthesis wavelets are cubic splines. The analysis waveforms are continuous (C^0) but do not have continuous derivative.

We display in Figure 1.26 frequency responses of the analysis and synthesis filter banks, which originate from the cubic continuous spline and the corresponding waveforms. This figure is organized similarly to Figure 1.24.

1.8.3.2. Discrete splines

Proposition 1.21. *If the prediction and update filters in the lifting scheme* $F_{\pm1}(1/z)=U^p_{\pm1,3}(z)=F_{\pm1}(z)$ *are derived from the continuous spline of order* p*, then*

(1) *the analysis filters* $\widetilde{H}^{\pm1}(z)$ *and synthesis filters* $H^{\pm1}(z)$ *have zero of multiplicity not less than* p *at* $z=1$*;*
(2) *the analysis wavelets* $\widetilde{\psi}^{\pm1}(t)$ *and synthesis wavelets* $\psi^{\pm1}(t)$ *have not less than* p *vanishing moments;*
(3) *the filter* $\widetilde{H}^0(z)=1+\widetilde{\chi}(z)$ *and the filter* $H^0(z)=3+\chi(z)$*, where the functions* $\widetilde{\chi}(z)$ *and* $\chi(z)$ *have zero of multiplicity not less than* p *at* $z=1$*;*
(4) *if* $p=2m+1$*, then* $\widetilde{H}^{-1}(z)$, $H^{-1}(z)$, $\widetilde{\chi}(z)$, *and* $\chi(z)$ *have zero of multiplicity* $p+1$ *at* $z=1$*;*
(5) *if* $p=2m+1$*, then* $\widetilde{\psi}^{-1}(t)$ *and* $\psi^{-1}(t)$ *have* $p+1$ *vanishing moments;*
(6) *if* $p=2m$*, then* $\widetilde{H}^1(z)$ *and* $H^1(z)$ *have zero of multiplicity* $p+1$ *at* $z=1$*;*
(7) *if* $p=2m$*, then the wavelets* $\widetilde{\psi}^1(t)$ *and* $\psi^1(t)$ *have* $p+1$ *vanishing moments.*

Quadratic discrete spline: from (1.56), we derive the analysis filter bank

$$\widetilde{H}^1(z) = \frac{(z^{-1}-2+z)(z-z^{-1})(z^2-z-3-z^{-1}+z^{-2})}{6(z^3+7+z^{-3})},$$

$$\widetilde{H}^0(z) = 1+\frac{3(z^{-1}-2+z)^2(z^3+6z^2+6z+1+6z^{-2}+6z^{-2}+z^{-3})}{(z^3+7+z^{-3})^2}\sum_{l=0}^{5}\widetilde{c}_l^0(z^l+z^{-l}),$$

$$\widetilde{H}^{-1}(z) = \frac{(z^{-1}-2+z)^2(z^2+z-1+z^{-1}+z^{-2})}{(z^3+7+z^{-3})}. \tag{1.123}$$

The synthesis filter bank is

$$H^1(z) = -\frac{(z^{-1}-2+z)(z-z^{-1})}{(z^3+7+z^{-3})^2}\sum_{l=0}^{5}c_l^1(z^l+z^{-l}),$$

$$H^0(z) = \frac{(z+1+z^{-1})^3}{z^3+7+z^{-3}} = 3+\frac{(z^{-1}-2+z)^2(2z+5z+z^{-1}+z^{-2})}{(z^3+7+z^{-3})}, \tag{1.124}$$

$$H^{-1}(z) = \frac{(z^{-1}-2+z)^2}{(z^3+7+z^{-3})^2}\sum_{l=0}^{5}c_l^{-1}(z^l+z^{-l}).$$

The transfer functions $\widetilde{H}^1(z)$ and $H^1(z)$ have zero of multiplicity 3 at $z=1$. Consequently, the wavelets $\widetilde{\psi}^1(t)$ and $\psi^1(t)$ have three vanishing moments. The transfer functions $\widetilde{H}^1(z)$ and $H^1(z)$ have zero of multiplicity 4 at $z=1$. The wavelets $\widetilde{\psi}^{-1}(t)$ and $\psi^{-1}(t)$ have four vanishing moments. The synthesis waveforms are continuous and have continuous derivative (C^1). The analysis waveforms are continuous (C^0) but do not have continuous derivative.

We display in Figure 1.27 frequency responses of the analysis and synthesis filter banks, which originate from the quadratic discrete spline and the corresponding waveforms. This figure is organized similarly to Figure 1.24.

Cubic discrete spline: from (1.57), we derive the analysis filter bank

$$\widetilde{H}^1(z) = \frac{(z^{-1}-2+z)^2(z-z^{-1})(4z^3+7+4z^{-3})}{4z^3+19+4z^{-3}},$$

$$\widetilde{H}^0(z) = 1-\frac{(z^{-1}-2+z)^2(4(z^5+z^{-5})-4(z^4+z^{-4})+123(z^3+z^{-3})+120(z+z^{-1}))}{3(4z^3+19+4z^{-3})^2},$$

$$\widetilde{H}^{-1}(z) = \frac{(z^{-1}-2+z)^2(4z^2+5z+5z^{-1}+4z^{-2})}{4z^3+19+4z^{-3}}. \tag{1.125}$$

The synthesis filter bank is

$$H^1(z) = -3\frac{\left(z^{-1}-2+z\right)^2\left(z-z^{-1}\right)}{\left(4z^3+19+4z^{-3}\right)^2}\sum_{l=0}^{4} c_l^1\left(z^l+z^{-l}\right),$$

$$H^0(z) = \frac{\left(z+1+z^{-1}\right)^4}{4z^3+19+4z^{-3}} = 3+\frac{\left(z-z^{-1}\right)^4\left(z^2-4z-12-4z^{-1}+z^{-2}\right)}{4z^3+19+4z^{-3}},$$

$$H^{-1}(z) = \frac{\left(z^{-1}-2+z\right)^2}{\left(4z^3+19+4z^{-3}\right)^2}\sum_{l=0}^{5} c_l^{-1}\left(z^l+z^{-l}\right). \tag{1.126}$$

The transfer functions $\widetilde{H}^1(z)$ and $H^1(z)$ have zero of multiplicity 5 at $z = 1$. Consequently, the wavelets $\widetilde{\psi}^1(t)$ and $\psi^1(t)$ have five vanishing moments. The transfer functions $\widetilde{H}^1(z)$ and $H^1(z)$ have zero of multiplicity 4 at $z = 1$.The wavelets $\widetilde{\psi}^{-1}(t)$ and $\psi^{-1}(t)$ have four vanishing moments. The synthesis waveforms are continuous and have two continuous derivatives (C^2). The analysis waveforms are continuous (C^0) but do not have continuous derivative.

We display in Figure 1.28 frequency responses of the analysis and synthesis filter banks, which originate from the cubic discrete spline and the corresponding waveforms. This figure is organized similarly to Figure 1.24.

Remarks 1.22. (1) The waveforms and the shape of the frequency responses of filters resulting from the discrete splines are very similar to their counterparts that stem from the continuous splines, although the structure of the filters resulting from the discrete splines is simpler.

(2) Unlike dyadic wavelet transforms, one step of the presented transform splits the frequency domain into three subbands. Three waveforms participate in the expansion of a signal. This promises better adaptivity of the expansion to the properties of the signal.

(3) A useful property of the transforms derived from the continuous splines is that the signal waveforms are splines.

(4) Currently, we investigate possible application of the presented transforms to compression and denoising.

Appendix

Implementation of recursive filters

Let $\mathbf{x} = \{x(k)\}$, $k = 1,\ldots,N$. To implement correctly recursive filtering of this finite-length signal, we have to extend $\mathbf{x}$ beyond the given interval. Since our filter banks are symmetric, we use the **HH** extension as in the terminology of [11]. It means that the signal is extended symmetrically with repetition of boundary

samples through both ends of the interval. Namely, $x(0) \triangleq x(1), x(-1) \triangleq x(2), \ldots,$ $x(-k) \triangleq x(k+1)$ and $x(N+1) \triangleq x(N), x(N+2) \triangleq x(N-1), \ldots, x(N+k) \triangleq$ $x(N-k+1)$. This results in periodization of the signal with period $2N$. This extended signal is denoted by $\tilde{\mathbf{x}}$.

Many recursive filters presented in the paper comprise a block of type

$$F(z) = \frac{1+z}{(1+\alpha z)(1+\alpha z^{-1})}. \tag{A.1}$$

We describe the application of the filter F, whose transfer function is given by $F(z)$, to a finite-length signal $\mathbf{x}$:

$$y(z) = F(z)x(z). \tag{A.2}$$

Equation (A.1) is equivalent to

$$F(z) = \frac{1}{1+\alpha}\left(\frac{1}{1+\alpha z^{-1}} + \frac{z}{1+\alpha z}\right). \tag{A.3}$$

Denote

$$\begin{aligned} y_1(z) &= \frac{1}{1+\alpha z^{-1}}x(z) = \sum_{n=0}^{\infty}(-\alpha)^n z^{-n}\tilde{x}(z), \\ y_2(z) &= \frac{z}{1+\alpha z}x(z) = \sum_{n=0}^{\infty}(-\alpha)^n z^{n+1}\tilde{x}(z). \end{aligned} \tag{A.4}$$

Then

$$y_1(k) = x(k) - \alpha y_1(k-1), \qquad y_2(k) = x(k+1) - \alpha y_2(k+1), \tag{A.5}$$

$$\Longleftrightarrow y_1(k) = \tilde{x}(k) + \sum_{n=1}^{\infty}(-\alpha)^n \tilde{x}(k-n), \qquad y_2(k) = \tilde{x}(k+1) + \sum_{n=1}^{\infty}(-\alpha)^n \tilde{x}(k+n+1). \tag{A.6}$$

We can use (A.5) for the computation of $y_1(k)$ and $y_2(k)$ provided that we know $y_1(1)$ and $y_2(N)$, respectively. To evaluate these samples, we employed (A.6) keeping in mind the extension of the input signal $\mathbf{x}$. We have

$$y_1(1) = x(1) + \sum_{n=1}^{\infty}(-\alpha)^n \tilde{x}(-n+1) \approx x(1) + \sum_{n=1}^{d}(-\alpha)^n x(n), \tag{A.7}$$

$$y_2(N) = \tilde{x}(N+1) + \sum_{n=1}^{\infty}(-\alpha)^n \tilde{x}(N+n+1) \approx x(N) + \sum_{n=1}^{d}(-\alpha)^n x(N-n), \tag{A.8}$$

where d is the prescribed initialization depth.

Filtering (A.2) can be implemented by the following parallel algorithm:

(1) evaluate $y_1(1)$ from (A.7) and $y_2(N)$ from (A.8);
(2) calculate $y_1(k) = x(k) - \alpha y_1(k-1)$, $k = 2, \ldots, N$, and $y_2(k) = x(k+1) - \alpha y_2(k+1)$, $k = N-1, \ldots, 1$;
(3) $y(k) = (y_1(k) + y_2(k))/(1+\alpha)$, $k = 1, \ldots, N$.

Equations (A.6) and (A.8) imply that $y_2(N) = y_1(N)$. Hence, it follows that

$$y(N) = \frac{y_1(N) + y_2(N)}{1+\alpha} = \frac{2y_1(N)}{1+\alpha}. \tag{A.9}$$

The cascade algorithm has the following form:

(1) evaluate $y_1(1)$ from (A.7);
(2) calculate $y_1(k) = x(k) - \alpha y_1(k-1)$, $k = 2, \ldots, N$;
(3) evaluate $y(N)$ from (A.9);
(4) calculate $y(k) = y_1(k) + y_1(k+1) - \alpha y(k+1)$, $k = N-1, \ldots, 1$.

Bibliography

[1] O. Amrani, A. Z. Averbuch, T. Cohen, and V. A. Zheludev, "Symmetric interpolatory framelets and their error correction properties," to appear in *International Journal of Wavelets, Multiresolution and Information Processing*.

[2] A. Z. Averbuch and V. A. Zheludev, "Construction of biorthogonal discrete wavelet transforms using interpolatory splines," *Applied and Computational Harmonic Analysis*, vol. 12, no. 1, pp. 25–56, 2002.

[3] A. Z. Averbuch, A. B. Pevnyi, and V. A. Zheludev, "Biorthogonal butterworth wavelets derived from discrete interpolatory splines," *IEEE Transactions on Signal Processing*, vol. 49, no. 11, pp. 2682–2692, 2001.

[4] A. Z. Averbuch, A. B. Pevnyi, and V. A. Zheludev, "Butterworth wavelet transforms derived from discrete interpolatory splines: recursive implementation," *Signal Processing*, vol. 81, no. 11, pp. 2363–2382, 2001.

[5] A. Z. Averbuch and V. A. Zheludev, "Wavelets transforms generated by splines," *International Journal of Wavelets, Multiresolution, and Information Processing*, vol. 5, no. 2, pp. 257–292, 2007.

[6] A. Z. Averbuch and V. A. Zheludev, "Lifting scheme for biorthogonal multiwavelets originated from hermite splines," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 487–500, 2002.

[7] A. Z. Averbuch and V. A. Zheludev, "A new family of spline-based biorthogonal wavelet transforms and their application to image compression," *IEEE Transactions on Image Processing*, vol. 13, no. 7, pp. 993–1007, 2004.

[8] A. Z. Averbuch, V. A. Zheludev, and T. Cohen, "Multiwavelet frames in signal space originated from Hermite splines," *IEEE Transactions on Signal Processing*, vol. 55, no. 3, pp. 797–808, 2007.

[9] A. Z. Averbuch, V. A. Zheludev, and T. Cohen, "Interpolatory frames in signal space," *IEEE Transactions on Signal Processing*, vol. 54, no. 6, part 1, pp. 2126–2139, 2006.

[10] A. Z. Averbuch, V. A. Zheludev, and T. Cohen, "Tight and sibling frames originated from discrete splines," *Signal Processing*, vol. 86, no. 7, pp. 1632–1647, 2006.

[11] C. M. Brislawn, "Classification of nonexpansive symmetric extension transforms for multirate filter banks," *Applied and Computational Harmonic Analysis*, vol. 3, no. 4, pp. 337–357, 1996.

[12] R. H. Chan, S. D. Riemenschneider, L. Shen, and Z. Shen, "High-resolution image reconstruction with displacement errors: a framelet approach," *International Journal of Imaging Systems and Technology*, vol. 14, no. 3, pp. 91–104, 2004.

[13] R. H. Chan, S. D. Riemenschneider, L. Shen, and Z. Shen, "Tight frame: an efficient way for high-resolution image reconstruction," *Applied and Computational Harmonic Analysis*, vol. 17, no. 1, pp. 91–115, 2004.

[14] C. K. Chui and W. He, "Compactly supported tight frames associated with refinable functions," *Applied and Computational Harmonic Analysis*, vol. 8, no. 3, pp. 293–319, 2000.
[15] A. Cohen, I. Daubechies, and J. C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 45, no. 5, pp. 485–560, 1992.
[16] Z. Cvetković and M. Vetterli, "Oversampled filter banks," *IEEE Transactions on Signal Processing*, vol. 46, no. 5, pp. 1245–1255, 1998.
[17] D. Marpe, G. Heising, H. L. Cycon, and A. P. Petukhov, "Video coding using a bilinear image warping motion model and wavelet-based residual coding," in *Wavelet Applications in Signal and Image Processing VII*, vol. 3813 of *Proceedings of SPIE*, pp. 401–408, Denver, Colo, USA, 1999.
[18] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 41, no. 7, pp. 909–996, 1988.
[19] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, Pa, USA, 1992.
[20] I. Daubechies, B. Han, A. Ron, and Z. Shen, "Framelets: MRA-based constructions of wavelet frames," *Applied and Computational Harmonic Analysis*, vol. 14, no. 1, pp. 1–46, 2003.
[21] F. Labeau, J.-C. Chiang, M. Kieffer, P. Duhamel, L. Vandendorpe, and B. Macq, "Oversampled filter banks as error correcting codes: theory and impulse noise correction," *IEEE Transactions on Signal Processing*, vol. 53, no. 12, pp. 4619–4630, 2005.
[22] R. W. Gerchberg, "Super-resolution through error energy reduction," *Optica Acta*, vol. 21, no. 9, pp. 709–720, 1974.
[23] V. K. Goyal, J. Kovacevic, and J. A. Kelner, "Quantized frame expansions with erasures," *Applied and Computational Harmonic Analysis*, vol. 10, no. 3, pp. 203–233, 2001.
[24] V. K. Goyal, M. Vetterli, and N. T. Thao, "Quantized overcomplete expansions in $\mathbb{R}^n$: analysis, synthesis, and algorithms," *IEEE Transactions on Information Theory*, vol. 44, no. 1, pp. 16–31, 1998.
[25] G. Rath and C. Guillemot, "Characterization of a class of error correcting frames for robust signal transmission over wireless communication channels," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 2, pp. 229–241, 2005.
[26] H. Bölcskei, F. Hlawatsch, and H. G. Feichtinger, "Frame-theoretic analysis of oversampled filter banks," *IEEE Transactions on Signal Processing*, vol. 46, no. 12, pp. 3256–3268, 1998.
[27] C. Herley and M. Vetterli, "Wavelets and recursive filter banks," *IEEE Transactions on Signal Processing*, vol. 41, no. 8, pp. 2536–2556, 1993.
[28] Q. Jiang, "Parameterizations of masks for tight affine frames with two symmetric/antisymmetric generators," *Advances in Computational Mathematics*, vol. 18, no. 2–4, pp. 247–268, 2003.
[29] J. Kovacević, P. L. Dragotti, and V. K. Goyal, "Filter bank frame expansions with erasures," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1439–1450, 2002.
[30] C. Lan, K. R. Narayanan, and Z. Xiong, "Scalable image transmission using rate-compatible irregular repeat accumulate (IRA) codes," in *Proceedings of IEEE International Conference on Image Processing (ICIP '02)*, vol. 3, pp. 717–720, Rochester, NY, USA, June 2002.
[31] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions of Image Processing*, vol. 1, no. 2, pp. 205–220, 1992.
[32] S. Marinkovic and C. Guillemot, "Joint source-channel coding based on cosine-modulated filter banks for erasure-resilient signal transmission," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 4, pp. 510–524, 2005.
[33] R. Motwani and C. Guillemot, "Tree-structured oversampled filterbanks as joint source-channel codes: application to image transmission over erasure channels," *IEEE Transactions on Signal Processing*, vol. 52, no. 9, pp. 2584–2599, 2004.
[34] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1989.
[35] A. Papoulis, "A new algorithm in spectral analysis and band-limited extrapolation," *IEEE Transactions on Circuits and Systems*, vol. 22, no. 9, pp. 735–742, 1975.
[36] A. P. Petukhov, "Biorthogonal wavelet bases with rational masks and their applications," *Trudy Sankt-Peterburgskogo Matematicheskogo Obshchestva*, vol. 7, pp. 168–193, 1999.

[37] A. P. Petukhov, "Wavelet frames and their applications to wireless transmission," in *Proceedings of Communication at the 5th AFA Conference on Curves and Surfaces*, Saint Malo, France, June-July 2002.
[38] A. P. Petukhov, "Symmetric framelets," *Constructive Approximation*, vol. 19, no. 1, pp. 309–328, 2003.
[39] A. Ron and Z. Shen, "Affine Systems in $l^2(\mathbb{R}^d)$ ii: dual systems," *Journal of Fourier Analysis and Applications*, vol. 3, no. 5, pp. 617–637, 1997.
[40] A. Ron and Z. Shen, "Compactly supported tight affine spline frames in $l^2(\mathbb{R}^d)$," *Mathematics of Computation*, vol. 67, no. 221, pp. 191–207, 1998.
[41] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, 1996.
[42] I. W. Selesnick and A. F. Abdelnour, "Symmetric wavelet tight frames with two generators," *Applied and Computational Harmonic Analysis*, vol. 17, no. 2, pp. 211–225, 2004.
[43] G. Strang and T. Nguen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Ellesley, Mass, USA, 1996.
[44] W. Sweldens, "The lifting scheme: a custom-design construction of biorthogonal wavelets," *Applied and Computational Harmonic Analysis*, vol. 3, no. 2, pp. 186–200, 1996.

Amir Z. Averbuch: The School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
Email: amir1@post.tau.ac.il

Valery A. Zheludev: The School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
Email: zhel@post.tau.ac.il

2 Recent developments in Haar wavelet transform for application to switching and multivalued logic functions representations

Radomir S. Stanković, Karen Egiazarian, and Jaakko Astola

In many instances, classical approaches in switching theory and logic design require brute force search procedures. Application of spectral transforms often provides a promising solution of the problem. In particular, the discrete Haar transform appears very interesting for such applications for its wavelet-like properties and fast calculation algorithms. The chapter briefly reviews recent development in this area and is intended to provide compact representations of discrete functions including switching and multivalued logic functions as examples. To this end, decision diagrams are treated in terms of the Haar transform and an algorithm for reduction of the number of paths and the size of such diagrams is described with potential applications to the problems of circuit verification and testing. These representations of multi-output switching functions can be a basis for efficient realizations and various other applications. For instance, word-level representations, including Haar series representations of switching functions and related decision diagrams, are useful in fast prototyping by LUT-FPGAs and in hardware-software codesign. Then, an extension of definitions of Haar spectral transform diagrams to multivalued functions is provided. It is shown that these decision diagrams determine a large class of multivalued Haar functions and related transforms.

2.1. Introduction: logic design and spectral transforms

Switching theory is a branch of applied mathematics that provides mathematical foundations for logic design, which can be viewed as an essential part of digital system design concerning realizations of systems whose input and output signals are described by logic functions. Thus, switching theory may be considered as a part of a more general system theory and it is closely related to signal processing.

In this context, switching theory being based on binary-valued logic deals with two-valued (switching) functions that are a basis for the realization of systems consisting of components with two stable states, which are nowadays highly

prevalent in practice. Multivalued (MV) logic functions may be considered as a generalization of two-valued switching functions enabling more efficient encoding of larger amounts of information than it is possible in the binary case. Thus, multivalued logic is often regarded as a promising alternative enabling competing with ever increasing amounts of information that should be stored, transmitted, and processed in every day practice.

The vast complexity of modern digital systems implies that they can only be handled by computer-aided design tools that are built on sophisticated mathematical models. Such models usually assume embedding sets representing domains and ranges for functions describing signals that should be processed into some algebraic structures. These structures should be selected such that they reflect at least some of the true properties of signals. Selecting groups for the domains, and fields for the ranges, leads to linear vector spaces, which are algebraic structures often used for mathematical modeling of signals. Spectral transforms over these structures have been proven to be very efficient tools in signal processing, system theory, and related areas. In particular, spectral transforms on finite dyadic groups, as the discrete Walsh and Haar transforms, are able to capture properties of binary-valued logic functions. That feature, together with their other useful properties, as fast computation methods, makes these spectral transforms useful also from the practical applications' point of view. When properly generalized, the same properties can be extended to spectral transforms for MV functions.

With that motivation, this chapter discusses applications of the discrete Haar transform and its generalizations to derivation of compact representations of binary and multivalued functions for potential applications in logic design and, in general, digital system design.

2.2. Discrete Haar functions

The discrete Haar transform is among the first discrete spectral transforms that have found applications in several areas of electrical and computer engineering, including switching theory and logic design [1, 2]. This transform is defined in terms of the discrete Haar functions that can be introduced and studied from different points of view.

(1) First, the discrete Haar functions and related transform can be viewed as the discrete counterpart of the Haar functions and related series defined in the Hilbert space $L_p[0,1]$ for $p \in [1,\infty]$ which take values from the set $\{0, \sqrt{2^i}\}$, i a positive integer [3]. The system introduced by the Hungarian mathematician Alfred Haar [3] has the property that every function continuous on $[0,1]$ can be represented by an infinite series in terms of the Haar functions.

In some applications, especially these related intensive computations, the unnormalized Haar functions taking values in $\{0, \pm 1\}$ are often more convenient.

(2) The discrete Haar functions, both normalized and unnormalized, can be derived by sampling the corresponding Haar functions at 2^n equidistant points in $[0,1]$.

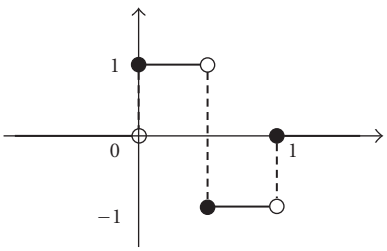


FIGURE 2.1. Haar wavelet.

(3) Alternatively, some authors prefer to consider the discrete Haar functions as a set of functions introduced independently of the Haar functions in L_p, and as a complete orthogonal system in the discrete Hilbert space l_2 of real and complex-valued functions defined in 2^n points. These functions take values from the same set as the Haar functions and the distribution of the values resembles the waveforms of Haar functions. In this case, the discrete Haar functions are defined in terms of matrices expressing particular Kronecker layered structure; see, for instance [4]. Efforts have been made to study these matrices and their relationships to the matrices defining other discrete transforms [5–8].

(4) The discrete Haar functions are a special example of the so-called bridge functions defined in [9] and elaborated further in a series of publications; see, for instance [10] and references therein.

(5) In a system theory approach, the discrete Haar functions are viewed as eigenfunctions of discrete systems modeled by the discrete differential equations in terms of the corresponding derivatives in the class of Gibbs derivatives [11, 12]. These functions are a particular example of the generalized Haar functions introduced in [13]. The discrete Haar functions can also be defined as solutions of differential equations in terms of the generalized Haar derivatives [14]. In a more general approach, viewed as a particular complete orthogonal system on finite Abelian groups, the discrete Haar functions can be derived as solutions of differential equations in terms of the corresponding class of Gibbs derivatives on finite Abelian groups [15]. A generalization of these results is presented in [16].

(6) The set of Haar functions can be split into packets, and functions within the same packet can be derived by shifting along the interval $[0, 1]$. Functions in a packet can be derived from functions in the preceding packet by scaling on the interval where these functions are different from zero followed by the multiplication with the power of $\sqrt{2}$.

Due to these properties, the Haar functions introduced in 1910 are related to the more recently developed wavelet theory, where the Haar wavelet (function) is viewed as the canonical example of an orthonormal wavelet, that is a wavelet providing a complete set of basic elements for $L_2(R)$, and being at the same time the simplest possible wavelet. Figure 2.1 shows the basic Haar wavelet. The disadvantage of the Haar wavelet is that it is not continuous and therefore not differentiable in the classical sense.

There is apparent a renewed interest in application of the discrete Haar transform and various generalizations; see, for instance [17] and references therein. For instance, a special session devoted exclusively to Haar transforms has been organized within the *International Conference on Informatics, Communications and Signal Processing (1st ICICS)*, 1997, in Singapore. Besides signal and image processing, see, for instance [18–24] and references in these publications, interesting applications have been found in switching theory and logic design for circuit synthesis, verification, and testing [5, 7, 25–32].

In this area, relating the discrete Haar transform with theory and practice of decision diagrams for representation of discrete signals proved to be especially interesting since it permits efficient, in terms of space and time, calculation of Haar spectra for functions defined in 2^n points for a large n, for instance, $n > 50$–100 or even more, depending on the available resources.

In the first part of this chapter, we briefly discuss the Haar spectral diagrams (HSDs). Then, we define the Haar transform decision diagrams (HSTDDs). We show an algorithm for minimization of HSTDDs in terms of the number of nodes and paths. This research was motivated by the following considerations.

In the second part of the chapter, we discuss extensions of the definition of Haar functions to multivalued case through the corresponding decision diagrams.

2.3. Decision diagrams and their optimization

Decision diagrams (DDs) are an efficient data structure for representation of discrete functions [34]. Formally, a decision diagram can be viewed as an acyclic directed graph consisting of a set of nonterminal nodes and constant nodes connected by edges. A decision diagram is characterized by several parameters.

(1) The number of nonterminal and constant nodes, that is, the *size* of the diagram.
(2) The number of levels at which nonterminal nodes are distributed, denoted as the *depth* of the diagram.
(3) The maximum number of nodes at a level, called the *width* of the diagram.
(4) The number of paths from the root node to the constant nodes, expressing, in a way, the complexity of interconnections in the diagram. Depending on the values shown by constant nodes, the 0-paths and c-paths, where c could be a complex number, an integer, or any other value different from zero, are distinguished.
(5) The number of outgoing edges per node, called the *cost* of a node.

These parameters are strongly mutually related and determine the *structure* of the decision diagram [33]. Depending on applications as well as particular classes of decision diagrams, some other parameters of decision diagrams, such as attributes at the edges, average paths length, and so forth, are also considered [34–36].

There is a variety of decision diagrams for discrete functions, which differ in their basic characteristics [37].

In most of applications of decision diagrams, the objective is to reduce some of these characteristics of the decision diagram for a given function f, which should ensure the efficiency of methods using decision diagrams. However, exact minimization algorithms exist only for some of these basic characteristics of decision diagrams. If the exact algorithms exist, then they perform an exhaustive search in the research space of all possible combinations of different parameters permitting reduction of a characteristic. Derivation of exact and deterministic algorithms for the optimization of decision diagrams is restricted by the large, even though, dimension of search space for possible solutions. Therefore, a study of exact algorithms for optimization of decision diagram representations is an interesting and important topic.

In this chapter, we are particularly interested in reduction of the number of paths of decision diagrams, which is a problem important in verification and testing of digital circuits, efficient evaluation of large discrete functions, and other areas where frequent traversing of a decision diagram may be required.

In [38], an exact algorithm is presented which for a given function f determines the Haar spectrum with the minimum number of nonzero coefficients. The algorithm performs reordering of elements in the vector of function values for f. In this respect, the algorithm relates to the algorithms for optimization of decision diagrams by variables ordering, input groupings, and outputs pairing [34, 39].

The algorithm by Karpovsky [38] can be used to minimize the number of paths in the HSTDD for a given function f. This research is motivated by the recent renewed interest in applications of the Haar transform in logic design and related areas [5, 27, 29, 31]. We believe that HSTDDs can improve applicability of earlier and recent spectral methods using the Haar transform and can be useful in its various generalizations [13, 14, 16, 29, 40].

2.4. Haar expressions for discrete functions

The discrete Haar series can be viewed as a discrete counterpart of the Haar series for functions on the interval $[0, 1)$ [3]. They can be considered as functional expressions defined on finite dyadic groups due to the isomorphism between $[0, 1)$ and the infinite dyadic group.

Consider the space $C(C_2^n)$ of complex functions f defined in 2^n points. The integer-valued functions and switching functions are considered as subsets of the complex functions with ranges of function values restricted to some particular subsets of complex numbers. Multiple-output switching functions are represented by integer functions whose values are determined by considering binary outputs as coordinates in binary representations of integers. These functions can be represented by the discrete Haar series, that is, series defined in terms of the discrete Haar functions.

Due to intended applications in representation of switching and multivalued logic functions, in this chapter, we will use the unnormalized Haar functions; see, for instance, [4, 36].

Definition 2.1 (Haar matrix). Denote by $\mathrm{har}(w,x)$, $w,x \in \{0,\ldots,2^n-1\}$, the Haar function of the index w. In matrix notation, the discrete Haar functions can be defined as rows of the $(2^n \times 2^n)$ Haar matrix defined as

$$\mathbf{H}(n) = \begin{bmatrix} \mathbf{H}(n-1) \otimes [1 \quad 1] \\ \mathbf{I}(n-1) \otimes [1 \quad -1] \end{bmatrix} \tag{2.1}$$

with $H(0) = [1]$, and where $\mathbf{I}(n-1)$ is the $(2^{n-1} \times 2^{n-1})$ identity matrix.

Example 2.2. For $n = 3$, the unnormalized discrete Haar functions in the sequency or Haar ordering is defined by the matrix

$$\mathbf{H}(3) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}. \tag{2.2}$$

Definition 2.3 (inverse Haar matrix). The inverse unnormalized Haar transform is defined as

$$\mathbf{H}^{-1}(n) = \frac{1}{2}\left[\mathbf{H}^{-1}(n-1) \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{I}(n-1) \otimes \begin{bmatrix} 2^{n-1} \\ -2^{n-1} \end{bmatrix}\right]. \tag{2.3}$$

Example 2.4. For $n = 3$, the inverse unnormalized Haar matrix is

$$\mathbf{H}^{-1}(3) = \frac{1}{8}\begin{bmatrix} 1 & 1 & 2 & 0 & 4 & 0 & 0 & 0 \\ 1 & 1 & 2 & 0 & -4 & 0 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & 4 & 0 & 0 \\ 1 & 1 & -2 & 0 & 0 & -4 & 0 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & 4 & 0 \\ 1 & -1 & 0 & 2 & 0 & 0 & -4 & 0 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & 4 \\ 1 & -1 & 0 & -2 & 0 & 0 & 0 & -4 \end{bmatrix}. \tag{2.4}$$

Definition 2.5 (Haar transform). For f represented by the vector of function values $\mathbf{F}(n) = [f(0), \ldots, f(2^n - 1)]^T$, the Haar spectrum $\mathbf{S}_f(n) = [S_f(0), \ldots, S_f(2^n - 1)]^T$ is given by

$$\begin{aligned} \mathbf{S}_f(n) &= 2^{-n}\mathbf{H}(n)\mathbf{F}(n), \\ \mathbf{F}(n) &= \mathbf{H}(n)^{-1}\mathbf{S}_f(n). \end{aligned} \tag{2.5}$$

Definition 2.6 (Haar coefficients). The Haar spectral coefficients for a function f defined at 2^n points are calculated as

$$S_f(w) = 2^{-n} \sum_{x=0}^{2^n-1} f(x)\,\mathrm{har}(w,x), \tag{2.6}$$

where $\mathrm{har}(w,x)$ are the Haar functions defined as rows of the Haar matrix $\mathbf{H}(n)$.

Definition 2.7 (Haar expression). The Haar expression for f is defined as

$$f(x) = \sum_{w=0}^{2^n-1} S_f(w)\,\mathrm{har}^{-1}(w,x), \tag{2.7}$$

where S_f are Haar coefficients, and $\mathrm{har}^{-1}(w,x)$ are defined by rows of the inverse Haar matrix $\mathbf{H}^{-1}(n)$.

2.5. Haar spectral diagrams

Haar spectral diagrams (HSDs) are introduced in [27] as an edge-valued modification of multiterminal binary decision diagrams (MTBDDs) [41] to represent the Haar spectrum for a given function f.

Definition of HSDs resembles the edge-valued binary decision diagrams (EVBDDs) [42] and their relationship to the arithmetic transform [43, 44], however, there are important differences when considered functional expressions related to the decision diagrams. Due to the mentioned relationship with arithmetic transform coefficients, an EVBDD represents f in the form of an arithmetic expression for f. That expression is determined in terms of the arithmetic coefficients assigned to the incoming edge of the root node and the right edges of other nonterminal nodes in the EVBDD for f. The arithmetic transform expansion rule [45] has been adapted to the characteristics of EVBDDs, and used as the expansion rule to assign f to an EVBDD [42]. The inverse rule is used to determine f from a given EVBDD in the form of an arithmetic expression for f.

In HSDs, the Haar coefficients are assigned to the incoming edge of the root node and the right outgoing edges in the MTBDDs for f. The assignment of the Haar coefficients to the MTBDDs for f is performed through a correspondence among the Haar coefficients and Haar functions in terms of which they are determined. A coefficient is situated at the edge ending the subpath consisting of edges

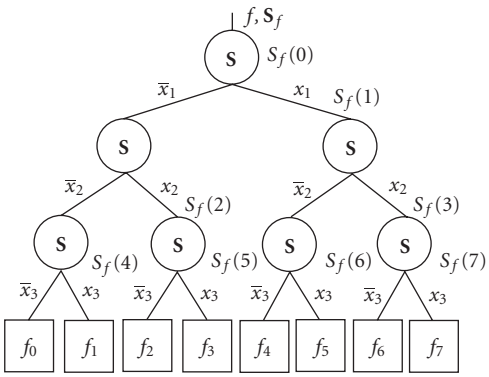


FIGURE 2.2. HST for $n = 3$.

whose labels are used to describe the corresponding Haar function in terms of binary variables.

However, no expansion rule is derived to determine the meaning of nodes. In HSDs, the nodes are MTBDD nodes. Therefore, from HSD for f, we can read f as from the MTBDD(f), and we can read the Haar coefficients as from the MTBDD(S_f), however, we cannot read f in terms of the Haar coefficients. In [27], it is shown that if f is a switching function in coding $(1, -1)$, the values of constant nodes in the MTBDD for f can be determined by the values of spectral coefficients assigned to the edges of nodes corresponding to x_n. This is possible, since from the definition of the Haar matrix, the processing of nodes at this level consists of subtraction of the function values at the neighboring points. However, that property cannot be used for integer or complex-valued functions.

Example 2.8. Figure 2.2 shows an example of the Haar spectral tree (HST) for $n = 3$. The following relation shows the correspondence among the Haar functions, labels at the edges in the multiterminal binary decision tree (MTBDT), and the Haar coefficients

$$\begin{aligned}
&f(0) \quad \mathrm{har}(0,x) = 1 && S_f(0),\\
&f(1) \quad \mathrm{har}(1,x) = (1-2x_1) && S_f(1),\\
&f(2) \quad \mathrm{har}(2,x) = (1-2x_2)\overline{x}_1 && S_f(2),\\
&f(3) \quad \mathrm{har}(3,x) = (1-2x_2)x_1 && S_f(3),\\
&f(4) \quad \mathrm{har}(4,x) = (1-2x_3)\overline{x}_1\overline{x}_2 && S_f(4),\\
&f(5) \quad \mathrm{har}(5,x) = (1-2x_3)\overline{x}_1 x_2 && S_f(5),\\
&f(6) \quad \mathrm{har}(6,x) = (1-2x_3)x_1\overline{x}_2 && S_f(6),\\
&f(7) \quad \mathrm{har}(7,x) = (1-2x_3)x_1 x_2 && S_f(7).
\end{aligned} \tag{2.8}$$

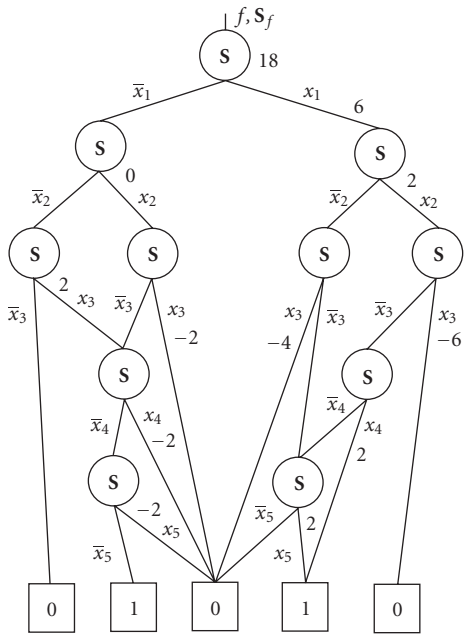


FIGURE 2.3. HSD for f in Example 2.10.

From this example, as stated above, a coefficient $S_f(w)$ is situated at the end of a subpath consisting of edges denoted by variables used in symbolic description of the corresponding Haar function $\text{har}(w, x)$ with respect to which this Haar coefficient is calculated.

Example 2.9. In HSD for $n = 3$, the coefficient $S_f(1)$ is situated at the edge labeled by x_1. The coefficient $S_f(2)$ is situated at the end of the subpath consisting of the edges labeled by $\overline{x}_1$ and x_2. In this way, $S_f(7)$ is situated at the path consisting of edges labeled by x_1, x_2, and x_3.

In the representation of the Haar functions in terms of switching variables, we keep $\overline{x}_i$ for logic NOT, since we would not like to change the notation which has been already used in discussions of HSD [31]. However, since we are working with integers or complex numbers, the proper meaning is $\overline{x}_i = (1 - x_i)$ with $x_i \in \{0, 1\}$, where 0 and 1 are the integers 0 and 1.

Example 2.10 (see [27]). Figure 2.3 shows the MTBDD and the HSD for the Haar spectrum for the function

$$f = x_1x_2\overline{x}_3x_4 \vee x_1\overline{x}_3x_5 \vee \overline{x}_1x_2\overline{x}_3\overline{x}_4\overline{x}_5 \vee \overline{x}_1\overline{x}_2x_3\overline{x}_4\overline{x}_5. \tag{2.9}$$

The truth-vector for f in the $(1,-1)$ encoding is

$$\begin{aligned}\mathbf{F} = [&1,1,1,1,-1,1,1,1,-1,1,1,1,1,1,1,1,\\ &1,-1,1,-1,1,1,1,1,1,-1,-1,-1,1,1,1,1]^T.\end{aligned} \tag{2.10}$$

The Haar spectrum for this function f in the Haar ordering is

$$\begin{aligned}\mathbf{S}_f = \frac{1}{32}[&18,6,0,2,2,-2,-4,-6,0,-2,-2,0,0,0,2,0,\\ &0,0,-2,0,-2,0,0,0,2,2,0,0,2,0,0,0]^T.\end{aligned} \tag{2.11}$$

Since in an HSD, the structure of the initial MTBDD for f is not changed and constant nodes are preserved, but ignored in the HSD, f remains represented by the same MTBDD from which the HSD is derived. This can be pointed out as an advantage of HSDs, since the complexity of the diagram to represent the Haar spectrum is equal to the complexity of the decision diagram representation of the initial function f whose spectrum is calculated. Recall that, in general, MTBDDs to represent integer-valued Haar spectra of binary-valued switching functions have a larger size than BDDs for the initial functions. Therefore, assigning the Haar coefficients to edges of BDDs for f saves space in representation of the Haar spectra.

It should be noticed that there are some classes of switching functions, that are important in applications, with the opposite property. That is, for some switching functions, Haar spectra are simple in the sense that can be represented by MTBDDs of the small size or the number of paths.

Remark 2.11. For a given f, the HSD for the Haar spectrum S_f is the MTBDD for f (MTBDD(f)) with the first Haar coefficient assigned to the incoming edge and other Haar coefficients to the right edges of other nonterminal nodes. That assignment corresponds to the assignment of the arithmetic transform coefficients to the edges in EVBDDs.

Alternatively, an HSD(f) is a modified MTBDD for the Haar spectrum (MTBDD(S_f)). The modification is done by moving Haar coefficients at the edges from the constant nodes. This modification permits to keep the structure of the MTBDD(f), instead of that of MTBDD(S_f), which, as noticed above, for switching functions is usually simpler than that of the MTBDD(S_f). However, the price is that we cannot read f in terms of the Haar coefficients, and that we cannot represent all the spectral coefficients explicitly. As it is explained in [27], the zero coefficients assigned to the edges at the lower levels are not explicitly shown. For example, in Figure 2.3, 12 of 32 coefficients are shown explicitly, and just a single

zero coefficient assigned to the right outgoing edge of the leftmost node corresponding to x_2. The other zero coefficients are not shown. It is assumed that they are assigned to the outgoing edges of the cross points pointing to constant nodes.

The assignment of Haar coefficients to the edges is determined from the following correspondence between the rows of the Haar matrix and the corresponding Haar coefficients for f. In determination of this correspondence, it is assumed that the discrete Haar functions are expressed in terms of switching variables, which are used as labels at the edges of MTBDDs. This is an alternative interpretation of the assignment of the Haar coefficients to the edges and it is different from that in [27]. However, the produced HSDs are the same, since the interpretation does not interfere with the definition of HSDs.

In [27], HDDs are defined with respect to the natural ordering of the Haar matrix. A decomposition of the Haar matrix into a sum of two Kronecker prod uct representable matrices is used. This decomposition was used in argumentation for the assignment of the Haar coefficients to the edges. Different ordering of the Haar matrix implies different enumeration of the Haar coefficients and different ordering of variables in related decision diagrams.

However, in any ordering, the coefficients defined with respect to the same Haar function are assigned to the same edges in the MTBDD(f).

2.5.1. HSDs and other decision diagrams

An MTBDD for a given f represents f through the disjunctive normal form for f. The product terms are determined as products of labels at the edges.

An HSD represents the Haar coefficients for f through the MTBDD for f. Thus, an HSD does not represent a given f in terms of a Haar expression for f. It follows that HSDs, although denoted as spectral diagrams, are not the spectral transform decision diagrams (STDDs) in the sense of the definition of that concept in [45].

The same as EVBDDs, HSDs are decision diagrams with attributed edges. In HSDs, the attributes are the Haar coefficients. In EVBDDs, they are the arithmetic transform coefficients [43]. However, an HSD does not use the Haar coefficients to represent f, in a way as EVBDDs represent f in terms of the arithmetic coefficients used as the attributes at the edges. It follows that HSDs are related to EVBDDs, but are not a counterpart of EVBDDs in the sense that the Haar coefficients instead the arithmetic coefficients are used in EVBDDs.

We have shown a formalism to assign Haar coefficients to the edges, expressed through the relations of rows of the transform matrices in terms of switching variables used as labels at the edges in MTBDDs. The formalism used to define HSDs may be extended to other spectral transforms where rows of the transform matrix are expressed in terms of switching variables. The following example illustrates this statement.

Example 2.12. For $n = 3$, the following correspondence between the Walsh functions wal(w, x), which are rows of the Walsh transform matrix, switching variables,

and Walsh coefficients may be shown:

$$\begin{aligned}
&f(0) \quad \mathrm{wal}(0,x) = 1 && S_f(0),\\
&f(1) \quad \mathrm{wal}(1,x) = (1-2x_3) && S_f(1),\\
&f(2) \quad \mathrm{wal}(2,x) = (1-2x_2) && S_f(2),\\
&f(3) \quad \mathrm{wal}(3,x) = (1-2x_2)(1-2x_3) && S_f(3),\\
&f(4) \quad \mathrm{wal}(4,x) = (1-2x_1) && S_f(4),\\
&f(5) \quad \mathrm{wal}(5,x) = (1-2x_1)(1-2x_3) && S_f(5),\\
&f(6) \quad \mathrm{wal}(6,x) = (1-2x_1)(1-2x_2) && S_f(6),\\
&f(7) \quad \mathrm{wal}(7,x) = (1-2x_1)(1-2x_2)(1-2x_3) && S_f(7).
\end{aligned} \tag{2.12}$$

Due to that, we may define a Walsh spectral tree (WST) for $n = 3$ corresponding to the HST in Figure 2.2, with the Walsh coefficients at the same positions where the Haar coefficients are in the HSD. However, since in this example we are using the Walsh transform in Kronecker ordering, the variables should be assigned to the levels in the descending order x_3, x_2, x_1.

2.5.2. HSDs and ordering of Haar functions

In applications, both normalized or unnormalized Haar matrices are used with rows ordered in different ways. Some of these orderings were intended to provide Kronecker or Kronecker-like representations of the Haar matrices suitable for their generation or calculations with Haar matrices [27, 38]. In [27], an ordering that allows a particular decomposition of the Haar matrix was used to assign Haar coefficients to the edges in MTBDD(f).

The ordering in Example 2.8 is denoted as the sequency ordering or Haar ordering. However, all the considerations may be extended to any other ordering. In the representations of Haar spectrum through decision diagrams, as MTBDDs and HSDs, the following may be stated.

Remark 2.13. Given an HSD showing Haar coefficients for a particular ordering of the Haar matrix. From this HSD, we can read the Haar coefficients for any other ordering by performing different traversal of the HSD, that is, by changing the way of visiting labels at the edges in the HSD.

We refer to [46] for more details about reading spectral coefficients from decision diagrams for different orderings of spectral transform matrices and in particular, for the Haar coefficients in different ordering.

It should be noted that ordering of rows in the Haar matrix should not be confused with the ordering of variables in f, although the rows of the Haar matrix can be expressed in terms of switching variables. The change of the ordering of the Haar functions in the Haar matrix change the order of the Haar coefficients in

the Haar spectrum, but does not change their values. However, for a fixed ordering of rows of the Haar matrix, the change of the order of variables in f changes the values of the Haar coefficients. This is a joint property of local wavelet-like transforms.

Reordering of variables is a particular case of permutation of elements in the vector of function values. The linear transformation of variables is an extension of this method of exploiting reordering of variables for reduction of the number of nonzero Haar coefficients. An exact algorithm for determination of the linear transformation of variables in f which minimizes the number of nonzero Haar coefficients is given in [38]. This linear transformation for a given f is denoted as the optimal transformation in the number of nonzero coefficients count. In terms of word-level decision diagrams [34], this linear transformation produces the minimum number of paths pointing to the nonzero constant nodes, and in many cases reduces also the number of nodes. Notice that in some cases, depending on the functions processed, the linear transformation of variables may reduce to the reordering of variables. Thus, the algorithm by Karpovsky in [38] provides a larger number of permutations of entries in the vector of function values compared to the reorderings corresponding to the variable ordering, and therefore, may produce Haar spectra with smaller number of coefficients.

2.6. Haar spectral transform decision diagrams

The Haar spectral transform decision diagrams (HSTDDs) are defined as decision diagrams which represent f in the form of the Haar series expression for f [47]. For the consistency with the notation in decision diagrams, we will use an ordering of the Haar matrix which is most compatible with decision diagrams. In this ordering, the Haar functions, rows of the Haar matrix, are described in terms of switching variables, as shown in Example 2.8, however, with all the variables in descending order of indices. For example, for $n = 3$, that descending order of variables implies that the Haar functions corresponding to $S_f(5)$ and $S_f(6)$ are permuted. As in other spectral transform decision diagrams (STDDs) [44, 45], in an HSTDD for f, each path corresponds to a term in the Haar expression for f. Therefore, the algorithm by Karpovsky for optimal linear transformation of variables obviously produces HSTDDs with the minimum number of paths.

2.6.1. Haar spectral transform decision trees

Definition 2.14. The Haar spectral transform decision trees (HSTDTs) are defined as the graphic representation of the Haar expression for f. In an HSTDD for f, each path from the root node to a constant node corresponds to a Haar function $\mathrm{har}(w, x)$. The constant nodes show values of Haar coefficients.

Example 2.15. Figure 2.4 shows the HSTDT for $n = 3$ defined by using the nonnormalized Haar transform. This HSTDT represents f in the form of the Haar

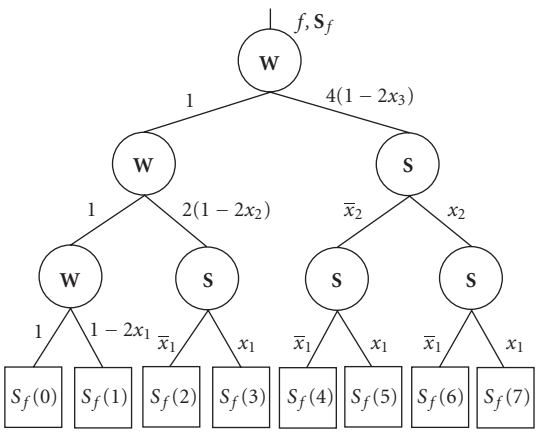


FIGURE 2.4. HSTDT for $n = 3$.

expression for f,

$$\begin{aligned} f = \frac{1}{8}\big(&S_f(0) + S_f(1)(1-2x_1) + 2S_f(2)(1-2x_2)\overline{x}_1 \\ &+ 2S_f(3)(1-2x_2)x_1 + 4Sf(4)(1-2x_3)\overline{x}_2\overline{x}_1 \\ &+ 4S_f(5)(1-2x_3)\overline{x}_2x_1 + 4S_f(6)(1-2x_3)x_2\overline{x}_1 \\ &+ 4S_f(7)(1-2x_3)x_2x_1\big). \end{aligned} \tag{2.13}$$

2.6.2. Haar spectral transform decision diagrams

Definition 2.16. Haar spectral transform decision diagrams (HSTDDs) are derived by the reduction of the corresponding HSTDTs by the generalized BDD reduction rules [45].

Definition 2.17 (generalized BDD reduction rules). (1) Delete all the redundant nodes where both edges point to the same node and connect the incoming edges of the deleted nodes to the corresponding successors. Relabel these edges as shown in Figure 2.5(a).

(2) Share all the equivalent subgraphs, Figure 2.5(b).

Example 2.18. In the ordering of Haar matrix used in the definition of HSTDT for f in Example 2.10, the Haar spectrum is given by

$$\begin{aligned} \mathbf{S}_f = \frac{1}{32}[&18, 6, 0, 2, 2, -4, -2, -6, 0, 0, -2, 2, -2, 0, 0, 0, \\ &0, 2, 0, 2, -2, 0, 0, 0, -2, 2, 0, 0, 0, 0, 0, 0]^T. \end{aligned} \tag{2.14}$$

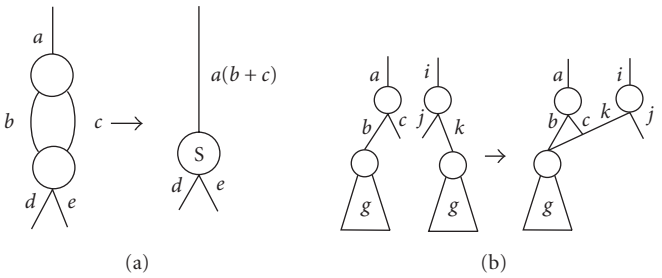


FIGURE 2.5. Generalized BDD rules.

Figure 2.6 shows the HSTDD for f in Example 2.10. This HSTDD represents f in the form of the Haar expression

$$
\begin{aligned}
f = \frac{1}{32}\big(&1 \cdot 18 + 6 \cdot (1 - 2x_1) + 2 \cdot 2(1 - 2x_2)x_1 \\
&+ 2 \cdot 4(1 - 2x_3)\overline{x}_2\overline{x}_1 - 4 \cdot 4(1 - 2x_3)\overline{x}_2 x_1 \\
&- 2 \cdot 4(1 - 2x_3)x_2\overline{x}_1 - 6 \cdot 4(1 - 2x_3)x_2 x_1 \\
&- 2 \cdot 8(1 - 2x_4)\overline{x}_3 x_2\overline{x}_1 + 2 \cdot 8(1 - 2x_4)\overline{x}_3 x_2 x_1 \\
&- 2 \cdot 8(1 - 2x_4)x_3\overline{x}_2\overline{x}_1 + 2 \cdot 16(1 - 2x_5)\overline{x}_4\overline{x}_3\overline{x}_2 x_1 \\
&+ 2 \cdot 16(1 - 2x_5)\overline{x}_4\overline{x}_3 x_2 x_1 - 2 \cdot 16(1 - 2x_5)\overline{x}_4 x_3\overline{x}_2\overline{x}_1 \\
&- 2 \cdot 16(1 - 2x_5)x_4\overline{x}_3\overline{x}_2\overline{x}_1 + 2 \cdot 16(1 - 2x_5)x_4\overline{x}_3\overline{x}_2 x_1\big).
\end{aligned}
\tag{2.15}
$$

2.7. HSTDDs with the minimal number of paths

2.7.1. Haar spectrum with the minimal number of nonzero coefficients

In bit-level decision diagrams, the 0-paths and 1-paths are distinguished. The 1-paths correspond to terms in AND-EXOR expressions in form of which a decision diagram represents f. In the word-level decision diagrams, we consider 0-paths and c-paths, where c is an arbitrary integer or complex number. Thus, a c-path is a path from the root node to a constant node showing the value c. We denote by Q_{DD} the number of c-paths in a given decision diagram DD.

As noticed above, Karpovsky presented in [38] an algorithm for ordering elements in the vector **F** representing a function f defined in 2^n points, such that the Haar spectrum for f has the minimum number of coefficients. The algorithm is based upon the calculation of the total autocorrelation function for f defined as follows.

Definition 2.19 (autocorrelation function). For a given n-variable switching function $f(x)$, $x = (x_0, \ldots, x_n)$, $x_i \in \{0, 1\}$, the autocorrelation function B_f is defined

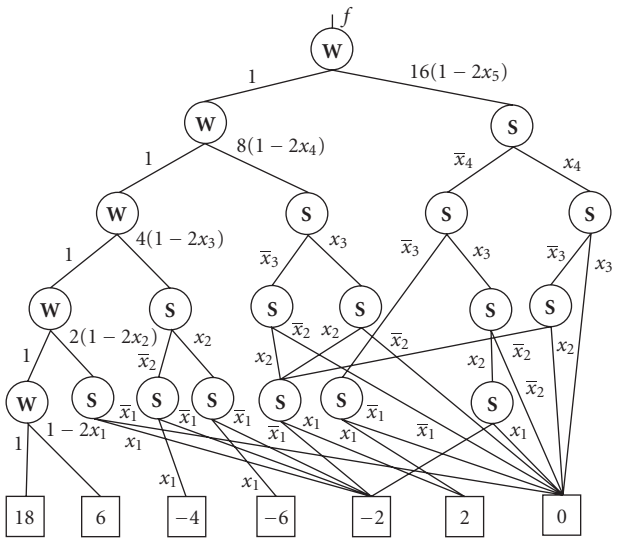


FIGURE 2.6. HSTDD for f in Example 2.10.

as

$$B_f(\tau) = \sum_{x=0}^{2^n-1} f(x) f(x \oplus \tau), \quad \tau \in \{0, \ldots, 2^n - 1\}, \tag{2.16}$$

Definition 2.20 (characteristic functions). For a system of k switching functions $f^{(i)}(x_1, \ldots, x_n)$, $i = 0, \ldots, k-1$, $x_i \in \{0, 1\}$, we define the integer-valued equivalent $f(x) = \sum_{i=0}^{k-1} 2^{k-1-i} f^{(i)}(x)$, $x = (x_1, \ldots, x_n)$. For each distinct value of $f(x)$, we determine the characteristic function f_i defined as

$$f_i(x) = \begin{cases} 1 & \text{if } f_z(x) = i, \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

Definition 2.21 (total autocorrelation function). For a system of k switching functions $f^{(i)}(x_1, \ldots, x_n)$, $i = 0, \ldots, k-1$, the total autocorrelation function is defined as the sum of autocorrelation functions for the characteristic functions $f_i(x)$ for the integer function $f(x)$ assigned to the system. Thus,

$$B_f(\tau) = \sum_{i=0}^{k-1} B_{f_i}(\tau). \tag{2.18}$$

K-procedure

(1) Assign to a given multi-output function $f = (f^{(0)}, \ldots, f^{(k-1)})$, $x = (x_0, \ldots, x_{n-1})$, $x_i \in \{0, 1\}$, an integer equivalent function $f(x)$.

(2) Denote by R the range of $f(x)$ assigned to f. For every $i \in R$, construct characteristic functions $f_i(x)$.

(3) Calculate the autocorrelation functions B_{f_i} for each $f_i(x)$, and the total autocorrelation function $B_f = \sum_i B_{f_i}$.

(4) Determine the n-tuple of input variables $\tau = (x_1, \ldots, x_n)$, where B_f takes the maximum value, excepting the value $B_f(0)$. If there are several choices, select anyone of them.

(5) Determine a matrix σ from the requirement

$$\sigma \odot \tau = (0, \ldots, 0, 1)^T, \tag{2.20}$$

where $\odot$ denotes the multiplication over GF(2).

(6) Determine a function f_σ such that

$$f_\sigma(\sigma \odot x) = f(x). \tag{2.21}$$

That means, reorder values in a vector $\mathbf{F}$ representing values of f by the mapping $x = (x_1, \ldots, x_n) \to x_\sigma$, where $x_\sigma = \sigma^{-1} \odot x$.

(7) In a vector $\mathbf{F}_\sigma$ representing the values of f_σ, perform an encoding of pairs of adjacent values by assigning the same symbol to the identical pairs. Denote the resulting function of $(n-1)$ variables by $\mathbf{Q}_{n-1}$.

(8) Repeat the previous procedure for $i = i - 1$ to some k until there are identical pairs in $\mathbf{Q}_k$.

(9) Determine MTBDD for f_{σ_k}.

End of procedure

Algorithm 2.1

The algorithm by Karpovsky assigns a function f_σ to a given function f, defined as

$$f_\sigma(\sigma \odot x) = f(x), \tag{2.19}$$

where $\odot$ denotes the multiplication modulo 2, and σ is an $(n \times n)$ matrix determining the transformation over the binary representation of arguments of f to determine the elements of $\mathbf{F}_\sigma$ representing f_σ. Thus, $\mathbf{F}_\sigma$ and $\mathbf{F}$ are vectors with equal elements, but in different order uniquely determined by σ. The algorithm determines σ in such a way that the Haar spectrum for f_σ has the minimum number of nonzero coefficients.

The procedure for minimization of the number of nonzero Haar coefficients can be performed through Algorithm 2.1.

Remark 2.22. The K-procedure produces the maximal number of identical pairs of values in $\mathbf{F}$ or, equivalently, pairs of isomorphic subtrees in the MTBDD(f) at the positions pointed by the outgoing edges $\overline{x}_i$ and x_i for all $i = n, n-1, \ldots, 1$.

Example 2.23 (see [38]). Table 2.1 shows a two-output function $f^{(0)}$, $f^{(1)}$ of four variables. This function is represented by the integer equivalent function $f = 2f^{(0)} + f^{(1)}$. This function has the Haar spectrum with 14 nonzero coefficients. The matrix determined by the algorithm proposed by Karpovsky,

$$\sigma = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \tag{2.22}$$

defines a reordering of the variables x_i, $i = 1, 2, 3, 4$, in the binary representation for $x = (x_1, x_2, x_3, x_4)$ through the relation

$$x_\sigma = \sigma^{-1} \odot x. \tag{2.23}$$

Since

$$\sigma^{-1} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \tag{2.24}$$

the vector

$$\begin{aligned} \mathbf{F} = \big[&f(0), f(1), f(2), f(3), f(4), f(5), f(6), f(7), f(8), \\ &f(9), f(10), f(11), f(12), f(13), f(14), f(15)\big]^T \end{aligned} \tag{2.25}$$

is transformed into the vector

$$\begin{aligned} \mathbf{F}_\sigma = \big[&f(0), f(15), f(6), f(9), f(7), f(8), f(1), f(14), \\ &f(11), f(4), f(13), f(2), f(12), f(3), f(10), f(5)\big]^T. \end{aligned} \tag{2.26}$$

The matrix σ defines a function $f_{\sigma\,\min}$ which has the Haar spectrum with 9 nonzero coefficients, compared to the 14 coefficients in the Haar spectrum for f.

2.7.2. HSTDDs for the minimized Haar spectrum

The following consideration permits derivation of optimized HSTDDs by exploiting the K-procedure.

Table 2.1. Function with the minimized Haar spectrum.

x, w	$f^{(0)}, f^{(1)}$	$f(x)$	$16S_f(w)$	$f_\sigma(x)$	$16S_{f_\sigma}$
0	00	0	22	0	22
1	10	2	0	0	2
2	00	0	−5	2	−8
3	01	1	1	0	2
4	10	2	1	3	−2
5	01	1	0	3	−2
6	10	2	−2	2	2
7	11	3	1	2	0
8	11	3	−2	2	0
9	00	0	3	2	0
10	01	1	1	2	0
11	10	2	−1	0	0
12	01	1	−1	1	2
13	10	2	−1	1	2
14	10	2	−1	1	0
15	00	0	2	1	0

The algorithm by Karpovsky for the optimization of the Haar spectrum reduces the number of nonzero coefficients in the Haar spectrum for f. Each nonzero coefficient corresponds to a term in the Haar expression for f. In an HSTDD for f, each c-path from the root node to a constant node corresponds to a term in the Haar expression for f. Thus, this algorithm reduces the number of c-paths in the HSTDD for f.

Remark 2.24. Consider a function f and the function f_σ for f determined by the K-procedure. Denote by $Q_{\mathrm{HSTDD}}(f)$ the number of c-paths in the HSTDD for f. Then, $Q_{\mathrm{HSTDD}}(f_\sigma) < Q_{\mathrm{HSTDD}}(f)$. Moreover, HSTDD for f_σ has the minimum number of c-paths among HSTDDs for functions generated for other possible orderings of elements in the vector **F** representing f.

It is shown in [48] that reduced number of implicant in the disjoint cover of a given function f, which means the reduced number of paths in the BDD for f, does not necessarily implies the reduced size of this BDD and vice versa. As is pointed out in [48], different ordering of variables produces BDDs with different paths and number of nodes, which corresponds to different disjoint covers of functions. However, the size of the disjoint cover, equivalently, the number of paths, cannot be used as a means to determine the best variable ordering in BDDs. The same result is extended to BDDs with complemented edges [48].

We believe that the same applies to any STDD, including MTBDDs, HSDs, and HSTDDs. In this case, instead of implicants, we consider the number of spectral coefficients, since each coefficient is assigned to a term in the function expression in respect to which the considered STDD is defined [45]. Reordering of variables can be considered as a particular example of the linear transform of variables used to reduce the number of Haar coefficients, when σ is a permutation matrix. Therefore, we were interested to consider the impact of the linear transform that reduces the number of Haar coefficients to the size of HSTDDs and MTBDDs.

Reduction of the number of paths in many cases implies the reduction of the size of the decision diagram, since each nonterminal node means branching of edges into two different subpaths. Therefore, the following remark is possible.

Remark 2.25 (size of HSTDD). In many cases, size(HSTD(f_σ)) < size(HSTD(f)).

For $w \geq 2^{n-1}$, we may write $w = 2^{n-1} + j$, where $j = 0, \ldots, 2^{n-1} - 1$. From definition of the Haar functions, for $w \geq 2^{n-1}$, $S_f(w) = 0$ if and only if $f(2j) = f(2j+1)$. In **F**, that property implies constant subvectors of order 2. In MTBDDs, and thus HSDs, this means the possibility to reduce a node at the level corresponding to x_n whose outgoing edges point to $f(2j)$ and $f(2j+1)$. Similar, for $w < 2^{n-1}$, $S_f(w) = 0$ if there are constant or equal subvectors of orders 2^k, $k = 2, \ldots, n-1$ in **F**. Equal and constant subvectors mean possibility to share or delete nonterminal nodes at upper levels in the MTBDD(f).

Remark 2.26 (size of MTBDD and HSD). In many cases, size(MTBDD(f_σ)) < size (MTBDD(f)). Notice that there are exceptions as, for example, the function in [48]. The same remark applies for HSDs, since the size of an HSD is equal to the size of the MTBDD from which it is derived. However, the total amount of data that should be stored in an HSD is increased for the values of nonzero coefficients, compared to that in the MTBDD .

Example 2.27. Figure 2.7.2 shows HSTDD for the function f in Table 2.1, and Figure 2.7.2 shows HSTDD for the function obtained after the linearization f_σ. In this example, size(MTBDD(f)) = 18, size(MTBDD(f_σ)) = 11, size(HSTDD(f)) = 22, size(HSTDD(f_σ)) = 15, $Q_{\mathrm{HSTDD}(f)} = 13$, and $Q_{\mathrm{HSTDD}(f_\sigma)} = 8$, since a path contains a cross-point.

However, unlike the number of c-paths that depends on the number of nonzero coefficients, the size of an HSTDD depends also on the number of different nonzero coefficients and their distribution in the spectrum.

Example 2.28. The output of a two-bit adder is described by the vector

$$\mathbf{F} = [0, 1, 2, 3, 1, 2, 3, 4, 2, 3, 4, 5, 3, 4, 5, 6]^T. \tag{2.27}$$

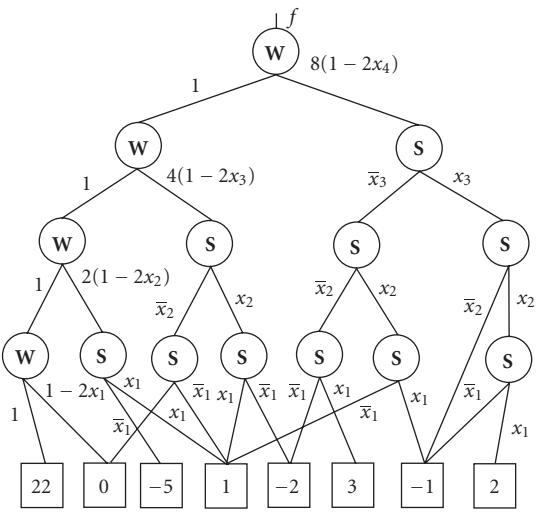


FIGURE 2.7. HSTDD for f in Example 2.27.

The Haar spectrum for this adder is

$$\begin{aligned}\mathbf{S}_{\text{add2}} = [&48, -16, -4, -4, -4, -4, -4, -4,\\ &-1, -1, -1, -1, -1, -1, -1, -1]^T.\end{aligned} \tag{2.28}$$

After the linear transformation of variables, we get the Haar spectrum

$$\mathbf{S}_{\text{add2}_\sigma} = [48, 0, 0, 0, -8, 0, -8, 0, -2, -2, 0, 0, -2, -2, 0, 0]^T. \tag{2.29}$$

Although the number of nonzero coefficient is reduced from 15 to 7, it is size(HSTDD)(add2-LTA) > size(HSTDD)(add2), since in the Haar spectrum for add2 there is a constant vector of order 4 and a constant vector of order four. That permits reduction of a subtree with 7 nonterminal nodes and a subtree with three nonterminal nodes. In the linearly transformed spectrum, equal subvectors are of order two which permits reduction of a single nonterminal node for each constant or equal subvector. Table 2.2 shows the basic characteristics of MTBDD and HSTDD for a two-bit adder. Figures 2.9 and 2.10 show HSTDD(add2) and HSTDD(add2-LTA), respectively.

2.8. Experimental results

We have performed a series of experiments to illustrate the statements in this paper. The experiments are performed over *mcnc* benchmark functions and randomly generated multiple-output switching functions.

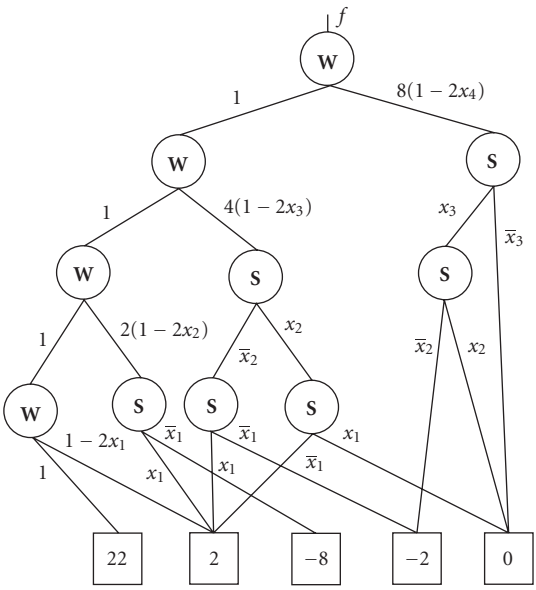


FIGURE 2.8. HSTDD for f_σ in Example 2.27.

TABLE 2.2. Complexity of MTBDD, HSTDs, and HSTDD-LTA for two-bit adder.

MTBDD						
ntn	cn	s	w	paths		
				0	c	total
13	7	20	6	1	15	16
HSTDD						
ntn	cn	s	w	paths		
				0	c	total
4	4	8	1	0	5	5
HSTDD-LTA						
ntn	cn	s	w	paths		
				0	c	total
6	4	10	2	4	3	7

Table 2.3 shows the number of nonzero Haar coefficients in the Haar spectrum for n-bit adders and the Haar spectrum for linearly transformed n-bit adders with linear transform of variables determined by the K-procedure. The number of coefficients is reduced for about 50%.

Table 2.4 compares the characteristics of HSTDDs for adders. Due to this linear transform of variables, the number of c-paths and the number of constant

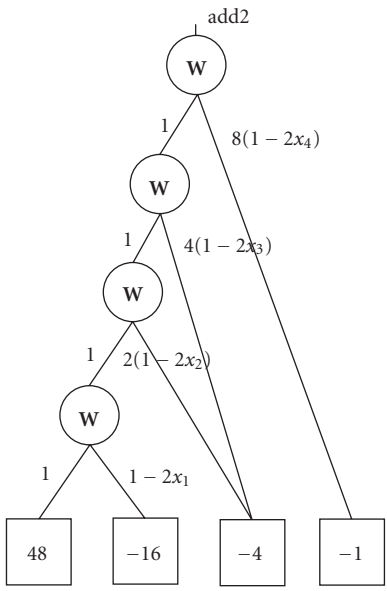


FIGURE 2.9. HSTDD for two-bit adder.

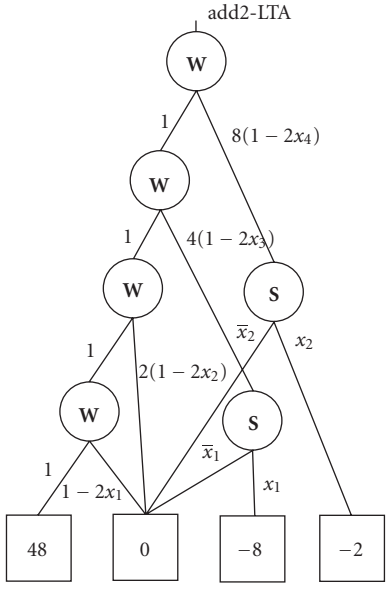


FIGURE 2.10. HSTDD for two-bit adder with linearly transformed variables.

Table 2.3. Haar coefficients for adders.

n	Haar	Haar-LTA	%
2	16	7	56.25
3	64	29	54.69
4	256	121	52.74
5	1024	497	51.47
6	4096	2017	50.76
7	16 384	8129	50.39

Table 2.4. HSTDDs and HSTDD-LTAs for adders.

	HSTDD							HSTDD-LTA						
n	ntn	cn	s	w	paths			ntn	cn	s	w	paths		
					0	c	total					0	c	total
2	4	4	8	1	0	5	5	6	4	10	3	4	3	7
3	6	7	13	1	0	7	7	9	5	14	4	6	4	10
4	8	7	15	1	0	9	9	12	6	18	5	8	5	13
5	10	11	21	1	0	11	11	15	7	22	6	10	6	16
6	12	10	22	1	0	13	13	18	8	26	7	12	7	19
7	14	15	29	1	0	15	15	21	9	30	8	14	8	22

nodes are reduced. However, the number of nonterminal nodes and the width are increased.

It should be noted that HSTDDs and HSTDD-LTA are quite smaller than MTBDDs with respect to all the characteristics of a decision diagram. We consider that as a justification of study of HSTDDs.

Table 2.5 illustrates the impact of the linear transform of variables determined by the K-procedure to the characteristics of MTBDDs. This transform reduces all the characteristics of MTBDDs.

Table 2.6 compares the number of nonzero coefficients in the Haar spectrum and the Haar spectrum after the linear transform of variables for some benchmark functions and randomly generated functions. The savings in the number of nonzero coefficients range from 0.81% to 88.24%. The average savings for benchmark functions are 49.48%, and 12.254% for random generated functions.

Tables 2.7 and 2.8 compare the complexity of HSTDDs before and after the linear transform of variables determined by the K-procedure. It is shown the number of nonterminal nodes (ntn), constant nodes (cn), whose sum is the size of the HSTDD (s = ntn + cn), and the width (w) of HSTDDs. We also show the number of 0-paths, c-paths, and the total number of paths. The K-procedure reduced the number of c-paths and the width of HSTDDs for all the considered functions. For 14 functions, the size is reduced. For 16 functions, the number of nonterminal nodes is reduced. For 11 functions, the number of constant nodes is reduced. In

Table 2.5. MTBDDs and MTBDD-LTAs for adders.

	MTBDD						
n	ntn	cn	s	w	paths		
					0	c	total
2	13	7	20	6	1	15	16
3	41	15	56	14	1	63	64
4	113	31	144	30	1	255	256
5	289	63	352	62	1	1023	1024
6	705	127	832	126	1	4096	4097
7	1665	255	1920	254	1	16383	16384
	LTA-MTBDD						
n	ntn	cn	s	w	paths		
					0	c	total
2	8	7	15	3	1	8	9
3	24	15	39	7	1	26	27
4	64	31	95	15	1	80	81
5	160	63	223	31	1	242	243
6	384	127	511	63	1	728	729
7	896	255	1151	127	1	2186	2187

other functions, these parameters are equal or increased as a price for the reduced number of c-paths and the width of HSTDDs.

Tables 2.9 and 2.10 compare the complexity of MTBDDs for the same set of functions before and after the application of the linear transform of variables used in HSTDDs. The linear transform used in HSTDDs reduced the number of c-paths for all the considered functions. The width is reduced for 14 functions, and size for 23 functions.

2.9. Multivalued Haar functions

In the previous sections, we started from the Haar functions and related expressions and defined the Haar spectral transform decision diagrams as the graphical form of these expressions. The discrete Haar functions are used to determine labels at the edges of these diagrams. For an extension of the Haar functions to multivalued case, we do the opposite. We generalize the definition of Haar spectral transform decision diagrams by appropriately selecting the decomposition rules at the nonterminal nodes. Then we define the corresponding multivalued Haar functions by traversing paths from the root node to the constant nodes in these diagrams.

The BDDs, MTBDDs, and HSTDDs discussed above are examples of decision diagrams in the space $C(C_2)$ of complex-valued functions defined in 2^n points, that is, defined on the group C_2^n, where C_2 is the cyclic group of order 2.

Table 2.6. Number of coefficients in the Haar spectrum and the Haar-LTA spectrum.

f	Haar	LTA-Haar	%
9sym	211	106	49.77
ex1010	989	971	1.83
misex1	32	28	12.50
rd53	32	22	31.25
rd73	128	32	75.00
rd84	265	64	75.85
xor5	17	2	88.24
add2	16	7	56.25
add3	64	29	54.69
f_1	161	131	18.64
f_6	141	122	13.48
f_7	163	133	18.41
f_8	234	212	8.41
f_9	247	245	0.81
n_1	162	136	1.05
n_2	169	108	36.10
n_3	165	155	6.07
n_4	165	152	7.88
n_5	156	139	10.90
n_6	147	125	14.97
n_7	150	127	15.34
n_8	150	133	11.34
n_9	161	137	14.10
n_{10}	273	242	11.36

In this section, we discuss generalization of the definition of Haar functions to multivalued case, that is, we consider the discrete Haar functions defined in p^n, $p >$, points, and taking values in a field P, that can be the finite Galois field GF(p) or the complex field C. These functions are defined by referring to paths in the corresponding decision diagrams for multivalued functions. Notice that in this case, multiple-place decision diagrams (MDDs) [49, 50], being defined with respect to the identity transform on the cyclic group C_p of order p performed at each node, are counterparts of BDDs, since the values of constant nodes are in P. A generalization of MTBDDs, is derived by allowing in MDDs complex numbers as the values of constant nodes [51]. We denote these decision diagrams as multi-terminal decision diagrams (MTDDs), that is, we delete the word binary, since in this case there are p outgoing edges of a node.

Definition 2.29. For $f \in P(C_p^n)$, Haar spectral transform multivalued decision trees (HSTMVDTs) are decision trees where the expansion rules at the leftmost

Table 2.7. Complexity of HSTDs.

f	HSTDD						
	ntn	cn	s	w	paths		
					0	c	total
9sym	111	16	127	26	189	197	381
ex1010	1023	639	1662	512	35	989	1024
misex1	34	17	51	11	22	18	40
rd53	23	9	32	9	0	30	30
rd73	52	11	63	14	0	103	103
rd84	74	16	90	18	0	201	201
xor5	12	4	16	3	4	17	21
add2	4	4	8	1	0	5	5
add3	6	7	13	1	0	7	7
f_1	137	13	150	47	72	140	212
f_6	135	13	148	46	80	122	202
f_7	137	10	147	49	68	137	205
f_8	220	33	253	93	22	227	249
f_9	250	55	305	123	9	246	255
f_{10}	192	26	218	66	37	208	245
n_1	150	13	163	52	70	143	213
n_2	155	13	168	58	65	150	215
n_3	160	15	175	58	69	156	225
n_4	160	14	174	56	68	155	223
n_5	152	17	169	52	70	147	217
n_6	130	12	142	46	71	125	196
n_7	133	10	143	48	73	131	204
n_8	135	11	146	48	79	133	212
n_9	137	11	148	51	74	142	216
n_{10}	235	15	250	73	158	234	392

nodes are defined by a basis Q for the Fourier series or a polynomial expressions in $P(C_p)$ and for other nodes by the identity transforms for functions in $P(C_p)$.

Definition 2.30. Haar functions in $P(C_p^n)$, are functions described by products of labels at the edges along the paths from the root node to the constant nodes in the HSTMVDTs for functions in $P(C_p)$.

Algorithm 2.2 performs construction of HSTMVDT in $P(C_p)$ [52].

When an HSTMVDT is constructed, we determine the corresponding multivalued Haar-like functions by multiplying labels at the edges in these decision trees. The method will be illustrated by the example of the Haar functions

Table 2.8. Complexity of HSTDD-LTA.

f	HSTDD-LTA						
	ntn	cn	s	w	paths		
					0	c	total
9sym	89	13	102	25	85	87	172
ex1010	1019	636	1655	508	53	969	1022
misex1	41	17	58	13	26	18	44
rd53	19	10	29	7	8	19	27
rd73	20	6	26	7	0	23	23
rd84	32	8	40	10	0	44	44
xor5	5	3	8	1	4	2	6
add2	6	4	10	2	4	3	7
add3	9	5	14	2	6	4	10
f_1	129	16	135	41	76	114	190
f_6	134	17	151	43	83	110	193
f_7	134	14	148	45	83	120	203
f_8	223	36	259	97	40	205	245
f_9	241	52	293	114	11	238	249
f_{10}	198	26	224	71	37	210	247
n_1	118	42	160	55	18	107	125
n_2	118	42	160	55	18	107	125
n_3	91	13	104	31	38	77	115
n_4	153	19	172	55	76	139	215
n_5	155	20	175	52	82	132	214
n_6	132	14	146	43	79	111	190
n_7	135	14	149	45	83	118	201
n_8	134	13	147	43	82	117	199
n_9	137	12	149	46	80	122	202
n_{10}	238	20	258	73	157	228	385

defined by using the basic Vilenkin-Chrestenson transform for $p = 3$ instead of the Walsh transform that is used in definition of the discrete Haar functions through the HSTMVDTs, as discussed in Section 2.6.

Example 2.31. For $p = 3$, the basic Vilenkin-Chrestenson matrix is given by

$$\mathbf{VC}_3(1) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & e_1 & e_2 \\ 1 & e_2 & e_1 \end{bmatrix}, \tag{2.30}$$

Table 2.9. Complexity of MTBDDs.

f	MTBDD						
	ntn	cn	s	w	paths		
					0	c	total
9sym	33	2	35	6	72	148	220
ex1010	894	177	1071	383	190	800	990
misex1	17	11	28	6	5	13	18
rd53	15	6	21	5	1	31	32
rd73	28	8	36	7	1	127	128
rd84	36	9	45	8	1	255	256
xor5	9	2	11	2	16	16	32
add2	13	7	20	6	1	15	16
add3	41	15	66	14	1	63	64
f_1	75	2	77	30	88	82	170
f_6	58	2	60	18	61	74	135
f_7	72	2	74	28	86	87	173
f_8	174	8	182	64	33	210	243
f_9	222	16	238	95	20	229	249
f_{10}	139	5	144	56	49	171	220
n_1	84	3	87	30	13	151	164
n_2	89	3	92	29	22	148	170
n_3	91	3	94	31	34	136	170
n_4	90	3	93	31	40	129	169
n_5	82	3	85	27	48	108	156
n_6	68	2	70	25	56	93	149
n_7	78	2	80	28	62	92	154
n_8	72	2	74	29	65	87	152
n_9	73	2	75	28	75	88	163
n_{10}	118	2	120	41	176	113	289

where $e_1 = -(1/2)(1 - i\sqrt{3})$, and $e_2 = e_1^* = -(1/2)(1 + i\sqrt{3})$, where z^* denotes the complex-conjugate of z.

The Vilenkin-Chrestenson transform is defined by the transform matrix

$$\mathbf{VC}^{-1} = \mathbf{VC}^* = \otimes_{i=1}^{n} \mathbf{VC}_3(1). \tag{2.31}$$

Vilenkin-Chrestenson-Haar spectral transform decision trees (VCHSTDTs) are defined as decision trees where the expansion rule for the leftmost nodes is

Table 2.10. Complexity of MTBDD-LTAs.

f	MTBDD-LTA						
	ntn	cn	s	w	paths		
					0	c	total
9sym	24	2	26	5	27	48	75
ex1010	871	177	1048	367	180	791	971
misex1	17	11	28	5	4	14	18
rd53	17	6	23	6	1	23	24
rd73	17	7	24	6	1	31	32
rd84	23	8	41	7	1	63	64
xor5	1	2	3	1	1	1	2
add2	8	7	15	3	1	8	9
add3	24	15	39	7	1	26	27
f_1	66	2	68	24	2	63	65
f_6	69	2	71	25	58	64	122
f_7	67	2	69	23	62	64	126
f_8	168	8	176	59	29	189	218
f_9	219	16	235	94	19	226	245
f_{10}	136	5	141	54	48	170	218
n_1	80	3	83	26	14	122	136
n_2	80	3	83	26	21	135	156
n_3	50	3	53	15	18	64	82
n_4	89	3	92	28	39	111	150
n_5	77	3	80	24	43	93	136
n_6	62	2	64	19	46	70	116
n_7	63	2	65	20	50	70	120
n_8	72	2	74	28	58	78	136
n_9	68	2	70	25	61	68	129
n_{10}	115	2	117	42	140	102	242

determined by the matrix $\mathbf{VC}(1)$, and for the other nodes by the identity matrix

$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.32}$$

In VCHSTDTs, the values of constant nodes are the Vilenkin-Chrestenson coefficients. The labels at the edges are determined by the analytic expression for columns of $\mathbf{VC}_3(1)$. Thus, they are $r_0 = 1$, $r_1 = 1 + vx + dx^2$, $r_2 = 1 + v^*x + d^*x^2$, where $v = -i\sqrt{3}$, and $d = -(3/2)(1 - i\sqrt{3})$.

Figure 2.9 shows VCHSTDT for $n = 2$. In this VCHSTDT, products of labels at the edges determine the Haar functions in $C(C_3^2)$ [38]. In the notation used in

(1) Given an MTDT in $P(C_p^n)$.
(2) Determine the positive Davio (pD) expansion from the basic matrix for the Fourier transform or the polynomial expressions in $P(C_p)$.
(3) In MTDD, assign the pD-expansion to the leftmost nodes and relabel the outgoing edges of these nodes.
(4) Reorder the variables in the descending order.
(5) Determine the columns of a $(p^n \times p^n)$ matrix $\mathbf{Q}$ as product of labels at the edges.
(6) Calculate the values of constant nodes by using the $\mathbf{Q}^{-1}$ inverse for $\mathbf{Q}$ over P.

Algorithm 2.2. Design of HSTMVDT.

VCHSTDTs, they are

$$\begin{aligned}
\mathrm{har}_3(0) &= 1,\\
\mathrm{har}_3(1) &= 1 + vx_1 + dx_1^2,\\
\mathrm{har}_3(2) &= 1 + v^* x_1 + d^* x_1^2,\\
\mathrm{har}_3(3) &= (1 + vx_2 + dx_2^2) J_0(x_1),\\
\mathrm{har}_3(4) &= (1 + vx_2 + dx_2^2) J_1(x_1),\\
\mathrm{har}_3(5) &= (1 + vx_2 + dx_2^2) J_2(x_1),\\
\mathrm{har}_3(6) &= (1 + v^* x_2 + d^* x_2) J_0(x_1),\\
\mathrm{har}_3(7) &= (1 + v^* x_2 + d^* x_2) J_1(x_1),\\
\mathrm{har}_3(8) &= (1 + v^* x_2 + d^* x_2^2) J_2(x_1),
\end{aligned} \tag{2.33}$$

where $J_i(x_j)$ are characteristic functions defined as $J_i(x_j) = 1$ for $x_j = i$, and $J_i(x_j) = 0$ for $x_j \neq i$.

In Figure 2.9, S_3 denotes the generalized Shannon expansion in $C(C_3)$ defined as

$$f = J_0(x_i) f_0 + J_1(x_i) f_1 + J_2(x_i) f_2, \tag{2.34}$$

where f_i, $i = 0, 1, 2$, are cofactors of f for $x_i \in 0, 1, 2$. The nodes labeled by VC_3 are the positive Davio nodes representing the positive Davio expansion defined from $\mathbf{VC}_3(1)$ as

$$\begin{aligned}
f &= 1 \cdot S_{f_0} + (1 + vx_i + dx_i^2) S_{f_1} + (1 + v^* x_i + d^* x_i^2) S_{f_2}\\
&= 1 \cdot (f_0 + f_1 + f_2) + (1 + vx_i + dx_i^2)(f_0 + e_2 f_1 + e_1 f_2)\\
&\quad + (1 + v^* x_i + d^* x_i^2)(f_0 + e_1 f_1 + e_2 f_2).
\end{aligned} \tag{2.35}$$

The Vilenkin-Chrestenson-Haar transform defined by the VCHSTDT in Figure 2.9 is equivalent up to reordering to the Watari transform [53–55]. This

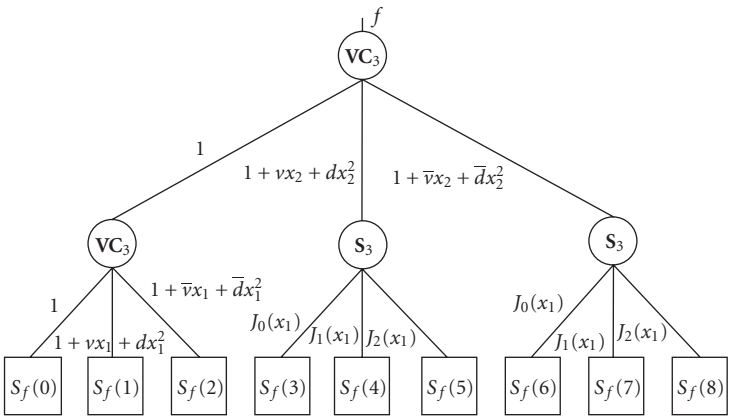


FIGURE 2.11. VCHSTDT for $n = 2$.

means that the decision tree discussed above may be properly modified to generate Watari transforms for different parameters.

Further, by selecting some other transforms to determine expansion rules in the leftmost nodes, various classes of p-valued Haar-like functions can be defined. For instance, we can select the Galois field transforms on C_p over GF(p) and define the corresponding decision diagrams and Haar functions.

2.10. Closing remarks

HSTDDs introduced in this chapter are a representative of STDDs. A given function f is assigned to an HSTDD through the Haar expression for f.

The algorithm by Karpovsky permits to minimize the number of paths pointing to the nonzero constant nodes in an HSTDD. In this way, HSTDDs belong to rare examples of decision diagrams with a deterministic algorithm for minimization of a characteristic of decision diagrams. Minimization of the number of paths usually implies minimization of the width and, in many cases, the size of the HSTDDs. It is believed that HSTDDs are useful in applications where descending of the decision diagrams many times is required.

The same linear transform of variables used in HSTDDs provides for the reduction of the number of c-paths, the width, and in many cases, the size of MTBDDs.

Since HSDs are MTBDDs with Haar coefficients assigned to the edges, the complexity of HSDs is equal to that of MTBDDs increased for the space to store the values of Haar coefficients.

Definition of Haar spectral transform decision diagrams can be directly extended to the corresponding diagrams for multivalued functions by appropriately selecting the basic transforms performed at the nodes. Conversely, thus defined decision diagrams determine the p-valued Haar functions.

Acknowledgment

This work was supported by the Academy of Finland, Finnish Center of Excellence Programme, Grant no. 213462.

Bibliography

[1] S. L. Hurst, *Logical Processing of Digital Signals*, Crane Russak and Edward Arnold, London, UK, 1978.

[2] S. L. Hurst, "The Haar transform in digital network synthesis," in *Proceedings of the 11th International Symposium on Multiple-Valued Logic (ISMVL '81)*, pp. 10–18, Oklahoma City, Okla, USA, May 1981.

[3] A. Haar, "Zur theorie der orthogonalen Funktionsysteme," *Mathematische Annalen*, vol. 69, no. 3, pp. 331–371, 1910.

[4] L. P. Yaroslavsky, *Digital Picture Processing*, Springer, Berlin, Germany, 1985.

[5] B. J. Falkowski, "Relationship between arithmetic and Haar wavelet transforms in the form of layered Kronecker matrices," *Electronics Letters*, vol. 35, no. 10, pp. 799–800, 1999.

[6] B. J. Falkowski, "Relationship between Haar and Reed-Müller spectral and functional domains," *IEICE Electronics Express*, vol. 2, no. 1, pp. 37–42, 2005.

[7] M. A. Thornton, R. Drechsler, and D. M. Miller, *Spectral Techniques in VLSI CAD*, Kluwer Academic, Boston, Mass, USA, 2001.

[8] M. A. Thornton, D. M. Miller, and R. Drechsler, "Transformations amongst the Walsh, Haar, arithmetic and Reed-Müller spectral domain," in *Proceedings of 5th International Workshop on Applications of the Reed-Müller Expansion in Circuit Design (Reed-Müller '01)*, pp. 215–225, Starkville, Miss, USA, August 2001.

[9] L. Zhihua and Q. Zhang, "Introduction to bridge functions," *IEEE Transactions on Electromagnetic Compatibility*, vol. 25, no. 4, pp. 459–464, 1983.

[10] Q. Zhang, "A summary of bridge functions," in *Proceedings of 5th International Workshop on Spectral Techniques*, pp. 128–135, Beijing, China, March 1994.

[11] R. S. Stanković, M. S. Stanković, and C. Moraga, "Remarks on systems and differential operators on groups," *Facta Universitatis, Series: Electronics and Energetics*, vol. 18, no. 3, pp. 531–545, 2005.

[12] R. S. Stanković and M. R. Stojić, "A note on the discrete Haar derivative," in *Proceedings of Alfred Haar Memorial Conference*, vol. 49 of *Colloquia Mathematica Societatis János Bolyai*, pp. 895–907, Budapest, Hungary, 1985.

[13] N. N. Aizenberg, V. P. Rudko, and E. V. Sisuev, "Haar functions and discrete mappings," *Tekhnicheskaya Kibernetika*, vol. 6, pp. 86–94, 1975 (Russian).

[14] R. S. Stanković and M. R. Stojić, "A note on the discrete generalized Haar derivative," *Automatika*, vol. 28, no. 3-4, pp. 117–122, 1987.

[15] R. S. Stanković, "A note on differential operators on finite Abelian groups," *Cybernetics and Systems*, vol. 18, no. 3, pp. 221–231, 1987.

[16] M. S. Stanković and N. N. Aizenberg, "Generalized discrete Gibbs derivatives and related linear equations," in *Theory and Applications of Gibbs Derivatives*, P. L. Butzer and R. S. Stanković, Eds., pp. 249–268, Mathematički Institut, Belgrade, Yugoslavia, 1990.

[17] M. G. Karpovsky, R. S. Stanković, and C. Moraga, "Spectral techniques in binary and multiple-valued switching theory," in *Proceedings of 31st IEEE International Symposium on Multiple-Valued Logic (ISMVL '01)*, p. 41, Warsaw, Poland, May 2001.

[18] K. Egiazarian and J. Astola, "Adaptive Haar transforms with arbitrary time and scale splitting," in *Nonlinear Image Processing and Pattern Analysis XII*, E. R. Dougherty and J. Astola, Eds., vol. 4304 of *Proceedings of SPIE*, pp. 242–255, San Jose, Calif, USA, January 2001.

[19] K. Egiazarian and J. Astola, "Tree-structured Haar transforms," *Journal of Mathematical Imaging and Vision*, vol. 16, no. 3, pp. 269–279, 2002.

[20] K. Egiazarian, A. Gotchev, and J. Astola, "On tree-structured Haar wavelets and their extension to Legendre multi-wavelets," in *Proceedings of the Conference on Computer Science and Information Technologies (CSIT '01)*, pp. 275–282, Yerevan, Armenia, September 2001.

[21] A. Gotchev, D. Rusanovskyy, R. Popov, K. Egiazarian, and J. Astola, "Feature extraction by best anisotropic Haar bases in an ORC system," in *Image Processing: Algorithms and Systems III*, E. R. Dougherty, J. T. Astola, and K. O. Egiazarian, Eds., vol. 5298 of *Proceedings of SPIE*, pp. 504–515, San Jose, Calif, USA, January 2004.

[22] E. Pogossova, K. Egiazarian, and J. Astola, "Fast algorithms and applications of TSH transform," in *Proceedings of International TICSP Workshop on Spectral Methods and Multirate Signal Processing (SMMSP '02)*, pp. 211–218, Tolouse, France, September 2002.

[23] E. Pogossova, K. Egiazarian, and J. Astola, "Signal denoising in tree-structured Haar basis," in *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis (ISPA '03)*, S. Lončarić, A. Neri, and H. Babić, Eds., vol. 2, pp. 736–739, Rome, Italy, September 2003.

[24] R. Popov, A. Gotchev, K. Egiazarian, and J. Astola, "Libraries of anisotropic non-dyadic Haar bases," in *Proceedings of the 5th International Conference and Exhibition on Digital Signal Processing and Its Applications*, pp. 169–171, Moscow, Russia, March 2003.

[25] C. H. Chang and B. J. Falkowski, "Haar spectra based entropy approach to quasi-minimization of FBDDs," *IEE Proceedings, Computers and Digital Techniques*, vol. 146, no. 1, pp. 41–49, 1999.

[26] B. J. Falkowski and C.-H. Chang, "Forward and inverse transformations between Haar spectra and ordered binary decision diagrams of Boolean functions," *IEEE Transactions on Computers*, vol. 46, no. 11, pp. 1272–1279, 1997.

[27] J. P. Hansen and M. Sekine, "Decision diagram based techniques for the Haar wavelet transform," in *Proceedings of the 1st International Conference on Information, Communications, and Signal Processing (ICICS '97)*, vol. 1, pp. 59–63, Singapore, September 1997.

[28] M. Stanković, D. Janković, and R. S. Stanković, "Efficient algorithms for Haar spectrum calculation," *Scientific Review*, no. 21-22, pp. 171–182, 1996.

[29] R. S. Stanković and B. J. Falkowski, "Haar functions and transforms and their generalizations," in *Proceedings of the 1st International Conference on Information, Communications, and Signal Processing (ICICS '97)*, vol. 4, pp. 1–5, Singapore, September 1997.

[30] M. A. Thornton, "Modified Haar transform calculation using digital circuit output probabilities," in *Proceedings of the 1st International Conference on Information, Communications, and Signal Processing (ICICS '97)*, vol. 1, pp. 52–58, Singapore, September 1997.

[31] M. A. Thornton, R. Drechsler, and W. Günther, "Probabilistic equivalence checking using partial Haar spectral diagrams," in *Proceedings of 4th International Workshop on Applications of the Reed-Müller Expansion in Circuit Design (Reed-Müller '99)*, pp. 123–132, Victoria, BC, Canada, August 1999.

[32] M. A. Thornton, R. Drechsler, and W. Günther, "A method for approximate equivalence checking," in *Proceedings of the 30th International Symposium on Multiple-Valued Logic (ISMVL '00)*, pp. 447–452, Portland, Ore, USA, May 2000.

[33] R. S. Stanković, "Some remarks on basic characteristics of decision diagrams," in *Proceedings of 4th International Workshop on Applications of the Reed-Müller Expansion in Circuit Design (Reed-Müller '99)*, pp. 139–146, Victoria, BC, Canada, August 1999.

[34] T. Sasao and M. Fujita, *Representations of Discrete Functions*, Kluwer Academic, Boston, Mass, USA, 1996.

[35] S. Nagayama and T. Sasao, "On the optimization of heterogeneous MDDs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 11, pp. 1645–1659, 2005.

[36] R. S. Stanković and J. T. Astola, *Spectral Interpretation of Decision Diagrams*, Springer, New York, NY, USA, 2003.

[37] R. S. Stanković and T. Sasao, "Decision diagrams for discrete functions: classification and unified interpretation," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '98)*, pp. 439–446, Yokohama, Japan, February 1998.

[38] M. G. Karpovsky, *Finite Orthogonal Series in the Design of Digital Devices*, John Wiley & Sons, New York, NY, USA, 1976.

[39] R. Drechsler and B. Becker, *Binary Decision Diagrams - Theory and Implementations*, Kluwer Academic, Boston, Mass, USA, 1998.

[40] Z. Gongli and C. Moraga, "Orthogonal transforms on finite discrete Abelian groups," in *Recent Developments in Abstract Harmonic Analysis with Applications in Signal Processing*, R. S. Stanković, M. R. Stojić, and M. S. Stanković, Eds., pp. 293–304, Nauka and Faculty of Electronics Niš, Belgrade, Yugoslavia, 1996.

[41] E. M. Clarke, X. Zhao, M. Fujita, Y. Matsunaga, and R. McGeer, "Fast Walsh transform computation with binary decision diagram," in *Proceedings of IFIP WG 10.5 International Workshop on Applications of the Reed-Müller Expansion in Circuit Design (Reed-Müller '93)*, pp. 82–85, Hamburg, Germany, September 1993.

[42] Y.-T. Lai, M. Pedram, and S. B. K. Vrudhula, "EVBDD-based algorithms for integer linear programming, spectral transformation, and function decomposition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 8, pp. 959–975, 1994.

[43] R. S. Stanković, "Some remarks about spectral transform interpretation of MTBDDs and EVBDDs," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '95)*, pp. 385–390, Massa, Chiba, Japan, August-September 1995.

[44] R. S. Stanković, *Spectral Transform Decision Diagrams in Simple Questions and Simple Answers*, Nauka, Belgrade, Yugoslavia, 1998.

[45] R. S. Stanković, T. Sasao, and C. Moraga, "Spectral transform decision diagrams," in *Representations of Discrete Functions*, T. Sasao and M. Fujita, Eds., pp. 55–92, Kluwer Academic, Boston, Mass, USA, 1996.

[46] R. S. Stanković, M. Stanković, and D. Janković, *Spectral Transforms in Switching Theory, Definitions and Calculations*, Nauka, Belgrade, Yugoslavia, 1998.

[47] R. S. Stanković, M. Stanković, J. T. Astola, and K. Egiazaraian, "Haar spectral transform decision diagrams with exact algorithm for minimization of the number of paths," in *Proceedings of 4th International Workshop on Boolean Problems*, pp. 99–106, Freiberg, Germany, September 2000.

[48] E. V. Dubrova and D. M. Miller, "On disjoint covers and ROBDD size," in *Proceedings of IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing (PACRIM '99)*, pp. 162–164, Victoria, BC, Canada, August 1999.

[49] A. Srinivasan, T. Ham, S. Malik, and R. K. Brayton, "Algorithms for discrete function manipulation," in *Proceedings of IEEE International Conference on Computer-Aided Design (ICCAD '90)*, pp. 92–95, Santa Clara, Calif, USA, November 1990.

[50] A. Thayse, M. Davio, and J.-P. Deschamps, "Optimization of multiple-valued decision diagrams," in *Proceedings of 8th International Symposium on Multiple-Valued Logic (ISMVL '78)*, pp. 171–177, Rosemont, Ill, USA, May 1978.

[51] D. M. Miller, "Spectral transformation of multiple-valued decision diagrams," in *Proceedings of the 24th International Symposium on Multiple-Valued Logic (ISMVL '94)*, pp. 89–96, Boston, Mass, USA, May 1994.

[52] R. S. Stanković, M. Stanković, and C. Moraga, "Design of Haar wavelet transforms and Haar spectral transform decision diagrams for multiple-valued functions," in *Proceedings of 31st IEEE International Symposium on Multiple-Valued Logic (ISMVL '01)*, pp. 311–316, Warsaw, Poland, May 2001.

[53] C. Moraga, "The two-dimensional Zhang-Watari orthogonal transform," Forschungsbericht 567, Fachbereich Informatik, Universität Dortmund, Dortmund, Germany, 1995.

[54] R. Oenning and C. Moraga, "Properties of the Zhang-Watari transform," in *Proceedings of 25th International Symposium on Multiple-Valued Logic (ISMVL '95)*, pp. 44–49, Bloomington, Ind, USA, May 1995.

[55] C. Watari, "A generalization of Haar functions," *Tohoku Mathematical Journal*, vol. 8, pp. 286–290, 1956.

Radomir S. Stanković: Department of Computer Science, Faculty of Electronics, University of Niš, 18 000 Niš, Serbia

Email: rstankovic@bankerinter.net

Karen Egiazarian: Institute of Signal Processing, Tampere University of Technology, 33101 Tampere, Finland

Email: karen.egiazarian@tut.fi

Jaakko Astola: Institute of Signal Processing, Tampere University of Technology, 33101 Tampere, Finland

Email: jaakko.astola@tut.fi

3 Discrete transforms, fast algorithms, and point spread functions of numerical reconstruction of digitally recorded holograms

Leonid P. Yaroslavsky

In digital reconstruction of holograms, holograms are sampled first by CCD or CMOS photosensitive cameras and the array of numbers obtained is then subjected in computer to digital transformations that imitate wave back propagation from the camera to the object plane. As a result, samples of the object wave front amplitude are obtained. Accuracy and computational complexity of numerical implementation of the wave propagation transforms are of primary importance for digital holography. The chapter addresses these problems. Specifically, (i) different versions are introduced of discrete representations of the integral Fourier, Fresnel and Kirchhoff-Reileigh-Zommerfeld transforms that correspond to different geometries of hologram and reconstructed image sampling, including canonical discrete Fourier transform (DFT), scaled shifted DFT, rotated DFT, affine DFT, canonical discrete Fresnel transform (DFrT), scaled shifted DFrT, partial DFrT, convolutional DFrT; (ii) fast computational algorithms for these transforms are outlined, and (iii) point spread functions of different hologram reconstruction algorithms are derived that show how reconstruction results depend on the holographic setup and photographic camera physical parameters such as object-to-camera distance, radiation wavelength, camera size, pitch, fill factor, and the like.

3.1. Introduction

Digital recording and numerical reconstruction of optical holograms in computers is a revolutionary breakthrough in optical holography and metrology. It eliminates the need in photochemical development for recording holograms, allows real time reconstruction, in a numerical form, of both amplitude and phase of the object wave front, and directly provides numerical data that can be in a straightforward way used in frame of modern informational technologies. In addition, it allows employing flexibility of digital signal and image processing tools for adaptive correction of imperfections of analog optics and optical sensors. Although first attempts to use digital computers for numerical reconstruction of optical holograms date back to 1968–1972 (see [1–3]), only relatively recently computers

and electro-optical light modulators and sensors reached the level high enough to make this application practical (see [4–6]).

In digital reconstruction of holograms, holograms are first sampled by CCD or CMOS photosensitive cameras and arrays of obtained numbers are then subjected in computer to digital transformations aimed at the reconstruction of samples of the object wave front. The core of these transformations is computation of diffraction integral transforms that describe wave back propagation from holograms to objects. Accuracy in numerical implementation of wave propagation transformations and computational complexity of the implementations are of primary importance for digital holography.

A quite number of methods for numerical integration are known in computational mathematics. However, standard numerical integration methods are intended for integration of analytically defined functions and are not fully adequate to the situation one faces in the problem of numerical reconstruction of holograms where the number of hologram samples is fixed and the option of increasing this number to achieve better integration accuracy, which is conventional for numerical integration methods, does not exists. Moreover, these methods do not take into consideration parameters of the process of hologram sampling.

In this chapter, we present an approach to discrete representation of diffractional transforms that is based on signal sampling theory and oriented on the use of fast computational algorithms. In Section 3.2, we outline principles of digital recording holograms and introduce major diffraction integrals. In Section 3.3, sampling theory-based principles of discrete representation of diffraction integral transforms are formulated and in Section 3.4, they are applied to show how hologram sampling results in the sampling of the integral transform kernel and leads to different versions of the discrete representation of the Fourier integral transform which plays the most fundamental role in holography, those of canonical discrete Fourier transform, shifted and scaled discrete Fourier transforms, affine and rotated, and scaled 2D discrete Fourier transforms. In Section 3.5, these principles are extended to derivation of different versions of discrete representation of Fresnel integral transform that are defined by ways in which hologram and object wave field are supposed to be sampled and by the needs of applications. Then, in Section 3.6, discrete representation of the most general diffraction transform, Kirchhoff-Rayleigh-Sommerfeld integral transform, is given. All introduced discrete transforms are oriented on the use of fast FFT-type algorithms for their computational implementation, and the ways in which it can be done are described.

Section 3.7 opens the second part of this chapter devoted to the analysis of metrological properties of the discrete transforms in the numerical reconstruction of holograms. These properties are formulated in terms of point spread functions of the numerical reconstruction processes and it is shown how they depend on the physical parameters of recording and sampling of holograms and on the type of the reconstruction algorithm. Some mathematical details and a summary table of the discrete transforms are collected in the appendix.

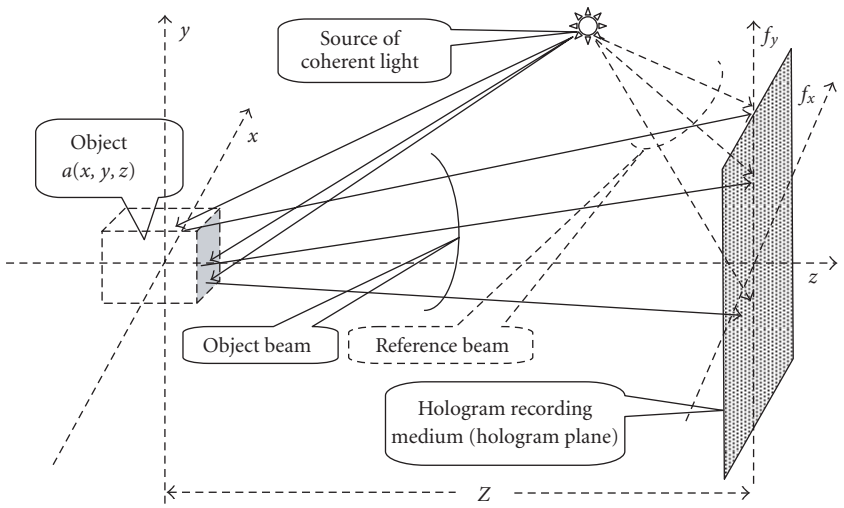


FIGURE 3.1. Schematic diagram of hologram recording.

3.2. Preliminaries

3.2.1. Mathematical models of recording and reconstruction of holograms

Consider schematic diagram of recording hologram shown in Figure 3.1. In hologram recording, an object whose hologram is to be recorded is illuminated by a coherent radiation from a radiation source, which simultaneously illuminates also a hologram recording medium with a "reference" beam. Usually, the photosensitive surface of the recording medium is a plane. We refer to this plane as a hologram plane. At each point of the hologram plane, the recording medium, whether it is continuous, such as a photographic film, or discrete, such as a photosensitive array of CMOS or CCD digital cameras, measures energy of the sum of the reference beam and the object beam reflected or transmitted by the object.

Conventional mathematical models of recording and reconstruction of holograms assume that (i) monochromatic coherent radiation that can be described by its complex amplitude as a function of spatial coordinates is used for hologram recording and reconstruction and (ii) object characteristics defining its ability to reflect or transmit incident radiation are described by radiation reflection or transmission factors which are also functions of spatial coordinates. Specifically, if $I(x, y, z)$ is complex amplitude of the object illumination radiation at point (x, y, z), complex amplitude $a(x, y, z)$ of the radiation reflected or transmitted by the object at this point is defined by the equation

$$a(x, y, z) = I(x, y, z)O(x, y, z), \tag{3.1}$$

where $O(x, y, z)$ is object reflection or, correspondingly, transmission factor.

If $\alpha(f_x, f_y)$ and $R(f_x, f_y)$ denote complex amplitudes of the object and reference beams, respectively, at point (f_x, f_y) of the hologram plane, signal recorded by the recording medium at this point is a squared module of their sum:

$$\begin{aligned} H(f_x, f_y) &= |\alpha(f_x, f_y) + R(f_x, f_y)|^2 \\ &= \alpha(f_x, f_y)R^*(f_x, f_y) + \alpha^*(f_x, f_y)R(f_x, f_y) \\ &\quad + |\alpha(f_x, f_y)|^2 + |R(f_x, f_y)|^2, \end{aligned} \tag{3.2}$$

where asterisk denotes complex conjugate. This signal is a hologram signal, or a hologram. The first term in the sum in the right-hand part of (3.2) is proportional to the object's beam complex amplitude. We will call this term a "*mathematical hologram*." Hologram reconstruction consists in applying to the mathematical hologram a transform that implements wave back propagation from the hologram plane to object. For this, one has either to eliminate, before the reconstruction, the other three terms or to apply the reconstruction transform to the entire hologram and then, separate the contribution of other terms in the reconstruction result from that of the mathematical hologram term.

The latter solution is known as a classical method of off-axis recording hologram (see [7]). In off-axis recording, a spatial angle between reference and object beams is introduced that exceeds the angular size of the object under which it is seen from the hologram plane. A drawback of this method is that it requires at least twice as much degrees of freedom (resolution cells) of the recording medium compared to that required for recording hologram without those interference terms.

The method of eliminating interfering terms in recorded holograms before the reconstruction is known as phase-shifting holography (see [5]). This method assumes several exposures of holograms of the object with shifting, for each exposure, the phase of the reference beam plane wave front that has the same propagation direction as the object wave front. With an appropriate selection of the phase, one can than determine the mathematical hologram term by solving system of (3.2) for different reference wave front phase shifts. One can show (see Appendix A) that at least three exposures with phase shifts $2\pi/3$ and $4\pi/3$ with respect to the first one are required in this method. Although the method can, in principle, be implemented optically for optical reconstruction of holograms, it was suggested and is used for numerical reconstruction of holograms.

As it is known, wave propagation transformations are, from the signal theory point of view, linear transformations. As such, they are mathematically modeled as integral transformation. Thus, object complex amplitude of radiation $a(x, y, z)$ and object beam wave front $\alpha(f_x, f_y)$ at hologram plane are related through "forward propagation"

$$\alpha(f_x, f_y) = \iiint_{-\infty}^{\infty} a(x, y, z)\,\mathrm{WPK}\,(x, y, z; f_x, f_y)\,dx\,dy\,dz \tag{3.3}$$

and "backward propagation"

$$a(x,y,z) = \iint_{-\infty}^{\infty} \alpha(f_x,f_y)\,\mathrm{WPK}\,(x,y,-z;f_x,f_y)\,df_x\,df_y \tag{3.4}$$

integral transforms, where $\mathrm{WPK}(\cdot;\cdot)$ is a transform (wave propagation) kernel. Commonly, this model is simplified by considering wave propagation between plane slices of the object and the hologram plane:

$$\begin{aligned}\alpha(f_x,f_y) &= \iint_{-\infty}^{\infty} a(x,y,0)\,\mathrm{WPK}\,(x,y,Z;f_x,f_y)\,dx\,dy\,dz,\\ a(x,y,0) &= \iint_{-\infty}^{\infty} \alpha(f_x,f_y)\,\mathrm{WPK}\,(x,y,-Z;f_x,f_y)\,df_x\,df_y,\end{aligned} \tag{3.5}$$

where Z is the distance between the object and hologram planes. Equations (3.5) are called diffraction transforms.

3.2.2. Diffraction integrals

In numerical reconstruction of holograms, diffraction transforms defined in the scalar diffraction theory of light propagation are usually considered and numerically implemented. According to the theory, wave front $\alpha(\mathbf{f})$ of an object $a(\mathbf{x})$ is defined by the Kirchhoff-Rayleigh-Sommerfeld integral (see [8]):

$$\alpha(\mathbf{f}) = \int_{-\infty}^{\infty} a(\mathbf{x})\frac{Z}{R}\left(1 - i\frac{2\pi}{\lambda}R\right)\frac{\exp(i2\pi R/\lambda)}{R^2}d\mathbf{x}, \tag{3.6}$$

where $\mathbf{x} = (x,y)$ is a coordinate vector in the object plane, $\mathbf{f} = (f_x,f_y)$ is a coordinate vector in the hologram plane, Z is a distance between object and hologram planes, $R = \sqrt{Z^2 + \|\mathbf{x}-\mathbf{f}\|^2}$, $\|\cdot\|$ symbolizes vector norm, and λ is the radiation wavelength.

Usually, an approximation to the integral (3.6) is made

$$\left(1 - i\frac{2\pi}{\lambda}R\right) \approx i\frac{2\pi}{\lambda}R, \tag{3.7}$$

and the above integral is reduced to

$$\begin{aligned}\alpha(f) &\approx -i\frac{2\pi}{\lambda}Z\int_{-\infty}^{\infty} a(x)\frac{\exp(i2\pi R/\lambda)}{R^2}dx\\ &\propto \int_{-\infty}^{\infty} a(x)\frac{\exp\left(i2\pi\left(Z\sqrt{(1+(x-f)^2/Z^2)}/\lambda\right)\right)}{1+(x-f)^2/Z^2}dx.\end{aligned} \tag{3.8}$$

This transform is called *Kirchhoff-Rayleigh-Sommerfeld integral transform.* In (3.8) and throughout the rest of the paper we, for the sake of brevity, use one-dimensional denotation unless otherwise is indicated.

Most frequently, object size and hologram size are small with respect to the object-to-hologram distance Z. Depending on how small they are, two other approximations to the Kirchhoff-Rayleigh-Sommerfeld integral equation (3.8) are used, "near-zone diffraction" (Fresnel) approximation:

$$\alpha(f) \propto \int_{-\infty}^{\infty} a(x) \exp\left[i\pi \frac{(x-f)^2}{\lambda Z}\right] dx \tag{3.9}$$

known as *Fresnel integral transform*, and "far-zone diffraction" (Fraunhofer) approximation which is well-known *Fourier integral transform*:

$$\alpha(f) \propto \int_{-\infty}^{\infty} a(x) \exp\left(-i2\pi \frac{xf}{\lambda Z}\right) dx. \tag{3.10}$$

There is also a version of the Fresnel transform called *angular spectrum propagation* (see [8]). In this version, Fresnel transform is treated as a convolution. By the Fourier convolution theorem, Fresnel transform (3.9) can be represented as inverse Fourier transform:

$$\begin{aligned}
\alpha(f) \propto & \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} a(x) \exp(i2\pi x\xi) dx \right] \\
& \times \left[\int_{-\infty}^{\infty} \exp\left(i\pi \frac{x^2}{\lambda Z}\right) \exp(i2\pi x\xi) dx \right] \exp(-i2\pi f\xi) d\xi \\
= & \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} a(x) \exp(i2\pi x\xi) dx \right] \\
& \times \left[\int_{-\infty}^{\infty} \exp\left(i\pi \frac{x^2}{\lambda Z}\right) \exp(i2\pi x\xi) dx \right] \exp(-i2\pi f\xi) d\xi \\
= & \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} a(x) \exp(i2\pi x\xi) dx \right] \exp\left(-i\pi\lambda Z\xi^2\right) \exp(-i2\pi f\xi) d\xi
\end{aligned} \tag{3.11}$$

of the product of signal Fourier transform and Fourier transform of the chirp function $\exp(i\pi x^2/\lambda Z)$, which is, in its turn, a chirp-function (Appendix B, (B.1)).

For numerical reconstruction of holograms recorded in different diffraction zones, as well as for synthesis of computer-generated holograms, discrete representations of the above diffraction integrals are needed that are suited for efficient computational implementation.

3.3. Discrete representation of transforms: principles

As far as quantum effects are not concerned, physical reality is a continuum whereas computers have only finite number of states. How can one imitate physical reality of optical signals and transforms in computers? Two principles lie in the base of digital representation of continuous signal transformations: the *conformity principle* with digital representation of signals and the *mutual correspondence* principle between continuous and discrete transformations (see [9]). The conformity principle requires that digital representation of signal transformations should parallel that of signals. Mutual correspondence between continuous and digital transformations is said to hold if both act to transform identical input signals into identical output signals. According to these principles, digital processors incorporated into optical information systems should be regarded and treated together with signal digitization and signal reconstruction devices as integrated analogous units.

Discretization is a process of measuring, in the discretization devices such as image scanners and hologram sensors, of coefficients of signal expansion into a series over a set of functions called *discretization basis functions*. These coefficients represent signals in computers. It is assumed that original continuous signals can be, with certain agreed accuracy, reconstructed by summation of functions called *reconstruction basis functions* with weights equal to the corresponding coefficients of signal discrete representation. The reconstruction is carried out in signal reconstruction devices such as, for instance, image displays and computer-generated hologram recorders.

In digital holography and image processing, discretization and reconstruction basis functions that belong to a family of "*shift,*" or sampling, *basis functions* are most commonly used. All functions from this family are obtained from one "mother" function by means of its spatial shifts through multiple of a *sampling interval*. Signal discrete representation coefficients obtained for such functions are called *signal samples*.

The mathematical formulation of signal sampling and reconstruction from the sampled representation is as follows. Let $a(\mathbf{x})$ be a continuous signal as a function of spatial coordinates given by a vector $\mathbf{x}^{(s)}$, let

$$\varphi_k^{(s)}\left(\mathbf{x}^{(s)}\right) = \varphi^{(s)}\left(\mathbf{x}^{(s)} - \widetilde{\mathbf{k}}^{(s)}\Delta\mathbf{x}^{(s)}\right), \qquad \varphi_k^{(r)}\left(\mathbf{x}^{(r)}\right) = \varphi^{(r)}\left(\mathbf{x}^{(r)} - \widetilde{\mathbf{k}}^{(r)}\Delta\mathbf{x}^{(r)}\right) \tag{3.12}$$

be sampling and reconstruction basis functions defined in the sampling and reconstruction devices coordinates $\{\mathbf{x}^{(s)}\}$ and $\{\mathbf{x}^{(r)}\}$, respectively, let $\Delta\mathbf{x}^{(s)}$ and $\Delta x^{(r)}$ be vectors of the sampling intervals in, respectively, sampling and reconstruction devices, let $\widetilde{\mathbf{k}}^{(s)} = \mathbf{k} + \mathbf{u}^{(s)}$ and $\widetilde{\mathbf{k}}^{(r)} = \mathbf{k} + \mathbf{u}^{(r)}$ be vectors of signal sample indices $\mathbf{k}$ biased by certain shift vectors $\mathbf{u}^{(s)}$ and $\mathbf{u}^{(r)}$. Shift vectors describe shift, in units of the corresponding sampling intervals, of the sampling grid with respect to the corresponding coordinate systems $\mathbf{x}^{(s)}$ and $\mathbf{x}^{(r)}$ such that samples with index $\mathbf{k} = 0$ are assumed to have respective coordinates $\mathbf{x}^{(s)} = \mathbf{u}^{(s)}$, $\mathbf{x}^{(r)} = \mathbf{u}^{(r)}$.

At sampling, signal samples $\{a_{\mathbf{k}}\}$ are computed as projections of the signal onto sampling basis functions:

$$a_k = \int_{\mathbf{X}} a(\mathbf{x})\varphi^{(s)}\left(\mathbf{x}^{(s)} - \widetilde{\mathbf{k}}^{(s)}\Delta\mathbf{x}^{(s)}\right)d\mathbf{x}, \tag{3.13}$$

assuming certain relationship between signal and sampling device coordinate systems $\{\mathbf{x}\}$ and $\{\mathbf{x}^{(s)}\}$.

Signal reconstruction from the set of their samples $\{a_{\mathbf{k}}\}$ is described as signal expansion over the set of reconstruction basis functions:

$$\widetilde{a}\left(\mathbf{x}^{(r)}\right) = \sum_{\mathbf{k}} a_{\mathbf{k}}\varphi^{(r)}\left(\mathbf{x}^{(r)} - \widetilde{\mathbf{k}}^{(r)}\Delta\mathbf{x}^{(r)}\right), \tag{3.14}$$

in a reconstruction device coordinate system.

The result $\widetilde{a}(\mathbf{x})$ of the signal reconstruction from its discrete representation obtained according to (3.14) is not, in general, identical to the initial signal $a(\mathbf{x})$. It is understood, however, that it can, in the given application, serve as a substitute for the initial signal.

Equations (3.13) and (3.14) model processes of hologram sampling in digital cameras and, respectively, object reconstruction in display devices.

According to the above-formulated conformity principle, (3.13) and (3.14) form the base for adequate discrete representation of signal transformations. In what follows, we describe discrete representations of optical diffraction integral transforms generated by this kind of signal representation.

3.4. Discrete Fourier transforms

In this section, we will provide, using the above outlined approach, a full derivation of discrete Fourier transforms as discrete representations of the integral Fourier transform. This derivation will explicitly demonstrate approximations that are done in this continuous-to-discrete conversion and will serve as an illustrative model for other diffraction transforms considered hereafter.

Let $\alpha(f)$ and $a(\mathbf{x})$ be, correspondingly, hologram and object wave fronts linked through the integral Fourier transform:

$$\alpha(f) = \int_{-\infty}^{\infty} a(x)\exp\left(i2\pi\frac{xf}{\lambda Z}\right)dx. \tag{3.15}$$

In digital recording of holograms, samples of the hologram wave front are obtained as

$$\alpha_r = \int_{-\infty}^{\infty} \alpha(f)\varphi^{(s)}\left(f - \widetilde{r}^{(s)}\Delta f^{(s)}\right)df. \tag{3.16}$$

Replacing here $\alpha(f)$ through its representation as Fourier transform of $a(\mathbf{x})$ (3.15), we obtain

$$\begin{aligned}
\alpha_r &= \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\infty} a(x) \exp\left(i2\pi \frac{xf}{\lambda Z}\right) dx \right\} \varphi^{(s)}\left(f - \tilde{r}^{(s)} \Delta f^{(s)}\right) df \\
&= \int_{-\infty}^{\infty} a(x) dx \left\{ \int_{-\infty}^{\infty} \varphi^{(s)}\left[f - \tilde{r}^{(s)} \Delta f^{(s)}\right] \exp\left(i2\pi \frac{xf}{\lambda Z}\right) \right\} df \\
&= \int_{-\infty}^{\infty} a(x) \exp\left(i2\pi \frac{x\tilde{r}^{(s)} \Delta f^{(s)}}{\lambda Z}\right) dx \left\{ \int_{-\infty}^{\infty} \varphi^{(s)}(f) \exp\left(i2\pi \frac{xf}{\lambda Z}\right) \right\} df \\
&= \int_{-\infty}^{\infty} a(x) \exp\left(i2\pi \frac{x\tilde{r}^{(s)} \Delta f^{(s)}}{\lambda Z}\right) \Phi^{(s)}(x) dx,
\end{aligned} \tag{3.17}$$

where

$$\Phi^{(s)}(x) = \int_{-\infty}^{\infty} \varphi^{(s)}(f) \exp\left(i2\pi \frac{xf}{\lambda Z}\right) df \tag{3.18}$$

is Fourier transform of the sampling device point spread function, or its "frequency response," $\{\tilde{r}^{(s)} = r + v^{(s)}\}$ are sample indices shifted as it was explained in Section 3.1. Now we can replace the object wave front $a(\mathbf{x})$ with its representation

$$a(x) = \sum_{k=0}^{N-1} a_k \varphi^{(r)}\left(x - \tilde{k}^{(r)} \Delta x^{(r)}\right) \tag{3.19}$$

through its samples $\{a_k\}$, assuming that N its samples are available, and that they are indexed from $k = 0$ to $k = N - 1$ with $\{\tilde{k} = k^{(r)} + u^{(r)}\}$ as correspondingly shifted indices. Using in (3.17) the representation of (3.19), we obtain

$$\begin{aligned}
\alpha_r &= \int_{-\infty}^{\infty} a(x) \exp\left(i2\pi \frac{x\tilde{r}^{(s)} \Delta f^{(s)}}{\lambda Z}\right) dx\, \Phi^{(s)}(x) \\
&= \int_{-\infty}^{\infty} \left\{ \sum_{k=0}^{N-1} a_k \varphi^{(r)}\left(x - \tilde{k}^{(r)} \Delta x^{(r)}\right) \right\} \exp\left(i2\pi \frac{x\tilde{r}^{(s)} \Delta f^{(s)}}{\lambda Z}\right) dx\, \Phi^{(s)}(x) \\
&= \left\{ \sum_{k=0}^{N-1} a_k \exp\left(i2\pi f \tilde{k}^{(r)} \tilde{r}^{(s)} \frac{\Delta x^{(r)} \Delta f^{(s)}}{\lambda Z}\right) \right\} \\
&\quad \times \left\{ \int_{-\infty}^{\infty} \varphi^{(r)}(x) \Phi^{(s)}\left(x + \tilde{k}^{(s)} \Delta x^{(r)}\right) \exp\left(i2\pi \frac{x\tilde{r}^{(s)} \Delta f^{(s)}}{\lambda Z}\right) dx \right\}.
\end{aligned} \tag{3.20}$$

As the discrete representation of the integral Fourier transform, only the first multiplier in the product in (3.20) is used

$$\alpha_r = \sum_{k=0}^{N-1} a_k \exp\left(i2\pi f \tilde{k}^{(r)} \tilde{r}^{(s)} \frac{\Delta x^{(r)} \Delta f^{(s)}}{\lambda Z}\right). \tag{3.21}$$

The second multiplier that depends on physical parameters of sampling and reconstruction (display) devices, on sampling intervals, and on object-to-hologram distance is ignored. It is this term that, in addition to the approximative signal reconstruction from the final number of its samples as described by (3.19), reflects approximative nature of the discrete representation of integral Fourier transform. The most straightforward way to quantitatively evaluate implication of the approximation is to consider the point spread function and the resolving power of signal Fourier spectrum analysis or of numerical reconstruction of Fourier holograms that can be achieved using discrete Fourier transforms. This issue is addressed in Section 3.7.2.

Sampling intervals in signal and Fourier domains $\Delta x^{(r)}$ and $\Delta f^{(s)}$ are known to be linked with the "uncertainty relationship":

$$\Delta x^{(r)} \geq \frac{\lambda Z}{N \Delta f^{(s)}}. \tag{3.22a}$$

The case

$$\Delta x^{(r)} = \frac{\lambda Z}{N \Delta f^{(s)}} \tag{3.22b}$$

associated with the assumption of the "band-limitedness" of signals is referred to as *cardinal sampling*. Depending on relationships between sampling intervals $\Delta x^{(r)}$and $\Delta f^{(s)}$and on shift parameters $u^{(s)}$ and $u^{(r)}$, the following modifications of the discrete Fourier transforms outlined in Sections 3.4.1–3.4.5 can be obtained.

3.4.1. 1D direct and inverse canonical discrete Fourier transforms

In the assumption that signal and its Fourier spectrum sampling intervals $\Delta x^{(r)}$ and $\Delta f^{(s)}$ satisfy the "cardinal" sampling relationship of (3.22b) and that object signal and object sampling device coordinate systems as well as those of the hologram and of the hologram sampling device are, correspondingly, identical, and object signal samples $\{a_k\}$ as well as samples $\{\alpha_r\}$ of its Fourier hologram are positioned in such a way that samples with indices $k = 0$ and $r = 0$ are taken in signal and its Fourier hologram coordinates in points $x = 0$ and $f = 0$, respectively, 1D direct

$$\alpha_r = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \exp\left(i2\pi \frac{kr}{N}\right) \tag{3.23a}$$

and inverse

$$a_k = \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \alpha_r \exp\left(- i2\pi \frac{kr}{N} \right). \tag{3.23b}$$

Canonical discrete Fourier transforms (DFTs) are obtained.

Canonical DFT plays a fundamental role in digital signal processing and, in particular, in digital holography and digital image processing thanks to the existence of fast Fourier transform (FFT) algorithms. With the use of FFT, computational complexity of transforms is as small as $O(\log N)$ operations per sample. All discrete transforms reviewed in this chapter are reducible to DFT and can be computed using FFT algorithms. Thanks to the existence of FFT algorithms, DFT is also very frequently used for fast computation of signal cyclic convolution through the following algorithm:

$$\sum_{n=0}^{N-1} a_n b_{(k-n) \bmod N} = \mathrm{IFFT}_N \left\{ \mathrm{FFT}_N \left(\{a_n\} \right) \bullet \mathrm{FFT}_N \left(\{b_n\} \right) \right\}, \tag{3.24}$$

where $\mathrm{FFT}_N\{\cdot\}$ and $\mathrm{IFFT}_N\{\cdot\}$ are operators of N-point FFTs applied to vectors of N signal samples and symbol $\bullet$ designates componentwise multiplication of the transform results.

3.4.2. 1D direct and inverse shifted DFTs: discrete cosine and cosine-sine transforms

If object and object signal sampling device coordinate systems as well as those of signal spectrum and the spectrum sampling device are laterally shifted such that signal sample $\{a_0\}$ and, correspondingly, sample $\{\alpha_0\}$ of its Fourier spectrum are taken in signal and spectrum coordinates at points $x = u^{(r)}\Delta x^{(r)}$ and $f = \nu^{(s)}\Delta f^{(s)}$, respectively, 1D direct

$$\alpha_r^{u,\nu} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \exp\left(i2\pi \frac{\widetilde{k}\widetilde{r}}{N} \right) \tag{3.25a}$$

and inverse

$$a_k^{u,\nu} = \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \alpha_r^{u,\nu} \exp\left(- i2\pi \frac{\widetilde{k}\widetilde{r}}{N} \right). \tag{3.25b}$$

Shifted DFTs (SDFTs(u,ν)) are obtained (see [9, 10]). The "cardinal" sampling relationship (3.22b) between object signal and its Fourier spectrum sampling intervals $\Delta x^{(r)}$ and $\Delta f^{(s)}$ is also assumed here.

SDFT can obviously be reduced to DFT and, therefore, be computed using FFT algorithms. The availability in SDFT of arbitrary shift parameters enables efficient algorithms for hologram reconstruction and object signal resampling with

discrete sinc-interpolation, which is, as it is shown in Chapter 8, the best possible interpolation method for sampled data. For instance, for **u**-shifted signal resampling using SDFT, the following relationship links initial signal samples $\{a_n\}$ and resampled ones $\{a_k^u\}$:

$$a_k^u = \sum_{n=0}^{N-1} a_n \frac{\sin\left[\pi(n-k+u)\right]}{N\sin\left[\pi(n-k+u)/N\right]} = \sum_{n=0}^{N-1} a_n \operatorname{sincd}\left[N, \pi(n-k+u)\right], \tag{3.26}$$

where

$$\operatorname{sincd}(N;x) = \frac{\sin x}{N\sin x/N} \tag{3.27}$$

is the discrete sinc-function (*sincd-function*).

Important special cases of shifted DFTs are *discrete cosine transform* (DCT):

$$\alpha_r = \sum_{k=0}^{N-1} a_k \cos\left(\pi\frac{k+1/2}{N}r\right) \tag{3.28a}$$

and *discrete cosine-sine transform* (DcST):

$$\alpha_r = \sum_{k=0}^{N-1} a_k \sin\left(\pi\frac{k+1/2}{N}r\right). \tag{3.28b}$$

They are SDFTs, with shift parameters 1/2 in the signal domain and 0 in the transform domain, of signals that exhibit even and, correspondingly, odd symmetry ($\{a_k = \pm a_{2N-1-k}\}$). DCT and DcST have fast computational algorithms that belong to the family of fast Fourier transform algorithms (see [9]).

DCT and DcST have numerous applications in image processing and digital holography. In particular, using fast DCT and DcST algorithms, one can efficiently, with the complexity of $O(\log N)$ operations per output sample, implement fast digital convolution with virtually no boundary effects. This allows to substantially alleviate severe boundary effects that are characteristic for using DFT for signal convolution as the DFT-based convolution algorithm (3.24) implements cyclic (periodic) convolution rather than usually required aperiodic convolution. This DCT-based algorithm for computing signal convolution is described in Appendix B. In conclusion of this section note that many other versions of SDFT, DCT, and DcST with semiinteger and integer shift parameters can be easily derived for different types of signal and its spectrum symmetries.

3.4.3. 1D scaled DFT

If one assumes that sampling rate of either signal samples or spectrum samples, or both is σ-times the "cardinal" sampling rate ($\Delta x^{(r)} = \lambda Z/\sigma N\Delta f^{(s)}$), and that signal and its Fourier hologram samples $\{a_0\}$ and $\{\alpha_0\}$ are positioned with shifts

$(u^{(r)}, v^{(s)})$ with respect to the origin of the corresponding signal and spectrum coordinate systems, *scaled DFT (ScDFT)*

$$\alpha_r^\sigma = \frac{1}{\sqrt{\sigma N}} \sum_{k=0}^{N-1} a_k \exp\left(i2\pi \frac{\widetilde{k}\widetilde{r}}{\sigma N}\right) \tag{3.29a}$$

is obtained. Modification of this transform for zero-shift parameters is also known under the names of "*chirp z-transform*" (see [11, 12]) and "*fractional discrete Fourier transform.*" The first name is associated with the way to compute it efficiently (see (3.30)). The second name assumes that it is a discretized version of the fractional integral Fourier transform (see [13, 14]) that has found some applications in optics and quantum mechanics. We prefer the name "scaled DFT" because it is more intuitive, refers to its physical interpretation, and fits the entire nomenclature of discrete Fourier transforms (shifted, scaled, rotated, scaled and rotated, affine) introduced in this chapter.

Scaled DFT has its inverse only if σN is an integer number ($\sigma N \in Z$). In this case, inverse ScDFT is defined as

$$a_k^\sigma = \frac{1}{\sqrt{\sigma N}} \sum_{r=0}^{\sigma N-1} \alpha_r \left(-i2\pi \frac{\widetilde{k}\widetilde{r}}{\sigma N}\right) = \begin{cases} a_k, & k = 0, 1, \ldots, N-1, \\ 0, & k = N, N+1, \ldots, \sigma N - 1. \end{cases} \tag{3.29b}$$

For computational purposes, it is convenient to express ScDFT as a cyclic convolution

$$\begin{aligned} \alpha_r^\sigma &= \frac{1}{\sqrt{\sigma N}} \sum_{k=0}^{N-1} a_k \exp\left(i2\pi \frac{\widetilde{k}\widetilde{r}}{\sigma N}\right) \\ &= \frac{\exp\left(i\pi(\widetilde{r}^2/\sigma N)\right)}{\sqrt{\sigma N}} \sum_{k=0}^{N-1} \left[a_k \exp\left(i\pi \frac{\widetilde{k}^2}{\sigma N}\right)\right] \exp\left[-i\pi \frac{(\widetilde{k}-\widetilde{r})^2}{\sigma N}\right], \end{aligned} \tag{3.30}$$

that can be computed using FFT algorithm as follows:

$$\begin{aligned} \alpha_r^{(\sigma)} = \mathrm{IFFT}_{\lfloor \sigma N \rfloor} \Bigg\{ & ZP_{\lfloor \sigma N \rfloor} \left[\mathrm{FFT}_N \left\{ a_k \exp\left(i\pi \frac{\widetilde{k}^2}{\sigma N}\right) \right\} \right] \\ & \bullet \mathrm{FFT}_{\lfloor \sigma N \rfloor} \left\{ \exp\left(-i\pi \frac{\widetilde{n}^2}{\sigma N}\right) \right\} \Bigg\}, \end{aligned} \tag{3.31}$$

where $\mathrm{FFT}_M\{\cdot\}$ and $\mathrm{IFFT}_M\{\cdot\}$ denote M-point direct and inverse FFTs, $\lfloor \sigma N \rfloor$ is an integer number defined by the inequality $\sigma N \le \lfloor \sigma N \rfloor < \sigma N + 1$, and $ZP_M[\cdot]$ is a zero-padding operator. If $\sigma > 1$, it pads the array of N samples with zeros to the array of $\lfloor \sigma N \rfloor$ samples. If $\sigma < 1$, it cuts array of Nsamples to size of $\lfloor \sigma N \rfloor$ samples.

The availability in ScDFT of the arbitrary scale parameter enables signal discrete sinc-interpolated resampling and Fourier hologram reconstruction in an

arbitrary scale. For instance, if one computes canonical DFT of a signal and then applies to the obtained spectrum scaled IDFT with a scale parameter σ, discrete sinc-interpolated samples of the initial signal in a scaled coordinate system will be obtained:

$$a_k^{\sigma} = \sum_{n=0}^{N-1} a_n \operatorname{sincd}\left[N; \pi\left(n - \frac{k}{\sigma}\right)\right]. \tag{3.32}$$

If $\sigma > 1$, signal $\{a_k\}$ is subsampled (its discrete representation is zoomed in) with discrete sinc-interpolation and the subsampled signal retains the initial signal bandwidth. If $\sigma < 1$, signal $\{a_k\}$ is downsampled (decimated). For signal downsampling, an appropriate signal low pass filtering required to avoid aliasing artifacts is automatically carried out by the imbedded zero-padding operator $ZP_{\lfloor \sigma N \rfloor}$.

3.4.4. 2D canonical separable DFTs

For 2D integral Fourier transform, the following separable 2D canonical direct and inverse DFTs:

$$\begin{aligned} \alpha_{r,s} &= \frac{1}{\sqrt{N_1 N_2}} \sum_{k=0}^{N_1-1} \exp\left(i2\pi \frac{kr}{N_1}\right) \sum_{l=0}^{N_2-1} a_{k,l} \exp\left(i2\pi \frac{ls}{N_1}\right), \\ a_{k,l} &= \frac{1}{\sqrt{N_1 N_2}} \sum_{r=0}^{N_1-1} \exp\left(-i2\pi \frac{kr}{N_1}\right) \sum_{s=0}^{N_2-1} \alpha_{r,s} \exp\left(-i2\pi \frac{ls}{N_2}\right) \end{aligned} \tag{3.33}$$

are obtained in the assumption that object signal and its hologram sampling and reconstruction are performed in rectangular sampling grids (rowwise, columnwise) collinear with the object coordinate system. Here N_1 and N_2 are dimensions of 2D arrays of signal and its Fourier spectrum samples. In separable 2D DFTs, 1D shifted and scaled DFTs can also be used to enable signal 2D separable discrete sinc-interpolated resampling and rescaling.

3.4.5. 2D rotated and affine DFTs

In 2D case, a natural generalization of 1D shifted and scaled DFTs is 2D *affine DFT (AffDFT)*. AffDFT is obtained in the assumption that either signal or its spectrum sampling or reconstructions are carried out in affine transformed, with respect to signal/spectrum coordinate systems (x, y), coordinates $(\tilde{x}, \tilde{y})$:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}. \tag{3.34}$$

With $\sigma_A = \lambda Z / N_1 A \Delta\tilde{x} \Delta f_x$, $\sigma_B = \lambda Z / N_2 B \Delta\tilde{y} \Delta f_x$, $\sigma_C = \lambda Z / N_1 C \Delta\tilde{x} \Delta f_y$, $\sigma_D = \lambda Z / N_2 D \Delta\tilde{y} \Delta f_y$, where $\Delta\tilde{x}$, $\Delta\tilde{y}$, Δf_x, and Δf_y are object and its Fourier hologram

sampling intervals in object $(\widetilde{x}, \widetilde{y})$ and hologram (f_x, f_y) planes, AffDTF is defined as

$$\alpha_{r,s} = \sum_{k=0}^{N_1-1} \sum_{l=0}^{N_2-1} a_{k,l} \exp\left[i2\pi \left(\frac{\widetilde{r}\widetilde{k}}{\sigma_A N_1} + \frac{\widetilde{s}\widetilde{k}}{\sigma_C N_1} + \frac{\widetilde{r}\widetilde{l}}{\sigma_B N_2} + \frac{\widetilde{s}\widetilde{l}}{\sigma_D N_2} \right) \right], \tag{3.35}$$

where $\{\widetilde{k}, \widetilde{l}\}$ and $\{\widetilde{r}, \widetilde{s}\}$ are biased (shifted) sampling indices in object and hologram planes as it is explained in Section 3.3.

A special case of affine transforms is rotation. For rotation angle θ,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \widetilde{x} \\ \widetilde{y} \end{bmatrix}. \tag{3.36}$$

With $N_1 = N_2 = N$, $\Delta\widetilde{x} = \Delta\widetilde{y} = \Delta x$, $\Delta f_x = \Delta f_y = \Delta f$, and $\Delta x \Delta f = \lambda Z/N$ (cardinal sampling), 2D *rotated DFT (RotDFT)* is obtained as

$$\alpha^{\theta}_{r,s} = \frac{1}{\sigma N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{k,l} \exp\left[i2\pi \left(\frac{\widetilde{k}\cos\theta + \widetilde{l}\sin\theta}{N} \widetilde{r} - \frac{\widetilde{k}\sin\theta - \widetilde{l}\cos\theta}{N} \widetilde{s} \right) \right]. \tag{3.37}$$

An obvious generalization of RotDFT is *rotated and scaled RotDFT (RotScDFT)*:

$$\alpha^{\theta}_{r,s} = \frac{1}{\sigma N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{k,l} \exp\left[i2\pi \left(\frac{\widetilde{k}\cos\theta + \widetilde{l}\sin\theta}{\sigma N} \widetilde{r} - \frac{\widetilde{k}\sin\theta - \widetilde{l}\cos\theta}{\sigma N} \widetilde{s} \right) \right], \tag{3.38}$$

that assumes 2D signal sampling in θ-rotated and σ-scaled coordinate systems. Similar to ScDFT, RotScDFT can be reduced to 2D cyclic convolution using the following identities:

$$\begin{aligned} \alpha^{\theta}_{r,s} &= \frac{1}{\sigma N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \widetilde{a}_{k,l} \exp\left[i2\pi \left(\frac{\widetilde{k}\cos\theta + \widetilde{l}\sin\theta}{\sigma N} \widetilde{r} - \frac{\widetilde{k}\sin\theta - \widetilde{l}\cos\theta}{\sigma N} \widetilde{s} \right) \right] \\ &= \frac{1}{\sigma N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \widetilde{a}_{k,l} \exp\left[i2\pi \left(\frac{\widetilde{k}\widetilde{r} + \widetilde{l}\widetilde{s}}{\sigma N} \cos\theta + \frac{\widetilde{l}\widetilde{r} - \widetilde{k}\widetilde{s}}{\sigma N} \sin\theta \right) \right], \end{aligned} \tag{3.39}$$

$$2(\widetilde{k}\widetilde{r} + \widetilde{l}\widetilde{s}) = \widetilde{r}^2 + \widetilde{k}^2 - \widetilde{s}^2 - \widetilde{l}^2 - (\widetilde{r} - \widetilde{k})^2 + (\widetilde{s} + \widetilde{l})^2, \tag{3.40}$$

$$2(\widetilde{l}\widetilde{r} - \widetilde{k}\widetilde{s}) = 2\widetilde{k}\widetilde{l} - 2\widetilde{r}\widetilde{s} + 2(\widetilde{r} - \widetilde{k})(\widetilde{s} + \widetilde{l}). \tag{3.41}$$

With inserting (3.39) and (3.40) into (3.39), we obtain

$$\begin{aligned}
\alpha_{r,s}^{\theta} &= \frac{1}{\sigma N}\sum_{r=0}^{N-1}\sum_{s=0}^{N-1}\widetilde{a}_{k,l}\exp\left[i2\pi\left(\frac{\widetilde{k}\widetilde{r}+\widetilde{l}\widetilde{s}}{\sigma N}\cos\theta+\frac{\widetilde{l}\widetilde{r}-\widetilde{k}\widetilde{s}}{\sigma N}\sin\theta\right)\right] \\
&= \frac{1}{\sigma N_a}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\widetilde{a}_{k,l}\exp\left[i\pi\frac{\widetilde{r}^2+\widetilde{k}^2-\widetilde{s}^2-\widetilde{l}^2-(\widetilde{r}-\widetilde{k})^2+(\widetilde{s}+\widetilde{l})^2}{\sigma N}\cos\theta\right] \\
&\quad\times\exp\left[-i2\pi\frac{2\widetilde{k}\widetilde{l}-2\widetilde{r}\widetilde{s}+2(\widetilde{r}-\widetilde{k})(\widetilde{s}+\widetilde{l})}{\sigma N}\sin\theta\right] \\
&= \left\{\frac{1}{\sigma N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}(\alpha_{r,s}A_{r,s})\,\mathrm{ChF}(\widetilde{s}+\widetilde{l},\widetilde{r}-\widetilde{k})\right\}\exp\left[-i\pi\frac{(\widetilde{k}^2-\widetilde{l}^2)\cos\theta-2\widetilde{k}\widetilde{l}\sin\theta}{\sigma N}\right],
\end{aligned} \tag{3.42}$$

where

$$\begin{aligned}
\mathrm{ChF}(\widetilde{s}+\widetilde{l},\widetilde{r}-\widetilde{k}) &= \exp\left[i\pi\frac{(\widetilde{s}+\widetilde{l})^2\cos\theta-(\widetilde{r}-\widetilde{k})^2\cos\theta-2(\widetilde{r}-\widetilde{k})(\widetilde{s}+\widetilde{l})\sin\theta}{\sigma N}\right], \\
A_{r,s} &= \left\{\exp\left[-i\pi\frac{(\widetilde{r}^2-\widetilde{s})^2\cos\theta+2\widetilde{r}\widetilde{s}\sin\theta}{\sigma N}\right]\right\}.
\end{aligned} \tag{3.43}$$

The convolution defined by (3.42) can be efficiently computed using FFT as

$$\begin{aligned}
\{\widetilde{a}_{k,l}\} = \mathrm{IFFT}\,2_{\lfloor\sigma N\rfloor}\Big\{&\mathrm{FFT}\,2_{\lfloor\sigma N\rfloor}\big[ZP_{\lfloor N\sigma\rfloor}\big[\mathrm{FFT}\,2_N(a_{k,l})\big]\bullet A_{r,s}\big] \\
&\bullet\,\mathrm{FFT}\,2_{\lfloor\sigma N\rfloor}\big[\,\mathrm{ChF}(r,s)\big]\Big\},
\end{aligned} \tag{3.44}$$

where $\mathrm{FFT}\,2_{\lfloor N\sigma\rfloor}[\cdot]$ and $\mathrm{IFFT}\,2_{\lfloor N\sigma\rfloor}[\cdot]$ are operators of direct and inverse $\lfloor N\sigma\rfloor$-point 2D FFT, $\lfloor N\sigma\rfloor$ is the smallest integer larger than $N\sigma$, $ZP_{\lfloor N\sigma\rfloor}[\cdot]$ is a 2D zero-padding operator. For $\sigma > 1$, it pads array of $N\times N$ samples with zeros to the array of $\lfloor\sigma N\rfloor\times\lfloor\sigma N\rfloor$ samples with $\lfloor\sigma N\rfloor$ defined, as above, by the inequality $\sigma N \le \lfloor\sigma N\rfloor < \sigma N+1$. For $\sigma < 1$, the padding operator cuts array of $N\times N$ samples to the size of $\lfloor\sigma N\rfloor\times\lfloor\sigma N\rfloor$ samples.

3.5. Discrete Fresnel transforms

In this section, we outline discrete transforms and their fast algorithms that can be used for numerical evaluation of the "near-zone" (Fresnel) diffraction integral. Using above described principles of discretization of signal transforms one can obtain its following discrete representations.

3.5.1. Canonical discrete Fresnel transform

Similar to canonical DFT, direct and inverse *canonical discrete Fresnel transforms (CDFrT)*

$$\begin{aligned} \alpha_r &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \exp\left[i\pi \frac{(k\mu - r/\mu)^2}{N} \right], \\ a_k &= \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \alpha_r \exp\left[-i\pi \frac{(k\mu - r/\mu)^2}{N} \right] \end{aligned} \tag{3.45a}$$

are obtained as a discrete representation of the integral Fresnel transform (3.5) in the assumption of the cardinal sampling relationship $\Delta x^{(r)} = \lambda Z / N \Delta f^{(s)}$ between sampling intervals in signal and transform domains and of zero shifts of object and hologram sampling grids with respect to the corresponding coordinate systems. Parameter μ^2 in (3.45) is defined as

$$\mu^2 = \frac{\lambda Z}{N \Delta f^2}. \tag{3.45b}$$

It plays in DFrT a role of a distance (focusing) parameter.

CDFrT can easily be expressed via DFT:

$$\alpha_r = \frac{1}{\sqrt{N}} \left\{ \sum_{k=0}^{N-1} \left[a_k \exp\left(i\pi \frac{k^2 \mu^2}{N} \right) \right] \exp\left(-i2\pi \frac{kr}{N} \right) \right\} \exp\left(i\pi \frac{r^2}{\mu^2 N} \right) \tag{3.46}$$

and, therefore, can be computed using FFT algorithms. In numerical reconstruction of Fresnel holograms, this method for computing DFrT is known as the "*Fourier reconstruction algorithm*."

3.5.2. Shifted discrete Fresnel transforms

If one assumes cardinal sampling condition and arbitrary shift parameters in object and/or its hologram sampling, *shifted discrete Fresnel transforms (SDFrTs)*

$$\alpha_r^{(\mu,w)} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \exp\left[-i\pi \frac{(k\mu - r/\mu + w)^2}{N} \right] \tag{3.47}$$

are obtained similar to how it was done for the discrete representation of the Fourier integral transform (3.10). Parameter w in (3.47) is a joint shift parameter that unifies shifts $u^{(r)}$ and $v^{(s)}$ of sampling grids in object and hologram planes:

$$w = \frac{u^{(r)}}{\mu} - v^{(s)}\mu. \tag{3.48}$$

3.5.3. Focal plane invariant discrete Fresnel transform

Because shift parameter w in shifted DFrT is a combination of shifts in object and hologram planes, shift in object plane causes a corresponding shift in Fresnel hologram plane, which, however, depends, according to (3.48), on the focusing parameter μ. One can break this interdependence if, in the definition of the discrete representation of integral Fresnel transform, one imposes a symmetry condition $\alpha_r^{(\mu,w)} = \alpha_{N-r}^{(\mu,w)}$ for the transform $\alpha_r^{(\mu,w)} = \exp[-i\pi(r/\mu - w)^2/N]$ of a point source $\delta(k) = 0^k$, $k = 0, 1, \ldots, N-1$. This condition is satisfied when $w = N/2\mu$, and SDFrT for such a shift parameter takes the form

$$\alpha_r^{(\mu,N/2\mu)} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \exp\left\{-i\pi \frac{\left[k\mu - (r - N/2)/\mu\right]^2}{N}\right\}. \tag{3.49}$$

We refer to this discrete transform as to *focal plane invariant discrete Fresnel transform (FPIDFrT)*. In numerical reconstruction of holograms, position of the reconstructed object in the output sampling grid depends on the object-hologram distance when canonical DFrT is used. FPIDFrT defined by (3.48) allows keeping position of reconstructed objects invariant to the object-hologram distance (see [15]), which might be useful in applications. For instance, invariance of the reconstruction object position with respect to the distance parameter can ease automatic object focusing and usage of pruned FFT algorithms in reconstruction of a part of the field of view (see [9]).

3.5.4. Partial discrete shifted Fresnel transform

When, in hologram reconstruction, only intensity of the object wave front is required, direct and inverse *partial discrete Fresnel transforms* (PDFrTs),

$$\begin{aligned}
\overset{\frown}{\alpha}_r^{(\mu,w)} &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \exp\left(-i\pi \frac{k^2\mu^2}{N}\right) \exp\left[i2\pi \frac{k(r - w\mu)}{N}\right], \\
a_k^{(\mu,w)} &= \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \overset{\frown}{\alpha}_r^{(\mu,w)} \exp\left[-i2\pi \frac{(r - w\mu)k}{N}\right] \exp\left(i\pi \frac{k^2\mu^2}{N}\right),
\end{aligned} \tag{3.50}$$

can be used as discrete representations of the integral Fresnel transform. They are obtained by removing from (3.45) exponential phase terms that do not depend on signal sampling index k. As one can see from (3.50), direct and inverse PDFrTs are essentially versions of SDFT. Their focal plane invariant versions can be defined correspondingly.

3.5.5. Invertibility of discrete Fresnel transforms and frincd-function

For shifted shift and focusing parameters, discrete Fresnel transforms are invertible orthogonal transforms. If, however, one computes, for a discrete signal $\{a_k\}$, $k = 0, 1, \ldots, N-1$, direct SDFrT with parameters (μ_+, w_+) and then inverts it with inverse SDFrT with parameters (μ_-, w_-), the following result, different from $\{a_k\}$, will be obtained:

$$a_k^{(\mu^{\pm}, w^{\pm})} = \frac{\exp\left[-i\pi\left((k\mu_- + w_-)^2/N\right)\right]}{N} \times \sum_{n=0}^{N-1} a_n \exp\left[i\pi\frac{(n\mu_+ + w_+)^2}{N}\right] \text{frincd}\left(N; q; n-k+\overline{w}_{\pm} + \frac{qN}{2}\right), \tag{3.51}$$

where $q = 1/\mu_+^2 - 1/\mu_-^2$, $\overline{w}_{\pm} = \overline{w}_+/\mu_+ - \overline{w}_-/\mu_-$, and

$$\text{frincd}\,(N; q; x) = \frac{1}{N}\sum_{r=0}^{N-1} \exp\left(i\pi\frac{qr^2}{N}\right)\exp\left(-i2\pi\frac{xr}{N}\right). \tag{3.52a}$$

It is a function analogous to the sincd-function of the DFT (3.27) and identical to it when $q = 0$. We refer to this function as to *frincd-function*. In numerical reconstruction of holograms, frincd-function is the convolution kernel that links object and its "out of focus" reconstruction.

Focal plane invariant version of frincd-function is obtained as

$$\overline{\text{frincd}}\,(N; q; x) = \frac{1}{N}\sum_{r=0}^{N-1} \exp\left[i\pi\frac{qr(r-N)}{N}\right]\exp\left(-i2\pi\frac{xr}{N}\right). \tag{3.52b}$$

Figure 3.2 illustrates behavior of absolute values of function $\overline{\text{frincd}}\,(N; q; x)$ for different values of q in the range $0 \le q \le 1$. In Figure 3.3, absolute values of function $\sqrt{q}\,\overline{\text{frincd}}\,(N; q; x)$ for q in the range $0 \le q \le 2.5$ are shown as an image in coordinates (q, x) to demonstrate aliasing artifacts that appear $q > 1$.

As one can see from (3.52), frincd-function is DFT of a chirp-function. It is known that integral Fourier transform of a chirp-function is also a chirp-function. In Appendix C it is shown that, in distinction from the continuous case, frincd-function can only be approximated by a chirp-function (see [16]),

$$\text{frincd}\,(N; \pm q; x) \cong \sqrt{\frac{\pm i}{Nq}}\exp\left[\mp i\pi\frac{x^2}{qN}\right]\text{rect}\left[\frac{x}{q(N-1)}\right]. \tag{3.53a}$$

For $q = 1$ and integer x, it is reduced to an exact chirp-function

$$\text{frincd}\,(N; 1; x) = \sqrt{\frac{i}{N}}\exp\left(-i\pi\frac{x^2}{N}\right), \quad x \in Z. \tag{3.53b}$$

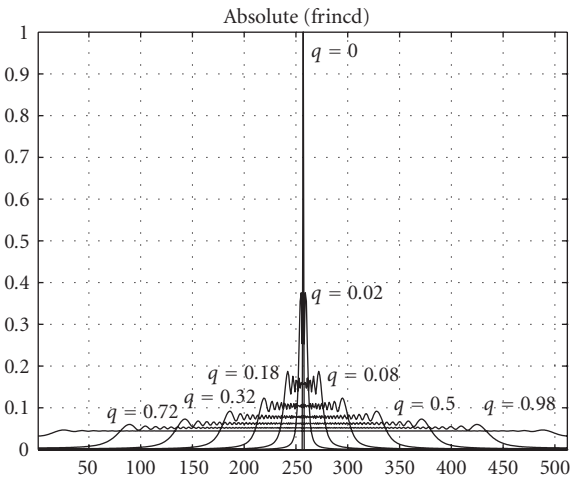


FIGURE 3.2. Absolute values of function $\overline{\text{frincd}}\,(N;q;x)$ for several values of q.

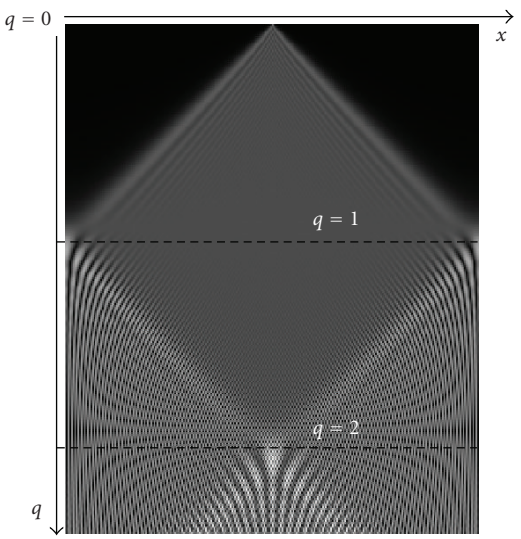


FIGURE 3.3. Function $\sqrt{q}\,\overline{\text{frincd}}\,(N;q;x)$ represented, in coordinates (q,x), as an image with lightness proportional to its absolute values of. Note aliasing artifacts for $q > 1$ that culminate when $q \geq 2$.

For $q = 0$, frincd-function reduces to sincd-function:

$$\text{frincd}\,(N;0;x) = \text{sincd}(N;\pi x)\exp\left(-i\pi\frac{N-1}{N}x\right). \tag{3.53c}$$

3.5.6. Convolutional discrete Fresnel transform

In numerical reconstruction of holograms, canonical discrete Fresnel transform and its above-described versions face aliasing problems for small distance Z, when focusing parameter $\mu^2 = \lambda Z/N\Delta f^2$ is becoming less than 1. For such cases, discrete representation of integral Fresnel transform can be used which is built on the base of the angular spectrum propagation version of the integral Fresnel transform (3.11):

$$\begin{aligned}\alpha(f) &\propto \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} a(x)\exp(i2\pi x\xi)dx\right]\exp\left(-i\pi\lambda Z\xi^2\right)\exp(-i2\pi f\xi)d\xi \\ &= \iint_{-\infty}^{\infty} a(x)\exp\left[i2\pi(x-f)\xi\right]\exp\left(-i\pi\lambda Z\xi^2\right)dx\,d\xi\end{aligned} \tag{3.54}$$

and in the deliberate assumption that sampling intervals $\Delta x^{(r)}$ and $\Delta f^{(s)}$ of the object signal and its Fresnel transform are identical; $\Delta x^{(r)} = \Delta f^{(s)}$. In this assumption, the following version of discrete Fresnel transform referred to as *convolutional discrete Fresnel transform (ConvDFrT)* is obtained for object and hologram sampling shift parameters $u^{(r)}$ and $v^{(s)}$:

$$\begin{aligned}\alpha_r &= \frac{1}{N}\sum_{s=0}^{N-1}\left[\sum_{k=0}^{N-1} a_k\exp\left(i2\pi\frac{k-r+w}{N}s\right)\right]\exp\left(-i\pi\frac{\mu^2s^2}{N}\right) \\ &= \frac{1}{N}\sum_{k=0}^{N-1} a_k\left[\sum_{s=0}^{N-1}\exp\left(-i\pi\frac{\mu^2s^2}{N}\right)\exp\left(i2\pi\frac{k-r+w}{N}s\right)\right],\end{aligned} \tag{3.55a}$$

or

$$\alpha_r = \sum_{k=0}^{N-1} a_k\ \mathrm{frincd}^{\,*}(N;\mu^2;k-r+w), \tag{3.55b}$$

where $w = u^{(r)} - v^{(s)}$ and asterisk denotes complex conjugate.

ConvDFrT, similar to DFTs and DFrTs, is an orthogonal transform with inverse *ConvDFrT* defined as

$$\begin{aligned}a_k &= \frac{1}{N}\sum_{s=0}^{N-1}\left[\sum_{r=0}^{N-1}\alpha_r\exp\left(-i2\pi\frac{k-r+w}{N}s\right)\right]\exp\left(i\pi\frac{\mu^2s^2}{N}\right) \\ &= \sum_{r=0}^{N-1}\alpha_r\ \mathrm{frincd}\,(N;\mu^2;k-r+w).\end{aligned} \tag{3.55c}$$

When $\mu^2 = 0$ and $w = 0$, ConvDFrT degenerates to the identical transform.

Although ConvDFrT as a discrete transform can be inverted for any μ^2, in numerical reconstruction of holograms it can be recommended only for $\mu^2 = \lambda Z/N\Delta f^2 \leq 1$. If $\mu^2 = \lambda Z/N\Delta f^2 > 1$, aliasing appears in the form of overlapping periodical copies of the reconstruction result.

The Fresnel hologram reconstruction algorithm based on ConvFrT is frequently referred to as the "*convolution reconstruction algorithm*".

3.5.7. 2D discrete Fresnel transforms: scaled and rotated transforms

2D discrete Fresnel transforms are usually defined as separable row-column transforms. For instance, canonical 2D Fresnel transform is defined as

$$\alpha_{r,s}^{(\mu,0,0)} = \frac{1}{\sqrt{N_1 N_2}} \sum_{k=0}^{N-1} \exp\left[-i\pi \frac{(k\mu - r/\mu)^2}{N} \right] \sum_{l=0}^{N_2-1} a_{k,l} \exp\left[-i\pi \frac{(l\mu - s/\mu)^2}{N_2} \right]. \tag{3.56}$$

As the last stage in the algorithmic implementations of all versions of discrete Fresnel transforms is discrete Fourier transform implemented via FFT algorithm, scaled and rotated modifications of DFT can be applied at this stage. This will enable scaling and/or rotation of the transform result, if they are required when using discrete Fresnel transforms for numerical reconstruction of digitally recorded holograms. For instance, scaled reconstruction is required in reconstruction of hologram of the same object recorded in different wavelengths, such as color holograms.

3.6. Discrete Kirchhoff-Rayleigh-Sommerfeld transforms

Using the above-described transform discretization principles and assuming, as in the case of the convolutional Fresnel transform, identical sampling intervals $\Delta x^{(r)}$ and $\Delta f^{(s)}$ of the signal and its transform, one can obtain the following 1D and 2D (for square data arrays) discrete representations of integral Kirchhoff-Payleigh-Sommerfeld transform:

$$\begin{aligned} \alpha_r &= \sum_{k=0}^{N-1} a_k \, \mathrm{KRS}^{(1\mathrm{D})}(\tilde{k} - \tilde{r}), \\ \alpha_{r,s} &= \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{k,l} \, \mathrm{KRT}^{(2\mathrm{D})}(\tilde{k} - \tilde{r}; \tilde{l} - \tilde{r}), \end{aligned} \tag{3.57}$$

where it is denoted as

$$\mathrm{KRS}^{(1\mathrm{D})}_{\widetilde{z},\mu}(n) = \frac{\exp\left[i2\pi\left(\widetilde{z}^2\sqrt{(1+n^2/\widetilde{z}^2)}/\mu^2 N\right)\right]}{1+n^2/\widetilde{z}^2},$$

$$\mathrm{KRS}^{(2\mathrm{D})}_{\widetilde{z},\mu}(m,n) = \frac{\exp\left[i2\pi\left(\widetilde{z}^2\sqrt{(1+m^2/\widetilde{z}^2+n^2/\widetilde{z}^2)}/\mu^2 N\right)\right]}{1+m^2/\widetilde{z}^2+n^2/\widetilde{z}^2}, \tag{3.58}$$

$$\widetilde{z} = \frac{Z}{\Delta f}, \qquad \mu^2 = \frac{\lambda Z}{N\Delta f^2} = \frac{\lambda\widetilde{z}}{N\Delta f}.$$

We refer to these transforms as 1D and 2D *discrete Kirchhoff-Rayleigh-Sommerfeld transforms (DKRSTs)*. When $\widetilde{z} \to 0$, DKRS-transform degenerates into the identical transform. When $\widetilde{z} \to \infty$, DKRS-transform converts to discrete Fresnel transforms. In distinction from 2D DFTs and DFrTs, 2D discrete KRS-transform is inseparable transform.

As one can see from (3.58), discrete Kirchhoff-Rayleigh-Sommerfeld transforms are digital convolutions. Therefore, they can be computed using FFT as

$$\{\alpha_{\mathbf{r}}\} = \mathrm{IFFT}\left\{\mathrm{FFT}\left[\{a_{\mathbf{k}}\}\right] \bullet \mathrm{FFT}\left[\mathrm{KRS}_{\widetilde{z},\mu}(\mathbf{n})\right]\right\}. \tag{3.59}$$

From this representation, inverse DKRS-transform can be obtained as

$$\{\widetilde{a}_{\mathbf{k}}\} = \mathrm{IFFT}\left\{\mathrm{FFT}\{\alpha_{\mathbf{r}}\} \bullet \frac{1}{\mathrm{FFT}\left[\mathrm{KRS}_{\widetilde{z},\mu}(\mathbf{n})\right]}\right\}. \tag{3.60}$$

3.7. Resolving power and point spread functions of numerical reconstruction of holograms

3.7.1. PSF of numerical reconstruction of holograms: a general formulation

In numerical reconstruction of holograms, samples of the object wave front are reconstructed out of samples of its hologram using above described discrete diffraction transforms. This process can be treated as object wave front sampling by a sampling system that consists of the hologram sampling device and a computer, in which the hologram is numerically reconstructed.

Signal sampling is a linear transformation that is fully specified by its point spread function (PSF) that establishes a link between an object signal $a(x)$ and its samples $\{a_k\}$:

$$a_k = \int_X a(x)\,\mathrm{PSF}(x,k)dx. \tag{3.61}$$

According to the sampling theory (see, e.g., [9]), for a given sampling interval Δx, PSF of the ideal sampling device is a sinc-function:

$$\mathrm{PSF}(x,k) = \sin c\left[\frac{\pi(x-k\Delta x)}{\Delta x}\right] = \frac{\sin\left[\pi(x-k\Delta x)/\Delta x\right]}{\pi(x-k\Delta x)/\Delta x}. \tag{3.62}$$

Provided that the continuous signal is reconstructed from its samples using, as reconstruction basis functions, the same sinc-functions, this PSF secures minimal root mean square signal reconstruction error.

In this section, we consider point spread functions of different reconstruction algorithms and how do they depend on algorithm parameters and physical parameters of holograms and their sampling devices. For the sake of simplicity, we will consider 1D holograms and transforms. Corresponding 2D results are straightforward in the conventional assumption of separability of sampling and transforms.

Let, in numerical reconstruction of holograms, samples $\{a_k\}$ of the object wave front be obtained through a transformation

$$a_k = \sum_{r=0}^{N-1} \alpha_r \,\mathrm{DRK}(r,k) \tag{3.63}$$

of available hologram samples $\{\alpha_r\}$ with a certain discrete reconstruction kernel $\mathrm{DRK}(r,k)$ that corresponds to the type of the hologram. Hologram samples $\{\alpha_r\}$ are measured by a hologram recording and sampling device as

$$\alpha_r = \int_{-\infty}^{\infty} \alpha(f)\varphi_f^{(s)}\left(f - \tilde{r}^{(s)}\Delta f^{(s)}\right)df, \tag{3.64}$$

where $\alpha(f)$ is a hologram signal, $\{\varphi_f^{(s)}(\cdot)\}$ is a point spread function of the hologram sampling device, $\Delta f^{(s)}$ is a hologram sampling interval, $\tilde{r}^{(s)} = r + \nu^{(s)}$, r is an integer index of hologram samples, and $\nu^{(s)}$ is a shift, in units of the hologram sampling interval, of the hologram sampling grid with respect to the hologram coordinate system.

The hologram signal $\alpha(f)$ is linked with object wave front $a(x)$ through a diffraction integral

$$\alpha(f) = \int_{-\infty}^{\infty} a(x)\,\mathrm{WPK}(x,f)dx, \tag{3.65}$$

where $\mathrm{WPK}(x,f)$ is a wave propagation kernel. Threrefore, one can rewrite (3.64) as

$$\begin{aligned}
\alpha_r &= \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} a(x)\,\mathrm{WPK}(x,f)dx\right]\varphi_f^{(s)}\left(f - \tilde{r}^{(s)}\Delta f^{(s)}\right)df \\
&= \iint_{-\infty}^{\infty} a(x)\,\mathrm{WPK}(x,f)\varphi_f^{(s)}\left(f - \tilde{r}^{(s)}\Delta f^{(s)}\right)dx\,df \\
&= \int_{-\infty}^{\infty} a(x)dx \int_{-\infty}^{\infty} \mathrm{WPK}(x,f)\varphi_f^{(s)}\left(f - \tilde{r}^{(s)}\Delta f^{(s)}\right)df.
\end{aligned} \tag{3.66}$$

Insert now (3.66) into (3.63) and establish a link between object wave front $a(x)$ and its samples $\{a_k\}$ reconstructed from the sampled hologram:

$$\begin{aligned} a_k &= \sum_{r=0}^{N-1} \left[\int_{-\infty}^{\infty} a(x)dx \int_{-\infty}^{\infty} \mathrm{WPK}(x,f)\varphi_f^{(s)}\left(f - \tilde{r}^{(s)}\Delta f^{(s)}\right)df \right] \mathrm{DRK}(r,k) \\ &= \int_{-\infty}^{\infty} a(x)dx \left[\int_{-\infty}^{\infty} \mathrm{WPK}(x,f)df \sum_{r=0}^{N-1} \mathrm{DRK}(r,k)\varphi_f^{(s)}\left(f - \tilde{r}^{(s)}\Delta f^{(s)}\right) \right] \\ &= \int_{-\infty}^{\infty} a(x)\,\mathrm{PSF}(x,k)dx, \end{aligned} \tag{3.67}$$

where function

$$\mathrm{PSF}(x,k) = \int_{-\infty}^{\infty} \mathrm{WPK}(x,f)df \sum_{r=0}^{N-1} \mathrm{DRK}(r,k)\varphi_f^{(s)}\left(f - \tilde{r}^{(s)}\Delta f^{(s)}\right) \tag{3.68}$$

can be treated as *a point spread function (PSF) of the numerical reconstruction of holograms*. As one can see from (3.68), it depends on all factors involved in the process of sampling and reconstruction of holograms: wave propagation kernel $\mathrm{WPK}(\cdot,\cdot)$, discrete reconstruction kernel $\mathrm{DRK}(\cdot,\cdot)$, and point spread function of the hologram sampling device $\varphi_f^{(s)}(\cdot)$.

For further analysis, it is convenient to replace point spread function of the hologram sampling device through its Fourier transform, or its frequency response $\Phi_f^{(s)}(\cdot)$:

$$\begin{aligned} \varphi_f^{(s)}\left(f - \tilde{r}^{(s)}\Delta f\right) &= \int_{-\infty}^{\infty} \Phi_f^{(s)}(\xi) \exp\left[i2\pi\left(f - \tilde{r}^{(s)}\Delta f^{(s)}\right)\xi\right]d\xi \\ &= \int_{-\infty}^{\infty} \Phi_f^{(s)}(\xi) \exp\left(-i2\pi\tilde{r}^{(s)}\Delta f^{(s)}\xi\right) \exp(i2\pi f\xi)d\xi. \end{aligned} \tag{3.69}$$

Then obtain

$$\begin{aligned} \mathrm{PSF}(x,k) &= \int_{-\infty}^{\infty} \mathrm{WPK}(x,f)df \sum_{r=0}^{N-1} \mathrm{DRK}(r,k) \\ &\quad \times \int_{-\infty}^{\infty} \Phi_f^{(s)}(\xi) \exp\left(-i2\pi\tilde{r}^{(s)}\Delta f^{(s)}\xi\right) \exp(i2\pi f\xi)d\xi \\ &= \int_{-\infty}^{\infty} \Phi_f^{(s)}(\xi)d\xi \int_{-\infty}^{\infty} \mathrm{WPK}(x,f) \exp(i2\pi f\xi)df \\ &\quad \times \sum_{r=0}^{N-1} \mathrm{DRK}(r,k) \exp\left(-i2\pi\tilde{r}^{(s)}\Delta f^{(s)}\xi\right). \end{aligned} \tag{3.70}$$

Introduce function

$$\begin{aligned}\overline{\mathrm{PSF}}(x,\xi;k) &= \int_{-\infty}^{\infty} \mathrm{WPK}(x,f)\exp(i2\pi f\xi)df \sum_{r=0}^{N-1} \mathrm{DRK}(r,k)\exp\left(-i2\pi\tilde{r}^{(s)}\Delta f^{(s)}\xi\right) \\ &= \overline{\mathrm{WPK}}(x,\xi)\cdot\overline{\mathrm{DRK}}(\xi,k),\end{aligned} \tag{3.71}$$

where

$$\overline{\mathrm{WPK}}(x,\xi) = \int_{-\infty}^{\infty} \mathrm{WPK}(x,f)\exp(i2\pi f\xi)df \tag{3.72}$$

is Fourier transform of the wave propagation kernel $\mathrm{WPK}(\cdot,\cdot)$ and

$$\overline{\mathrm{DRK}}(\xi,k) = \sum_{r=0}^{N-1} \mathrm{DRK}(r,k)\exp\left(-i2\pi r^{(s)}\Delta f^{(s)}\xi\right) \tag{3.73}$$

which is a Fourier series expansion with the discrete reconstruction kernel $\mathrm{DRK}(r,k)$ as expansion coefficients. Function $\overline{\mathrm{PSF}}(x,\xi;k)$ does not depend on parameters of the hologram sampling device. We will refer to this function as *PSF of sampled hologram reconstruction*. PSF of the numerical reconstruction of holograms $\mathrm{PSF}(x,k)$ and PSF of sampled hologram reconstruction $\overline{\mathrm{PSF}}(x,\xi;k)$ are linked through the integral transform

$$\mathrm{PSF}(x,k) = \int_{-\infty}^{\infty} \Phi_f^{(s)}(\xi)\overline{\mathrm{PSF}}(x,\xi;k)d\xi \tag{3.74}$$

with frequency response of the hologram sampling device as a transform kernel.

3.7.2. Point spread function of numerical reconstruction of holograms recorded in far diffraction zone (Fourier holograms)

For far diffraction zone (3.10), wave propagation kernel is

$$\mathrm{WPK}(x,f) = \exp\left(-i2\pi\frac{xf}{\lambda Z}\right). \tag{3.75}$$

Its Fourier transform is a delta function:

$$\begin{aligned}\overline{\mathrm{WPK}}(x,\xi) &= \int_{-\infty}^{\infty} \mathrm{WPK}(x,f)\exp(i2\pi f\xi)df \\ &= \int_{-\infty}^{\infty} \exp\left[-i2\pi f\left(\frac{x}{\lambda Z}-\xi\right)\right]df = \delta\left(\frac{x}{\lambda Z}-\xi\right).\end{aligned} \tag{3.76}$$

Assume that shifted DFT with discrete reconstruction kernel

$$\mathrm{DRK}(r,k) = \exp\left[i2\pi\frac{(k+u)(r+v)}{N}\right] \tag{3.77}$$

is used for that for numerical reconstruction of Fourier hologram. Fourier series expansion over this discrete reconstruction kernel is

$$\begin{aligned}
\overline{\mathrm{DRK}}(\xi,k) &= \sum_{r=0}^{N-1} \mathrm{DRK}(r,k)\exp\Big(-i2\pi\tilde{r}^{(s)}\Delta f^{(s)}\xi\Big) \\
&= \sum_{r=0}^{N-1} \exp\left[i2\pi\frac{(k+u)(r+v)}{N}\right]\exp\left[-i2\pi\Big(r+v^{(s)}\Big)\Delta f^{(s)}\xi\right] \\
&= \exp\Big(-i2\pi v^{(s)}\Delta f^{(s)}\xi\Big)\exp\left[i2\pi\frac{(k+u)v}{N}\right] \\
&\quad\times\sum_{r=0}^{N-1}\exp\left[i2\pi\Big(\frac{(k+u)}{N}-\Delta f^{(s)}\xi\Big)r\right] \\
&= \exp\Big(-i2\pi v^{(s)}\Delta f^{(s)}\xi\Big)\exp\left[i2\pi\frac{(k+u)v}{N}\right] \\
&\quad\times\frac{\exp\left[i2\pi N\Big((k+u)/N-\Delta f^{(s)}\xi\Big)\right]-1}{\exp\left[i2\pi\Big((k+u)/N-\Delta f^{(s)}\xi\Big)\right]-1} \\
&= \exp\left[-i2\pi\Big(v^{(s)}+\frac{N-1}{2}\Big)\Delta f^{(s)}\xi\right]\exp\left\{i2\pi\frac{(k+u)}{N}\Big(v+\frac{N-1}{2}\Big)\right\} \\
&\quad\times\frac{\sin\left[\pi N(\Delta f^{(s)}\xi-(k+u)/N)\right]}{\sin\left[\pi(\Delta f^{(s)}\xi-(k+u)/N)\right]}.
\end{aligned} \tag{3.78}$$

It is only natural now to choose shift parameters $v^{(s)}$ and v of sampling and reconstruction transform as

$$v^{(s)} = v = -\frac{N-1}{2}. \tag{3.79}$$

With these shift parameters,

$$\begin{aligned}
\overline{\mathrm{DRK}}(\xi,k) &= \frac{\sin\left[\pi(\Delta f^{(s)}\xi-(k+u)/N)N\right]}{\sin\left[\pi(\Delta f^{(s)}\xi-(k+u)/N)\right]} \\
&= N\,\mathrm{sincd}\left[N;\pi\Big(\Delta f^{(s)}\xi-\frac{(k+u)}{N}\Big)N\right],
\end{aligned} \tag{3.80}$$

$$\begin{aligned}
\overline{\mathrm{PSF}}(x,\xi;k) &= \overline{\mathrm{WPK}}(x,\xi)\cdot\overline{\mathrm{DRK}}(\xi,k) \\
&= N\,\mathrm{sincd}\left[N;\pi\Big(\Delta f^{(s)}\xi-\frac{(k+u)}{N}\Big)N\right]\delta\Big(\frac{x}{\lambda Z}-\xi\Big).
\end{aligned} \tag{3.81}$$

Then, finally, obtain that the point spread function of numerical reconstruction of Fourier hologram is

$$\begin{aligned}\mathrm{PSF}(x,k) &= \int_{-\infty}^{\infty} \Phi_f^{(s)}(\xi)\overline{\mathrm{PSF}}(x,\xi;k)d\xi \\ &= N\,\mathrm{sincd}\left[N;\pi\left(\frac{\Delta f^{(s)}x}{\lambda Z}-\frac{(k+u)}{N}\right)N\right]\int_{-\infty}^{\infty}\Phi_f^{(s)}(\xi)\delta\left(\frac{x}{\lambda Z}-\xi\right)d\xi \\ &= N\,\mathrm{sincd}\left[N;\pi\left(\frac{\Delta f^{(s)}x}{\lambda Z}-\frac{(k+u)}{N}\right)N\right]\Phi_f^{(s)}\left(\frac{x}{\lambda Z}\right) \\ &= N\,\mathrm{sincd}\left[N;\pi\frac{(x-(k+u)\Delta_x)}{\Delta_x}\right]\Phi_f^{(s)}\left(\frac{x}{\lambda Z}\right),\end{aligned} \tag{3.82}$$

where

$$\Delta_x = \frac{\lambda Z}{N\Delta f^{(s)}} = \frac{\lambda Z}{S_H} \tag{3.83}$$

and $S_H = N\Delta f^{(s)}$ is the physical size of the hologram.

Formula (3.82) has a clear physical interpretation illustrated in Figure 3.4. As it was mentioned, point spread function of the ideal signal sampling device is a sinc-function and its frequency response is a rect-function. Provided the hologram sampling device is such an ideal sampler with frequency response:

$$\Phi_f^{(s)}\left(\frac{x}{\lambda Z}\right) = \mathrm{rect}\left(\frac{x+\lambda Z/2\Delta f^{(s)}}{\Delta f^{(s)}/\lambda Z}\right) = \begin{cases}1, & -\dfrac{\lambda Z}{2\Delta f^{(s)}} \le x \le \dfrac{\lambda Z}{2\Delta f^{(s)}},\\ 0, & \text{otherwise},\end{cases} \tag{3.84}$$

point spread function of numerical reconstruction of Fourier hologram is a discrete sinc-function, and object wave front samples are measured within the object's spatial extent interval $-\lambda Z/2\Delta f^{(s)} \le x \le \lambda Z/2\Delta f^{(s)}$ defined by the spread of the hologram sampling device frequency response. Therefore, with an ideal hologram sampler, numerical reconstruction of Fourier hologram is almost ideal object wave front sampling, as discrete sinc-function approximates continuous sinc-function within the interval $-\lambda Z/2\Delta f^{(s)} \le x \le \lambda Z/2\Delta f^{(s)}$ relatively closely, the closer, the larger the number of hologram samples N. Parameter Δ_x defined by (3.83) is half width of the main lobe of the discrete sinc-function. It characterizes the virtual intersample distance and the resolving power of the reconstruction algorithm.

In reality, hologram sampling devices are, of course, not ideal samplers, and their frequency responses are not rectangular functions. They are not uniform within the basic object extent interval $-\lambda Z/2\Delta f^{(s)} \le x \le \lambda Z/2\Delta f^{(s)}$ and decay not abruptly outside this interval but rather gradually. As a consequence, each of the object samples is a combination of the sample measured by the main lobe of the discrete sinc-function within the basic object extent interval and samples collected

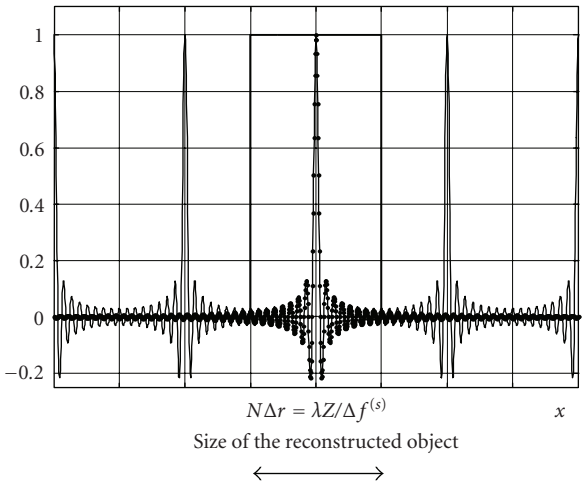


FIGURE 3.4. PSF of numerical reconstruction of holograms digitally recorded in far diffraction zone (thin solid line). Rectangle in bold line is Fourier transform of the hologram ideal sampling device. Bold dots plot sinc-function, the ideal sampling PSF.

by other lobes of the discrete sinc-function outside the basic interval. This is one source of the measurement errors. In particular, for diffusely reflecting objects, it may result in an additional speckle noise in the reconstructed object image. One can avoid this distortion if, in the process of making object hologram, object is illuminated strictly within the basic interval as defined by the hologram sampling interval (camera pitch).

The second source of the reconstruction errors is associated with nonuniformity of the hologram sampler frequency response within the basic interval. These errors can be compensated by multiplying the reconstruction results by the function inverse to the frequency response of the hologram sampler.

One can also see from (3.82) that the resolving power of numerical reconstruction of Fourier hologram is determined by the distance between zeros of the discrete sinc-function, which is equal to $\lambda Z/N\Delta f^{(s)} = \lambda Z/S_H$, where $S_H = N\Delta f^{(s)}$ is size of the sampled hologram. Due to this finite resolving power, one can also expect, for diffuse objects, a certain amount of speckle noise in the reconstructed object.

3.7.3. Point spread function of numerical reconstruction of holograms recorded in near diffraction zone (Fresnel holograms)

For near diffraction zone (3.9), wave propagation kernel is

$$\mathrm{WPK}(x, f) \propto \exp\left[i\pi \frac{(x-f)^2}{\lambda Z}\right]. \tag{3.85}$$

Its Fourier transform is

$$\begin{aligned}\overline{\mathrm{WPK}}(x,\xi) &\propto \int_{-\infty}^{\infty} \exp\left[i\pi\frac{(x-f)^2}{\lambda Z}\right]\exp(i2\pi f\xi)df \\ &= \exp(i2\pi x\xi)\int_{-\infty}^{\infty}\exp\left(i\pi\frac{f^2}{\lambda Z}\right)\exp(i2\pi f\xi)df,\end{aligned} \tag{3.86}$$

or, with an account of (B.1) (Appendix B),

$$\overline{\mathrm{WPK}}(x,\xi) \propto \exp(i2\pi x\xi)\exp\left(-i\pi\lambda Z\xi^2\right). \tag{3.87}$$

In what follows, we will separately consider point spread function for Fourier and convolution reconstruction algorithms. For simplicity, zero shifts will be assumed in both, hologram and object wave front domains.

3.7.3.1. Fourier reconstruction algorithm

In the Fourier reconstruction algorithm, discrete reconstruction kernel is, with zero shifts in hologram and object wave front domains,

$$\mathrm{DRK}(r,k) = \left[\exp\left(-i\pi\frac{k^2\mu^2}{N}\right)\exp\left(i2\pi\frac{kr}{N}\right)\right]\exp\left(-i\pi\frac{r^2}{\mu^2 N}\right). \tag{3.88}$$

Fourier series expansion over this kernel is

$$\begin{aligned}\overline{\mathrm{DRK}}(\xi,k) &= \exp\left(-i\pi\frac{k^2\mu^2}{N}\right)\sum_{r=0}^{N-1}\exp\left(-i\pi\frac{r^2}{\mu^2 N}\right) \\ &\quad\times\exp\left(i2\pi\frac{kr}{N}\right)\exp\left(-i2\pi r\Delta f^{(s)}\xi\right) \\ &= \exp\left(-i\pi\frac{k^2\mu^2}{N}\right)\sum_{r=0}^{N-1}\exp\left(-i\pi\frac{r^2}{\mu^2 N}\right)\exp\left[i2\pi r\left(\frac{k}{N}-\Delta f^{(s)}\xi\right)\right],\end{aligned} \tag{3.89a}$$

or, in a more compact form,

$$\overline{\mathrm{DRK}}(\xi,k) = \exp\left(-i\pi\frac{k^2\mu^2}{N}\right)\mathrm{frincd}^{*}\left(N;\frac{1}{\mu^2};\frac{k}{N}-\Delta f^{(s)}\xi\right). \tag{3.89b}$$

Then, obtain that PSF of sampled hologram reconstruction is

$$\begin{aligned}\overline{\mathrm{PSF}}(x,\xi;k) &= \overline{\mathrm{WPK}}(x,\xi)\cdot\overline{\mathrm{DRK}}(\xi,k)\\ &\propto \exp(i2\pi x\xi)\exp\left(-i\pi\lambda Z\xi^2\right)\\ &\quad\times\exp\left(-i\pi\frac{k^2\mu^2}{N}\right)\mathrm{frincd}^*\left(N;\frac{1}{\mu^2};\frac{k}{N}-\Delta f^{(s)}\xi\right)\end{aligned} \tag{3.90}$$

and that PSF of numerical reconstruction of Fresnel holograms with Fourier reconstruction algorithm is

$$\begin{aligned}\mathrm{PSF}(x,k) &= \exp\left(-i\pi\frac{k^2\mu^2}{N}\right)\\ &\quad\times\int_{-\infty}^{\infty}\Phi_f^{(s)}(\xi)\exp\left(-i\pi\lambda Z\xi^2\right)\mathrm{frincd}^*\left(N;\frac{1}{\mu^2};\frac{k}{N}-\Delta-+f^{(s)}\xi\right)\\ &\quad\times\exp(i2\pi x\xi)d\xi.\end{aligned} \tag{3.91}$$

Equation (3.91) is much more involved for an analytical treatment than the corresponding equation for PSF of numerical reconstruction of Fourier holograms and, in general, requires numerical methods for analysis. In order to facilitate its treatment, rewrite (3.91) using (3.89a) for $\overline{\mathrm{DRK}}(\xi,k)$:

$$\begin{aligned}\mathrm{PSF}(x,k) &= \exp\left[i\pi\left(\frac{x^2}{\lambda Z}-\frac{k^2\mu^2}{N}\right)\right]\\ &\quad\times\int_{-\infty}^{\infty}\Phi_f^{(s)}(\xi)\sum_{r=0}^{N-1}\exp\left(-i\pi\frac{r^2}{\mu^2N}\right)\\ &\quad\times\exp\left[i2\pi r\left(\frac{k}{N}-\Delta f^{(s)}\xi\right)\right]\exp\left[-i\pi\frac{(x-\lambda Z\xi)^2}{\lambda Z}\right]d\xi\\ &= \exp\left[i\pi\left(\frac{x^2}{\lambda Z}-\frac{k^2\mu^2}{N}\right)\right]\sum_{r=0}^{N-1}\exp\left(-i\pi\frac{r^2}{\mu^2N}\right)\exp\left(i2\pi\frac{kr}{N}\right)\\ &\quad\times\int_{-\infty}^{\infty}\Phi_f^{(s)}(\xi)\exp\left[-i\pi\frac{(x-\lambda Z\xi)^2}{\lambda Z}\right]\exp\left(-i2\pi r\Delta f^{(s)}\xi\right)d\xi.\end{aligned} \tag{3.92}$$

Assume now that frequency response of the hologram sampling device $\Phi_f^{(s)}(\xi)$ is constant, which is equivalent to the assumption that its PSF is a delta function.

Practically, this means that the hologram recording photographic camera is assumed to have a very small fill factor, or ratio of the size of camera sensitive elements to the interpixel distance. In this simplifying assumption,

$$
\begin{aligned}
\mathrm{PSF}_0(x,k) &= \exp\left[i\pi\left(\frac{x^2}{\lambda Z}-\frac{k^2\mu^2}{N}\right)\right]\sum_{r=0}^{N-1}\exp\left(-i\pi\frac{r^2}{\mu^2 N}\right)\\
&\quad\times\exp\left(i2\pi\frac{kr}{N}\right)\int_{-\infty}^{\infty}\exp\left[-i\pi\frac{(x-\lambda Z\xi)^2}{\lambda Z}\right]\exp\left(-i2\pi r\Delta f^{(s)}\xi\right)d\xi\\
&= \exp\left[i\pi\left(\frac{x^2}{\lambda Z}-\frac{k^2\mu^2}{N}\right)\right]\sum_{r=0}^{N-1}\exp\left(-i\pi\frac{r^2}{\mu^2 N}\right)\exp\left(i2\pi\frac{kr}{N}\right)\\
&\quad\times\int_{-\infty}^{\infty}\exp\left[-i\pi\frac{(x-\lambda Z\xi)^2}{\lambda Z}\right]\exp\left(-i2\pi r\Delta f^{(s)}\xi\right)d\xi\\
&= \exp\left[i\pi\left(\frac{x^2}{\lambda Z}-\frac{k^2\mu^2}{N}\right)\right]\sum_{r=0}^{N-1}\exp\left(-i\pi\frac{r^2}{\mu^2 N}\right)\exp\left(i2\pi\frac{kr}{N}\right)\\
&\qquad\times\int_{-\infty}^{\infty}\exp\left(-i\pi\frac{\widetilde{\xi}^2}{\lambda Z}\right)\exp\left(-i2\pi r\Delta f^{(s)}\frac{x-\widetilde{\xi}}{\lambda Z}\right)d\xi\\
&= \exp\left[i\pi\left(\frac{x^2}{\lambda Z}-\frac{k^2\mu^2}{N}\right)\right]\sum_{r=0}^{N-1}\exp\left(-i\pi\frac{r^2}{\mu^2 N}\right)\\
&\quad\times\exp\left(i2\pi\frac{kr}{N}\right)\exp\left(-i2\pi r\Delta f^{(s)}\frac{x}{\lambda Z}\right)\\
&\quad\times\int_{-\infty}^{\infty}\exp\left(-i\pi\frac{\widetilde{\xi}^2}{\lambda Z}\right)\exp\left(i2\pi\frac{r\Delta f^{(s)}}{\lambda Z}\widetilde{\xi}\right)d\xi\\
&\propto \exp\left[i\pi\left(\frac{x^2}{\lambda Z}-\frac{k^2\mu^2}{N}\right)\right]\\
&\quad\times\sum_{r=0}^{N-1}\exp\left(-i\pi\frac{r^2}{\mu^2 N}\right)\exp\left[i2\pi\left(\frac{k}{N}-\frac{\Delta f^{(s)}x}{\lambda Z}\right)r\right]\exp\left(i\pi\frac{r^2\Delta f^{(s)2}}{\lambda Z}\right)\\
&= \exp\left[i\pi\left(\frac{x^2}{\lambda Z}-\frac{k^2\mu^2}{N}\right)\right]\sum_{r=0}^{N-1}\exp\left[-i\pi\left(\frac{N\Delta f^{(s)2}}{\lambda Z}-\frac{1}{\mu^2}\right)\frac{r^2}{N}\right]\\
&\quad\times\exp\left[i2\pi\left(\frac{k}{N}-\frac{\Delta f^{(s)}x}{\lambda Z}\right)r\right],
\end{aligned}
\tag{3.93a}
$$

or

$$
\mathrm{PSF}_0(x,k)=\exp\left[i\pi\left(\frac{x^2}{\lambda Z}-\frac{k^2\mu^2}{N}\right)\right]\mathrm{frincd}^*\left(N;\frac{N\Delta f^{(s)2}}{\lambda Z}-\frac{1}{\mu^2};\frac{k}{N}-\frac{\Delta f^{(s)}x}{\lambda Z}\right).
\tag{3.93b}
$$

As one can see from (3.93b), PSF of numerical reconstruction of Fresnel holograms recorded with cameras with very small fill factor is basically proportional to frincd-function illustrated in Figures 3.2 and 3.3.

An important special case is "in focus" reconstruction, when

$$\mu^2 = \frac{\lambda Z}{N\Delta f^{(s)2}}. \tag{3.94}$$

In this case numerical reconstruction point spread function is discrete sinc-function

$$\begin{aligned}\mathrm{PSF}_0(x,k) &= \exp\left[i\frac{\lambda Z}{\Delta f^{(s)2}}\left(\frac{\Delta f^{(s)2}x^2}{\lambda^2 Z^2} - \frac{k^2}{N^2}\right)\right]\mathrm{sincd}\left[N;\pi\left(\frac{\Delta f^{(s)}x}{\lambda Z} - \frac{k}{N}\right)N\right] \\ &= \exp\left[i\frac{\lambda Z}{\Delta f^{(s)2}}\left(\frac{\Delta f^{(s)2}x^2}{\lambda^2 Z^2} - \frac{k^2}{N^2}\right)\right]\mathrm{sincd}\left[N;\frac{\pi(x-k\Delta_x)}{\Delta_x}\right],\end{aligned} \tag{3.95}$$

where $\Delta_x = \lambda Z/N\Delta f^{(s)} = \lambda Z/S_H$ is defined by (3.83). As one can see, "in focus" reconstruction PSF is essentially the same as that of numerical reconstruction of Fourier holograms (3.82) for the same assumption regarding the hologram sampling device. It has the same resolving power and provides aliasing free object reconstruction within the interval $S_o = \lambda Z/\Delta f^{(s)}$. One can establish a link between the size of this interval and the value $\mu^2 = \lambda Z/N\Delta f^{(s)2} = \lambda ZN/S_H^2$ of the focusing parameter required for the reconstruction:

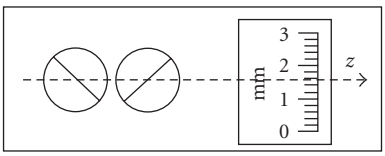
$$S_o = \frac{\lambda Z}{\Delta f^{(s)}} = \frac{\lambda ZN}{S_H} = \mu^2 S_H. \tag{3.96}$$

From this relationship it follows that aliasing free reconstruction of the object from a hologram recorded on a distance defined by the focusing parameter μ^2 is possible if the object size does not exceed the value $\mu^2 S_H$. Therefore, for $\mu^2 < 1$, allowed object size should be less than the hologram size otherwise aliasing caused by the periodicity of the discrete sinc-function will appear.

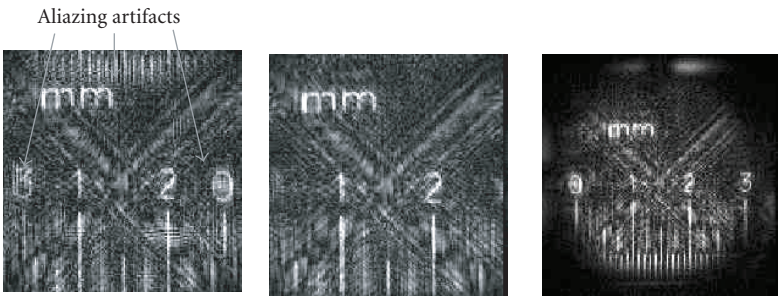
This phenomenon is illustrated in Figure 3.5. Figure 3.5(a) shows a sketch of an object that consists of two crossed thin wires and a ruler installed at different distances from the hologram.[1]

A result of reconstruction of the hologram for focusing parameter $\mu^2 = 0.66$ using the Fourier reconstruction algorithm is shown in Figure 3.5(b) in which one can clearly see aliasing artifacts due to overlapping of reconstructions from two side lobes of the discrete sinc-functions. One can remove these artifacts if, before the reconstruction, one sets to zeros outer parts of the hologram outside the circle

[1]The hologram courtesy to Dr. J. Campos, Autonomous University of Barcelona, Bellaterra, Barcelona, Spain.



(a)



(b) (c) (d)

FIGURE 3.5. Fresnel hologram reconstruction for $\mu^2 < 1$ using Fourier and convolution reconstruction algorithms: (a) sketch of the object setup, (b) and (c), respectively, reconstructions using the Fourier reconstruction algorithm without and with hologram masking with a circular window function of the diameter equal to μ^2th fraction of the hologram size, (d) reconstruction using the convolution reconstruction algorithm.

with diameter equal to μ^2th fraction of the hologram size. The result of such a reconstruction is shown in Figure 3.5(c). These artifacts are the reason why, for $\mu^2 < 1$, the convolution reconstruction algorithm is frequently used. The result of the reconstruction using the convolution algorithm is shown in Figure 3.5(d). Note that, with respect to the convolution algorithm reconstruction, Fourier reconstruction algorithm, with above-mentioned zeroing of the excessive part of the hologram, for $\mu^2 < 1$, acts as a "magnifying glass." Object aliasing artifacts for $\mu^2 < 1$ can also be avoided if the hologram is reconstructed by fragments of $\mu^2 S_H$ size.

3.7.3.2. Convolution reconstruction algorithm

In the convolution reconstruction algorithm, discrete reconstruction kernel is, with zero shifts in hologram and object wave front domains,

$$\mathrm{DRK}(r,k) = \mathrm{frincd}\,(N;\mu^2;k-r) = \frac{1}{N}\sum_{s=0}^{N-1}\exp\left(i\pi\frac{\mu^2 s^2}{N}\right)\exp\left[-i2\pi\frac{(k-r)s}{N}\right]. \tag{3.97}$$

Fourier series expansion over this kernel is

$$\begin{aligned}\overline{\mathrm{DRK}}(\xi,k) &= \frac{1}{N}\sum_{r=0}^{N-1}\sum_{s=0}^{N-1}\exp\left(i\pi\frac{\mu^2 s^2}{N}\right)\exp\left[-i2\pi\frac{(k-r)s}{N}\right]\exp\left(-i2\pi r\Delta f^{(s)}\xi\right)\\ &= \frac{1}{N}\sum_{s=0}^{N-1}\exp\left(i\pi\frac{\mu^2 s^2}{N}\right)\exp\left(-i2\pi\frac{ks}{N}\right)\sum_{r=0}^{N-1}\exp\left[i2\pi r\left(\frac{s}{N}-\Delta f^{(s)}\xi\right)\right].\end{aligned} \tag{3.98}$$

Then obtain that PSF of sampled hologram reconstruction is

$$\begin{aligned}\overline{\mathrm{PSF}}(x,\xi;k) = \overline{\mathrm{WPK}}(x,\xi)\cdot\overline{\mathrm{DRK}}(\xi,k) &\propto \frac{\exp(i2\pi x\xi)\exp(-i\pi\lambda Z\xi^2)}{N}\\ &\times\sum_{s=0}^{N-1}\exp\left(i\pi\frac{\mu^2 s^2}{N}\right)\exp\left(-i2\pi\frac{ks}{N}\right)\sum_{r=0}^{N-1}\exp\left[i2\pi r\left(\frac{s}{N}-\Delta f^{(s)}\xi\right)\right]\end{aligned} \tag{3.99}$$

and that PSF of numerical reconstruction of Fresnel holograms with Fourier reconstruction algorithm is

$$\begin{aligned}\mathrm{PSF}(x,k) &= \int_{-\infty}^{\infty}\Phi_f^{(s)}(\xi)\exp\left(-i\pi\lambda Z\xi^2\right)\exp(i2\pi x\xi)d\xi\\ &\quad\times\frac{1}{N}\sum_{s=0}^{N-1}\exp\left(i\pi\frac{\mu^2 s^2}{N}\right)\exp\left(-i2\pi\frac{ks}{N}\right)\sum_{r=0}^{N-1}\exp\left[i2\pi r\left(\frac{s}{N}-\Delta f^{(s)}\xi\right)\right]\\ &= \int_{-\infty}^{\infty}\Phi_f^{(s)}(\xi)\exp\left(-i\pi\lambda Z\xi^2\right)\exp(i2\pi x\xi)d\xi\\ &\quad\times\sum_{s=0}^{N-1}\exp\left(i\pi\frac{\mu^2 s^2}{N}\right)\exp\left(-i2\pi\frac{ks}{N}\right)\\ &\quad\times\frac{\sin\left[\pi\left(s-N\Delta f^{(s)}\xi\right)\right]}{N\sin\left[(\pi/N)\left(s-N\Delta f^{(s)}\xi\right)\right]}\exp\left[i\pi\frac{N-1}{N}\left(s-N\Delta f^{(s)}\xi\right)\right],\end{aligned} \tag{3.100}$$

or finally,

$$\begin{aligned}\mathrm{PSF}(x,k) &= \int_{-\infty}^{\infty}\Phi_f^{(s)}(\xi)\exp\left(-i\pi\lambda Z\xi^2\right)\exp(i2\pi x\xi)d\xi\\ &\quad\times\sum_{s=0}^{N-1}\exp\left(i\pi\frac{\mu^2 s^2}{N}\right)\exp\left(-i2\pi\frac{ks}{N}\right)\\ &\quad\times\mathrm{sincd}\left[N;\pi\left(s-N\Delta f^{(s)}\xi\right)\right]\exp\left[i\pi\frac{N-1}{N}\left(s-N\Delta f^{(s)}\xi\right)\right].\end{aligned} \tag{3.101}$$

Note that the last phase exponential term in (3.101) is inessential, as it appears due to the assumption of zero-shift parameters in the reconstruction algorithm.

Although (3.101) is quite nontransparent for analysis, at least one important property, that of periodicity of the PSF over object sample index k with period N, can be immediately seen from it. As, by the definition of the convolutional Fresnel transform, sampling interval $\Delta x^{(r)}$ in the object plane is identical to the hologram sampling interval $\Delta f^{(s)}$, this periodicity of the PSF implies that object wave front is reconstructed within the physical interval $N\Delta x = N\Delta f^{(s)} = S_H$, where S_H is the physical size of the hologram. Further detailed analysis is not feasible without bringing in numerical methods. Some results of such a numerical analysis[2] are illustrated in Figure 3.6

Figures 3.6(a) and 3.6(b), in particular, reveal that, although object sampling interval in the convolution method is deliberately set equal to the hologram sampling interval, resolving power of the method is still defined by the same fundamental value $\lambda Z/S_H$ as that of the Fourier reconstruction algorithm and of the Fourier reconstruction algorithm for Fourier holograms. One can clearly see this when one compares width of the main lobe of the point spread function in Figure 3.6(a) with the distance between vertical ticks that indicate object sampling positions and from observing, in Figure 3.6(b), three times widening of the width of the main lobe of PSF that corresponds to the object-to-hologram distance $Z = 15$ with respect to that for $Z = 5$. Figure 3.6(c) shows reconstruction of nine point sources placed uniformly within the object size. The plot vividly demonstrates that the hologram sampling device point function acts very similarly to its action in case of the Fourier reconstruction algorithm and of reconstruction of Fourier holograms: it is masking the reconstruction result with a function close to its Fourier transform.

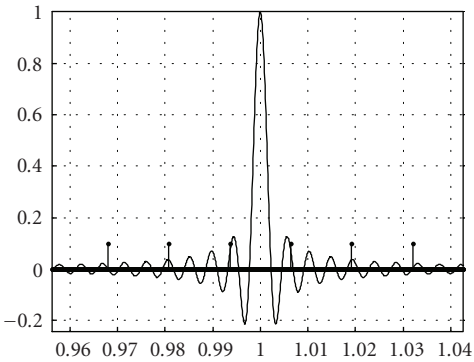
Finite width of the reconstruction algorithm PSFs does not only limit their resolving power, but it also produces speckle noise in the reconstruction of diffuse objects. This phenomenon is illustrated in Figure 3.7 that compares reconstructions of rectangular shaped nondiffuse and diffuse objects.

3.8. Conclusion

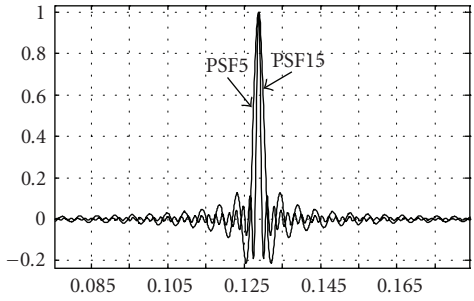
In this chapter, we described discrete transforms and algorithms for numerical reconstruction of digitally recorded holograms. The transforms are numerical approximations to diffractional integral transforms and, as such, are oriented on FFT-type of fast algorithms for their computer implementation. In Appendix B, a summary table of transforms is provided for reference and ease of comparison.

We also analyzed point spread functions of main hologram reconstruction algorithms to show how reconstruction aliasing artifacts and reconstruction resolving power depend on physical parameters of holographic optical setups and digital cameras used for hologram recording.

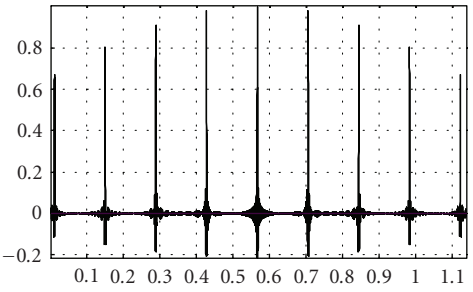
[2]The analysis was carried out with the help of Dr. Fucai Zhang, Institute of Technical Optics, Stuttgart University, Germany.



(a)



(b)



(c)

FIGURE 3.6. Point spread function of the convolution algorithm: (a) central lobe of the PSF shown along with boundaries of object sampling interval (vertical ticks); (b) PSF of reconstructions for two distances between object and hologram ($Z = 5$ and $Z = 15$); (c) reconstruction result for 9 point sources placed uniformly within object area.

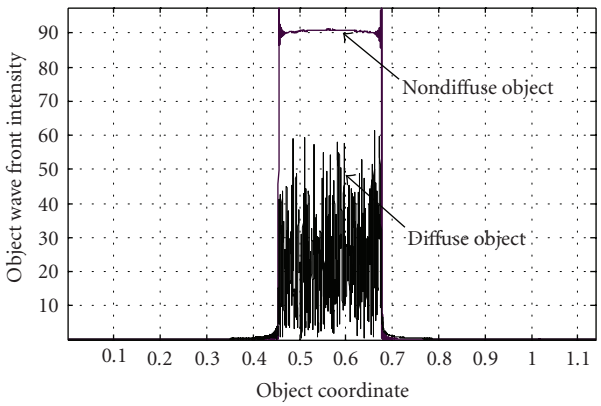


FIGURE 3.7. Results of reconstruction of a Fresnel hologram of nondiffuse and diffuse objects that demonstrate appearance of heavy speckle noise for the diffuse object.

One can also add that same discrete transforms can be used for synthesis of computer-generated holograms as well. In this application, computer-generated holograms are reconstructed optically, and the relationship between properties of transforms and the reconstruction results depends on methods of hologram recording. Interested readers can find more details in [3, 9].

Appendices

A. The principle of phase-shifting holography

In phase-shifting holography, several holograms are recorded with a reference beam having different preset phase shifts. Let θ_k be a phase shift of the reference beam in kth hologram exposure, $k = 1, \ldots, K$. Then,

$$\begin{aligned} H_k(f_x, f_y) &= \left|\alpha(f_x, f_y) + R(f_x, f_y)\exp(i\theta_k)\right|^2 \\ &= \alpha(f_x, f_y)R^*(f_x, f_y)\exp(-i\theta_k) \\ &\quad + \alpha^*(f_x, f_y)R(f_x, f_y)\exp(i\theta_k) + \left|\alpha(f_x, f_y)\right|^2 + \left|R(f_x, f_y)\right|^2 \end{aligned} \tag{A.1}$$

is a hologram recorded in kth exposure. For reconstruction of the term $\alpha(f_x, f_y)$ that represents object wave front, K holograms $\{H_k\}$ are summed up with the

same phase shift used in their recording:

$$\begin{aligned}\overline{H} &= \frac{1}{K}\sum_{k=1}^{K} H_k \exp(i\theta_k) \\ &= \alpha(f_x, f_y)R^*(f_x, f_y) + \alpha^*(f_x, f_y)R(f_x, f_y)\sum_{k=1}^{K}\exp(i2\theta_k) \\ &\quad + \left[\,|\alpha(f_x, f_y)|^2 + |R(f_x, f_y)|\,\right]\sum_{k=1}^{K}\exp(i\theta_k).\end{aligned} \tag{A.2}$$

For perfect reconstruction of the first term, phases $\{\theta_k\}$ should be found from equations

$$\sum_{k=1}^{K}\exp(i\theta_k) = 0, \tag{A.3a}$$

$$\sum_{k=1}^{K}\exp(i2\theta_k) = 0. \tag{A.3b}$$

Assume $\theta_k = k\theta_0$. Then,

$$\begin{aligned}\sum_{k=1}^{K}\exp(i\theta_k) &= \sum_{k=1}^{K}\exp(ik\theta_0) = \frac{\exp[i(K+1)\theta_0] - \exp(i\theta_0)}{\exp(i\theta_0) - 1} \\ &= \frac{\exp(iK\theta_0) - 1}{\exp(i\theta_0) - 1}\exp(i\theta_0), \\ \sum_{k=1}^{K}\exp(i2\theta_k) &= \sum_{k=1}^{K}\exp(i2k\theta_0) = \frac{\exp[i2(K+1)\theta_0] - \exp(i2\theta_0)}{\exp(i2\theta_0) - 1} \\ &= \frac{\exp(i2K\theta_0) - 1}{\exp(i2\theta_0) - 1}\exp(i2\theta_0) \\ &= \frac{\exp(iK\theta_0) - 1}{\exp(i\theta_0) - 1}\frac{\exp(iK\theta_0) + 1}{\exp(i\theta_0) + 1}\exp(i2\theta_0),\end{aligned} \tag{A.4}$$

from which it follows that solution of (A.3a) is $\theta_k = 2\pi(k/K)$ for any integer $K \geq 3$. For $K = 2$, solution $\theta_0 = \pi$ does not satisfy (A.3b) as

$$\frac{\exp(i2\theta_0) - 1}{\exp(i\theta_0) - 1}\frac{\exp(i2\theta_0) + 1}{\exp(i\theta_0) + 1}\exp(i2\theta_0) = [\exp(i2\pi) + 1]\exp(i2\pi) = 2. \tag{A.5}$$

B. Computation of convolution of signals with "mirror reflection" extension using discrete cosine and discrete cosine/sine transforms

Let signal $\{\widetilde{a}_k\}$ be obtained from signal $\{a_k\}$ of N samples by its extension by "mirror reflection" and periodical replication of the result with a period of $2N$ samples:

$$\widetilde{a}_{(k) \bmod 2N} = \begin{cases} a_k, & k = 0, 1, \ldots, N-1, \\ a_{2N-k-1}, & k = N,\ N = 1, \ldots, 2N-1, \end{cases} \tag{B.1}$$

and let $\{h_n\}$ be a convolution kernel of N samples ($n = 0, 1, \ldots, N-1$). Then,

$$\widetilde{c}_k = \sum_{n=0}^{N-1} h_n \widetilde{a}_{(k-n+[N/2]) \bmod 2N}, \tag{B.2}$$

where

$$\left[\frac{N}{2}\right] = \begin{cases} \dfrac{N}{2} & \text{for even } N, \\ \dfrac{(N-1)}{2} & \text{for odd } N \end{cases} \tag{B.3}$$

is a convolution of the extended signal $\{\widetilde{a}_k\}$ with kernel $\{h_n\}$. It will coincide with the cyclic convolution of period $2N$:

$$c_{(k) \bmod 2N} = \sum_{n=0}^{N-1} \widetilde{h}_{(n) \bmod 2N} \widetilde{a}_{(k-n+[N/2]) \bmod 2N} \tag{B.4}$$

for kernel

$$\widetilde{h}_{(n) \bmod 2N} = \begin{cases} 0, & n = 0, \ldots, \left[\dfrac{N}{2}\right] - 1, \\ h_{n-[N/2]}, & n = \left[\dfrac{N}{2}\right], \ldots, \left[\dfrac{N}{2}\right] + N - 1, \\ 0, & \left[\dfrac{N}{2}\right] + N, \ldots, 2N-1. \end{cases} \tag{B.5}$$

The cyclic convolution described by (B.4) and (B.5) is illustrated in Figure B.1.

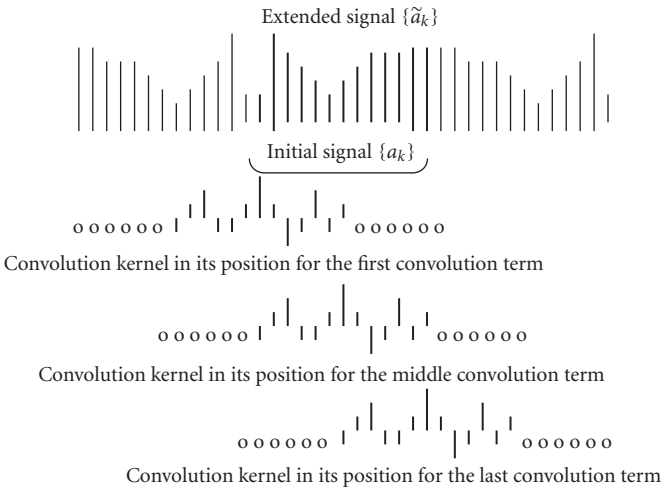


FIGURE B.1. Cyclic convolution of a signal extended by its mirror reflection from its borders.

Consider computing convolution of such signals by means of IDFT of product of signal DFT spectra. DFT spectrum of the extended signal $\{\widetilde{a}_k\}$ is

$$\begin{aligned}\widetilde{\alpha}_r &= \frac{1}{\sqrt{2N}} \sum_{k=0}^{2N-1} \widetilde{a}_k \exp\left(i2\pi \frac{kr}{2N}\right) \\ &= \left\{ \frac{2}{\sqrt{2N}} \sum_{k=0}^{N-1} a_k \cos\left[\pi \frac{(k+1/2)r}{N}\right] \right\} \exp\left(-i\pi \frac{r}{2N}\right) = \alpha_r^{(\mathrm{DCT})} \exp\left(-i\pi \frac{r}{2N}\right),\end{aligned} \tag{B.6}$$

where

$$\alpha_r^{(\mathrm{DCT})} = \mathrm{DCT}\,\{a_k\} = \frac{2}{\sqrt{2N}} \sum_{k=0}^{N-1} a_k \cos\left(\pi \frac{k+1/2}{N}\right) \tag{B.7}$$

is discrete cosine transform (DCT) of the initial signal $\{a_k\}$. Therefore, DFT spectrum of the signal extended by "mirror reflection" can be computed via DCT using fast DCT algorithm. From properties of DCT, it follows that DCT spectra feature the following symmetry property:

$$\alpha_N^{\mathrm{DCT}} = 0, \qquad \alpha_k^{\mathrm{DCT}} = -\alpha_{2N-k}^{\mathrm{DCT}}. \tag{B.8}$$

For computing convolution, the signal spectrum defined by (B.6) should be multiplied by the filter frequency response for signals of $2N$ samples and then, the

inverse DFT should be computed for the first N samples:

$$b_k = \frac{1}{\sqrt{2N}} \sum_{r=0}^{2N-1} \alpha_r^{(\mathrm{DCT})} \exp\left(-i\pi \frac{r}{2N}\right) \widetilde{\eta}_r \exp\left(-i2\pi \frac{kr}{2N}\right), \tag{B.9}$$

where $\{\widetilde{\eta}_r\}$ are the DFT coefficients of samples of the filter PSF (filter discrete frequency response):

$$\widetilde{\eta}_r = \frac{1}{\sqrt{2N}} \sum_{n=0}^{2N-1} \widetilde{h}_{(n) \bmod 2N} \exp\left(i2\pi \frac{nr}{2N}\right). \tag{B.10}$$

They are real numbers and, therefore, they feature the symmetry property

$$\{\eta_r = \eta_{2N-r}^*\}, \tag{B.11}$$

where asterisk symbolizes complex conjugate. From (B.9) and (B.11), one can now obtain that

$$\begin{aligned} b_k &= \frac{1}{\sqrt{2N}} \left\{ \alpha_0^{(\mathrm{DCT})} \eta_0 + \sum_{r=1}^{N-1} \alpha_r^{(\mathrm{DCT})} \eta_r \exp\left(-i2\pi \frac{k+1/2}{2N} r\right) \right. \\ &\qquad \left. + \sum_{r=1}^{N-1} \alpha_{2N-r}^{(\mathrm{DCT})} \eta_{2N-r} \exp\left[-i2\pi \frac{k+1/2}{2N} (2N-r)\right] \right\} \\ &= \frac{1}{\sqrt{2N}} \left\{ \alpha_0^{(\mathrm{DCT})} \eta_0 + \sum_{r=1}^{N-1} \alpha_r^{(\mathrm{DCT})} \left[\eta_r \exp\left(-i2\pi \frac{k+1/2}{2N} r\right) \right. \right. \\ &\qquad \left. \left. + \eta_r^* \exp\left(i2\pi \frac{k+1/2}{2N} r\right) \right] \right\}. \end{aligned} \tag{B.12}$$

As

$$\begin{aligned} &\eta_r \exp\left(-i2\pi \frac{k+1/2}{2N} r\right) + \eta_r^* \exp\left(i2\pi \frac{k+1/2}{2N} r\right) \\ &\quad = 2\,\mathrm{Re}\left[\eta_r \exp\left(-i2\pi \frac{k+1/2}{2N} r\right)\right] \\ &\quad = \eta_r^{\mathrm{re}} \cos\left(\pi \frac{k+1/2}{N} r\right) - \eta_r^{\mathrm{im}} \sin\left(\pi \frac{k+1/2}{N} r\right), \end{aligned} \tag{B.13}$$

where $\{\eta_r^{\mathrm{re}}\}$ and $\{\eta_r^{\mathrm{im}}\}$ are real and imaginary parts of $\{\eta_r\}$, we obtain, finally,

$$b_k = \frac{1}{\sqrt{2N}}\left\{\alpha_0^{(\mathrm{DCT})}\eta_0 + \sum_{r=1}^{N-1}\alpha_r^{(\mathrm{DCT})}\eta_r^{\mathrm{re}}\cos\left(\pi\frac{k+1/2}{N}r\right) - \sum_{r=1}^{N-1}\alpha_r^{(\mathrm{DCT})}\eta_r^{\mathrm{im}}\sin\left(\pi\frac{k+1/2}{N}r\right)\right\}. \tag{B.14}$$

The first two terms of this expression constitute inverse DCT of the product $\{\alpha_r^{(\mathrm{DCT})}\eta_r^{\mathrm{re}}\}$ while the third term is discrete cosine/sine transform (DcST) of the product $\{\alpha_r^{(\mathrm{DCT})}\eta_r^{\mathrm{im}}\}$. Both transforms can be computed using corresponding fast algorithms ([9]).

C. Approximation of function frincd $(\cdot)$

Consider a discrete analog of the known relationship (see [17]):

$$\int_{-\infty}^{\infty}\exp\left(i\pi\sigma^2x^2\right)\exp(-i2\pi fx)dx = \frac{\sqrt{i}}{\sigma}\exp\left(-i\pi\frac{f^2}{\sigma^2}\right). \tag{C.1}$$

By definition of integral,

$$\int_{-\infty}^{\infty}\exp\left(i\pi\sigma^2x^2\right)\exp(-i2\pi fx)dx = \lim_{\substack{N\to\infty\\ \Delta x\to 0}}\sum_{k=-N/2}^{N/2-1}\exp\left(i\pi\sigma^2k^2\Delta x^2\right)\exp(-i2\pi rk\Delta x\Delta f)\Delta x, \tag{C.2}$$

where $x = k\Delta x$, and the integral is considered in points $f = r\Delta f$. Select $\Delta f\Delta x = 1/N$ and assume that **N** is an odd number. Then,

$$\int_{-\infty}^{\infty}\exp\left(i\pi\sigma^2x^2\right)\exp(-i2\pi fx)dx = \lim_{\substack{N\to\infty\\ \Delta x\to 0}}\sum_{k=-(N-1)/2}^{(N-1)/2}\exp\left(i\pi\sigma^2k^2\Delta x^2\right)\exp(-i2\pi rk\Delta x\Delta f)\Delta x. \tag{C.3}$$

Therefore,

$$\lim_{\substack{N\to\infty\\ \Delta x\to 0}} \frac{1}{N\Delta f} \sum_{k=-(N-1)/2}^{(N-1)/2} \exp\left(i\pi \frac{\sigma^2}{N\Delta f^2}\frac{k^2}{N^2}\right) \exp\left(-i2\pi\frac{rk}{N}\right) = \lim_{N\to\infty} \frac{\sqrt{i}}{\sigma} \exp\left(-i\pi\frac{r^2\Delta f^2}{\sigma^2}\right). \tag{C.4}$$

Denote $\sigma^2/N\Delta f^2 = q$. Then,

$$\lim_{N\to\infty} \frac{1}{N\Delta f} \sum_{k=-(N-1)/2}^{(N-1)/2} \exp\left(i\pi\frac{qk^2}{N}\right) \exp\left(-i2\pi\frac{rk}{N}\right) = \lim_{N\to\infty} \frac{\sqrt{i}}{\Delta f\sqrt{Nq}} \exp\left(-i\pi\frac{r^2\Delta f^2}{q\Delta f^2 N}\right) = \lim_{N\to\infty} \frac{\sqrt{i}}{\Delta f\sqrt{Nq}} \exp\left(-i\pi\frac{r^2}{qN}\right), \tag{C.5}$$

or

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=-(N-1)/2}^{(N-1)/2} \exp\left(i\pi\frac{qk^2}{N}\right) \exp\left(-i2\pi\frac{rk}{N}\right) = \lim_{N\to\infty} \frac{\sqrt{i}}{\sqrt{Nq}} \exp\left(-i\pi\frac{r^2}{qN}\right). \tag{C.6}$$

Therefore, one can say that

$$\frac{1}{N} \sum_{k=-(N-1)/2}^{(N-1)/2} \exp\left(i\pi\frac{qk^2}{N}\right) \exp\left(-i2\pi\frac{rk}{N}\right) \cong \frac{\sqrt{i}}{\sqrt{Nq}} \exp\left(-i\pi\frac{r^2}{qN}\right). \tag{C.7}$$

It is assumed in this formula that both k and r are running in the range $-(N-1)/2, \ldots, 0, \ldots, (N-1)/2$. Introduce variables $n = k+(N-1)/2$ and $s = r+(N-1)/2$ to convert this formula to the formula that corresponds to the canonical DFT in

which variables n and s run in the range $0,\ldots,N-1$,

$$\begin{aligned}
&\frac{1}{N}\sum_{k=-(N-1)/2}^{(N-1)/2}\exp\left(i\pi\frac{qk^2}{N}\right)\exp\left(-i2\pi\frac{rk}{N}\right)\\
&\quad=\frac{1}{N}\sum_{n=0}^{N-1}\exp\left\{i\pi\frac{q[n-(N-1)/2]^2}{N}\right\}\\
&\qquad\times\exp\left\{-i2\pi\frac{[s-(N-1)/2][n-(N-1)/2]}{N}\right\}\\
&\quad=\frac{1}{N}\exp\left[i\pi\left(\frac{q}{2}-1\right)\frac{(N-1)^2}{2N}\right]\exp\left[i\pi\frac{s(N-1)}{N}\right]\\
&\qquad\times\sum_{n=0}^{N-1}\exp\left(i\pi\frac{qn^2}{N}\right)\exp\left[i\pi\frac{(1-q)(N-1)}{N}n\right]\exp\left(-i2\pi\frac{ns}{N}\right)\\
&\quad\cong\frac{\sqrt{i}}{\sqrt{Nq}}\exp\left\{-i\pi\frac{[s-(N-1)/2]^2}{qN}\right\}.
\end{aligned}\tag{C.8}$$

From this, we have

$$\begin{aligned}
&\frac{1}{N}\sum_{n=0}^{N-1}\exp\left(i\pi\frac{qn^2}{N}\right)\exp\left(-i2\pi\frac{n[s-(1-q)(N-1)/2]}{N}\right)\\
&\qquad\cong\frac{\sqrt{i}}{\sqrt{Nq}}\exp\left\{-i\pi\left[\left(\frac{q}{2}-1\right)\frac{(N-1)^2}{2N}\right]\right\}\\
&\qquad\times\exp\left[-i\pi\frac{s(N-1)}{N}\right]\exp\left\{-i\pi\frac{[s-(N-1)/2]^2}{qN}\right\}
\end{aligned}\tag{C.9}$$

within boundaries $(1-q)((N-1)/2) < s < (1+q)((N-1)/2)$, $0 \le q \le 1$, or finally,

$$\begin{aligned}
&\frac{1}{N}\sum_{n=0}^{N-1}\exp\left(i\pi\frac{qn^2}{N}\right)\exp\left[-i2\pi\frac{n(s-v_q)}{N}\right]\\
&\qquad\cong\sqrt{\frac{i}{Nq}}\exp\left[-i\pi\frac{(s-v_q)^2}{qN}\right]\operatorname{rect}\left[\frac{s-v_q}{q(N-1)}\right],
\end{aligned}\tag{C.10}$$

where $v_q=(1-q)(N-1)/2$, $0\le q\le 1$, and s runs from 0 to $N-1$.

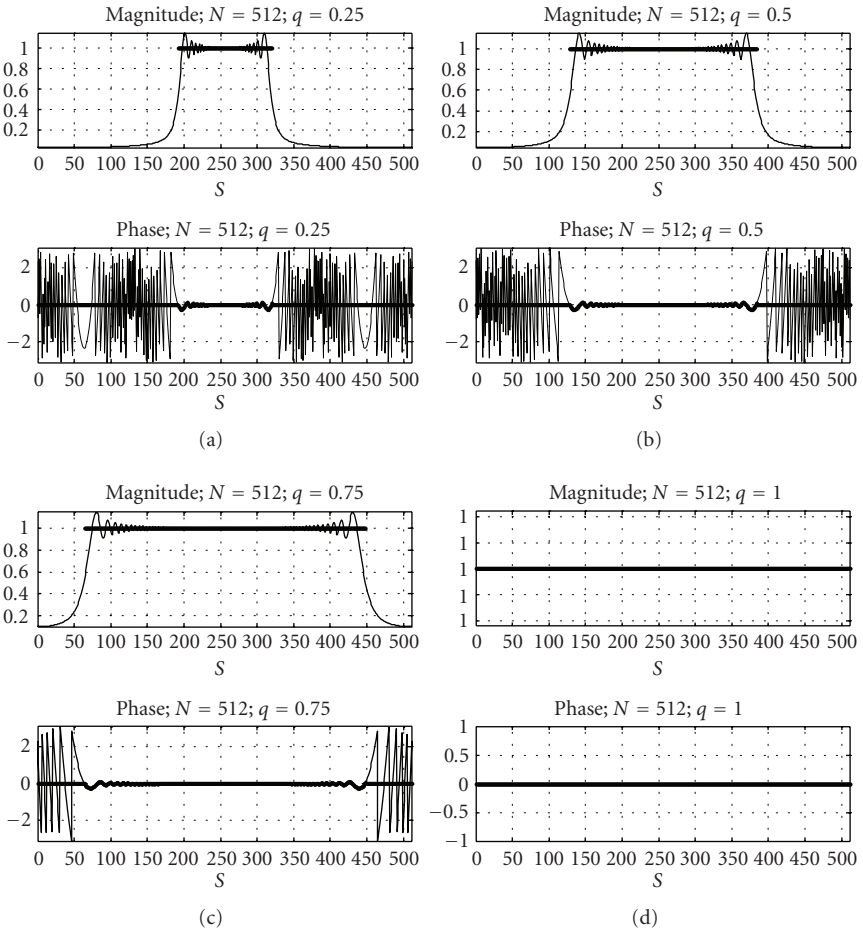


FIGURE C.1. Plots of absolute values (magnitude) and phase (in radians) of function RCT (**N**; **q**, **s**, (C.11), for $N = 512$ and four different values of **q** (0.25; 0.5; 0.75 and 1.0). Rect-function that approximates this function is shown in bold line.

Numerical evaluation of the relationship

$$\begin{aligned}\mathrm{RCT}\,(N;q;s) &= \frac{1}{N}\sum_{n=0}^{N-1}\exp\left(i\pi\frac{qn^2}{N}\right)\\ &\quad\times\exp\left[-i2\pi\frac{n(s-v_q)}{N}\right]\Big/\sqrt{\frac{i}{Nq}}\exp\left[-i\pi\frac{(s-v_q)^2}{qN}\right]\\ &= \frac{\sqrt{q}\exp\left[i\pi\left((s-v_q)^2/qN\right)\right]}{\sqrt{iN}}\\ &\quad\times\sum_{n=0}^{N-1}\exp\left(i\pi\frac{qn^2}{N}\right)\exp\left[-i2\pi\frac{n(s-v_q)}{N}\right]\cong\mathrm{rect}\left[\frac{s-v_q}{q(N-1)}\right]\end{aligned}\tag{C.11}$$

confirms the validity of this approximation (see [16]). This is illustrated in Figure C.1 for $N = 512$ and four different values of q (0.25, 0.5, 0.75, and 1.0). Rect-function of (C.11) is shown in bold line.

D. Summary of the fast discrete diffractional transforms

Table D.1

Transform	
Canonical discrete Fourier transform (DFT)	$\alpha_r = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \exp\left(i2\pi \frac{kr}{N}\right)$
Shifted DFT	$\alpha_r^{u,v} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \exp\left[i2\pi \frac{(k+u)(r+v)}{N}\right]$
Discrete cosine transform(DCT)	$\alpha_r^{\mathrm{DCT}} = \frac{2}{\sqrt{2N}} \sum_{k=0}^{N-1} a_k \cos\left(\pi \frac{k+1/2}{N} r\right)$
Discrete cosine-sine transform (DcST)	$\alpha_r^{\mathrm{DcST}} = \frac{2}{\sqrt{2N}} \sum_{k=0}^{N-1} a_k \sin\left(\pi \frac{k+1/2}{N} r\right)$
Scaled DFT	$\alpha_r^{\sigma} = \frac{1}{\sqrt{\sigma N}} \sum_{k=0}^{N-1} a_k \exp\left[i2\pi \frac{(k+u)(r+v)}{\sigma N}\right] = \frac{1}{\sqrt{\sigma N}} \sum_{k=0}^{N-1} a_k \exp\left(i2\pi \frac{\widetilde{k}\widetilde{r}}{\sigma N}\right)$
Scaled DFT as a cyclic convolution	$\alpha_r^{\sigma} = \frac{\exp\left(i\pi(\widetilde{r}^2/\sigma N)\right)}{\sqrt{\sigma N}} \sum_{k=0}^{N-1} \left[a_k \exp\left(i\pi \frac{\widetilde{k}^2}{\sigma N}\right)\right] \exp\left[-i\pi \frac{(\widetilde{k}-\widetilde{r})^2}{\sigma N}\right]$
Canonical 2D DFT	$\alpha_{r,s} = \frac{1}{\sqrt{N_1 N_2}} \sum_{k=0}^{N_1-1} \sum_{l=0}^{N_2-1} a_{k,l} \exp\left[i2\pi\left(\frac{kr}{N_1} + \frac{ls}{N_1}\right)\right]$
Affine DFT	$\alpha_{r,s} = \sum_{k=0}^{N_1-1} \sum_{l=0}^{N_2-1} a_{k,l} \times \exp\left[i2\pi\left(\frac{rk}{\sigma_A N_1} + \frac{sk}{\sigma_C N_1} + \frac{rl}{\sigma_B N_2} + \frac{sl}{\sigma_D N_2}\right)\right]$
Rotated DFT (RotDFT)	$\alpha_{r,s} = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{k,l} \exp\left[i2\pi\left(\frac{r\cos\theta - s\sin\theta}{N} k + \frac{r\sin\theta + s\cos\theta}{N} l\right)\right] = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{k,l} \exp\left[i2\pi\left(\frac{rk+sl}{N}\cos\theta - \frac{sk-rl}{N}\sin\theta\right)\right]$
Rotated scaled DFT	$\alpha_{r,s} = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{k,l} \exp\left[i2\pi\left(\frac{r\cos\theta - s\sin\theta}{\sigma N} k + \frac{r\sin\theta + s\cos\theta}{\sigma N} l\right)\right] = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} a_{k,l} \exp\left[i2\pi\left(\frac{rk+sl}{\sigma N}\cos\theta - \frac{sk-rl}{\sigma N}\sin\theta\right)\right]$
Discrete sinc-function	$\mathrm{sincd}(N,x) = \frac{\sin x}{N \sin(x/N)}$
Canonical discrete Fresnel transform (DFrT)	$\alpha_r = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \exp\left[i\pi \frac{(k/\mu - r\mu)^2}{N}\right]; \quad \mu^2 = \lambda Z / N \Delta f^2$

Table D.1. Continued.

Transform	
Shifted DFrT	$\alpha_r^{(\mu,w)} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \times \exp\left[-i\pi \frac{(k\mu - r/\mu + w)^2}{N}\right]; \quad w = u/\mu - v\mu$
Fourier reconstruction algorithm for Fresnel holograms	$\alpha_r^{(\mu,w)} = \frac{\exp\left(-i\pi(r^2/\mu^2 N)\right)}{\sqrt{N}} \times \sum_{k=0}^{N-1} a_k \exp\left[-i\pi \frac{(k\mu + w)^2}{N}\right] \exp\left(i2\pi \frac{k + w/\mu}{N} r\right)$
Focal plane invariant DFrT	$\alpha_r^{(\mu,(N/2\mu))} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \times \exp\left\{-i\pi \frac{[k\mu - (r - N/2)/\mu]^2}{N}\right\}$
Partial DFrT (PDFT)	$\hat{\alpha}_r^{(\mu,w)} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \times \exp\left(-i\pi \frac{k^2\mu^2}{N}\right) \exp\left[i2\pi \frac{k(r - w\mu)}{N}\right]$
Convolutional discrete Fresnel transform (ConvDFrT)	$\alpha_r = \sum_{k=0}^{N-1} a_k \text{frincd}\,(N; \mu^2; r + w - k) = \frac{1}{N} \sum_{s=0}^{N-1} \left[\sum_{k=0}^{N-1} a_k \exp\left(i2\pi \frac{k - r - w}{N} s\right)\right] \times \exp\left(-i\pi \frac{\mu^2 s^2}{N}\right)$
Convolutional reconstruction algorithm for Fresnel holograms	$\alpha_r = \frac{1}{N} \sum_{s=0}^{N-1} \left[\sum_{k=0}^{N-1} a_k \exp\left(i2\pi \frac{ks}{N}\right)\right] \times \exp\left(-i\pi \frac{\mu^2 s^2}{N}\right) \exp\left(-i2\pi \frac{r + w}{N} s\right)$
Frincd-function	$\text{frincd}\,(N; q; x) = \frac{1}{N} \sum_{r=0}^{N-1} \exp\left(i\pi \frac{qr^2}{N}\right) \times \exp\left(-i2\pi \frac{xr}{N}\right)$ $\text{frincd}\,(N; 1; x) = \sqrt{\frac{i}{N}} \exp\left(-i\pi \frac{x^2}{N}\right); \quad x \in Z$ Analytical approximation: $\text{frincd}\,(N; \pm q; x) \cong \sqrt{\frac{\pm i}{Nq}} \exp\left[\mp i\pi \frac{x^2}{qN}\right] \times \text{rect}\left[\frac{x}{q(N-1)}\right]$
Discrete Kirchhoff-Rayleigh-Sommerfeld transform (DKRST)	$\alpha_r = \sum_{k=0}^{N-1} a_k \times \frac{\exp\left[i2\pi\left(\tilde{z}^2\sqrt{1 + (\tilde{k} - \tilde{r})^2/z^2}/\mu^2 N\right)\right]}{1 + (\tilde{k} - \tilde{r})^2/z^2}$

Bibliography

[1] J. W. Goodman and R. W. Lawrence, "Digital image formation from electronically detected holograms," *Applied Physics Letters*, vol. 11, no. 3, pp. 77–79, 1967.

[2] M. A. Kronrod, N. S. Merzlyakov, and L. P. Yaroslavsky, "Reconstruction of a hologram with a computer," *Soviet Physics-Technical Physics*, vol. 17, no. 2, pp. 419–420, 1972.

[3] L. P. Yaroslavsky and N. S. Merzlyakov, *Methods of Digital Holography*, Consultant Bureau, New York, NY, USA, 1980, (English translation from Russian, L. P. Yaroslavsky, N. S. Merzlyakov, *Methods of Digital Holography*, Izdatel'stvo Nauka, Moscow, Russia, 1977).

[4] U. Schnars and W. Juptner, "Direct recording of holograms by a CCD target and numerical reconstruction," *Applied Optics*, vol. 33, no. 2, pp. 179–181, 1994.

[5] I. Yamaguchi and T. Zhang, "Phase-shifting digital holography," *Optics Letters*, vol. 22, no. 16, pp. 1268–1270, 1997.

[6] E. Cuche, F. Bevilacqua, and Ch. Depeursinge, "Digital holography for quantitative phase-contrast imaging," *Optics Letters*, vol. 24, no. 5, pp. 291–293, 1999.

[7] E. N. Leith and J. Upatnieks, "New techniques in wavefront reconstruction," *Journal of the Optical Society of America*, vol. 51, pp. 1469–1473, 1961.

[8] J. Goodman, *Introduction to Fourier Optics*, McGraw-Hill, New York, NY, USA, 1996.

[9] L. P. Yaroslavsky, *Digital Holography and Digital Image Processing*, Kluwer Academic, Boston, Mass, USA, 2004.

[10] L. P. Yaroslavsky, "Shifted discrete Fourier transforms," in *Digital Signal Processing*, V. Cappellini, Ed., pp. 69–74, Academic Press, London, UK, 1980.

[11] L. R. Rabiner, R. W. Schafer, and C. M. Rader, "The chirp z-transform algorithm and its application," *The Bell System Technical Journal*, vol. 48, pp. 1249–1292, 1969.

[12] X. Deng, B. Bihari, J. Gan, F. Zhao, and R. T. Chen, "Fast algorithm for chirp transforms with zooming-in ability and its applications," *Journal of the Optical Society of America A*, vol. 17, no. 4, pp. 762–771, 2000.

[13] V. Namias, "The fractional order Fourier transform and its application to quantum mechanics," *Journal of the Institute of Mathematics and Its Applications*, vol. 25, no. 3, pp. 241–265, 1980.

[14] D. H. Bailey and P. N. Swarztrauber, "The fractional Fourier transform and applications," *SIAM Review*, vol. 33, no. 3, pp. 389–404, 1991.

[15] L. P. Yaroslavsky and N. Ben-David, "Focal-plane-invariant algorithm for digital reconstruction of holograms recorded in the near diffraction zone," in *Optical Measurement Systems for Industrial Inspection III*, W. Osten, M. Kujawinska, and K. Creath, Eds., vol. 5144 of *Proceedings of SPIE*, pp. 142–149, Munich, Germany, June 2003.

[16] L. P. Yaroslavsky, F. Zhang, and I. Yamaguchi, "Point spread functions of digital reconstruction of digitally recorded holograms," in *Information Optics and Photonics Technology*, G. Mu, F. T. S. Yu, and S. Jutamulia, Eds., vol. 5642 of *Proceedings of SPIE*, pp. 271–282, Beijing, China, November 2005.

[17] I. S. Gradstein and I. M. Ryzhik, *Tables of Integrals, Series, and Products*, Academic Press, Boston, Mass, USA, 1994.

Leonid P. Yaroslavsky: Department of Interdisciplinary Studies, Faculty of Engineering,
Tel Aviv University, Tel Aviv 69978, Israel
Email: yaro@eng.tau.ac.il

4 Irregular sampling for multidimensional polar processing of integral transforms

A. Averbuch, R. Coifman, M. Israeli, I. Sedelnikov, and Y. Shkolnisky

We survey a family of theories that enable to process polar data via integral transforms. We show the relation between irregular sampling and discrete integral transforms, demonstrate the application of irregular (polar) sampling to image processing problems, and derive approximation algorithms that are based on unequally spaced samples. It is based on sampling the Fourier domain. We describe 2D and 3D irregular sampling geometries of the frequency domain, derive efficient numerical algorithms that implement them, prove their correctness, and provide theory and algorithms that invert them. We also show that these sampling geometries are closely related to discrete integral transforms. The proposed underlying methodology bridges via sampling between the continuous nature of the physical phenomena and the discrete nature world. Despite the fact that irregular sampling is situated in the core of many scientific applications, there are very few efficient numerical tools that allow robust processing of irregularly sampled data.

4.1. Introduction

Polar (directional) processing and irregular sampling interleave each other in many physical, scientific, and computational disciplines. Despite the fact that it is situated in the core of many scientific applications, there are very few efficient numerical tools that allow robust processing of irregularly sampled polar data. As a result, the solution for problems that involve irregular sampling usually resorts to approximation or regridding to a Cartesian grid, where efficient numerical tools exist. This necessarily sacrifices some aspects of the solution like accuracy and analytical properties. Sometimes it is possible to trade accuracy at the expense of speed by using slow algorithms. However, this approach fails in high dimensions, where straightforward slow algorithms are impractical. As a result, solutions for high-dimensional problems sometimes consist of separable application of 1D tools, which sacrifices the interrelations between dimensions. In this chapter, we

Professor Moshe Israeli passed away on February 18, 2007. This chapter is dedicated to his memory.

survey a coherent related family of approaches that enable to process irregularly sampled data, as well as some applications of these algorithms to image processing problems. We consider sampling of the Fourier domain of discrete objects. We describe 2D and 3D irregular sampling geometries of the frequency domain, derive efficient numerical algorithms that implement them, prove their correctness, and provide theory and algorithms that invert them. We present the 2D pseudopolar Fourier transform, which samples the Fourier transform of an image on a near-polar grid. The 2D pseudopolar Fourier transform also presents discrete integral transforms, which provide discretizations of continuous integral transforms to discrete objects. The formulation of these directional processing tools in the frequency domain results in special irregular sampling patterns. The algorithms that implement the discrete transforms are based on sampling the Fourier transform of the input object on these irregularly distributed points. We proved that these transforms are algebraically accurate and preserve the geometric properties of the continuous transforms.

We present the 2D and 3D pseudopolar Fourier transforms. The pseudopolar Fourier transform is a fast algorithm that samples the Fourier transform of an image on the pseudopolar grid. The pseudopolar grid, also known as the concentric squares grid, consists of equally spaced samples along rays, where different rays are equally spaced and not equally angled. This grid is different from the polar grid since the angles between the rays are unequal. The algorithm that computes the pseudopolar Fourier transform is shown to be fast (the same complexity as the FFT), stable, invertible, requires only 1D operations, and uses no interpolations. As a result, the pseudopolar Fourier transform is accurate and requires no parameters besides the input function. The algorithm that computes the pseudopolar Fourier transform is based on 1D applications of the fractional Fourier transform,[1] which samples the Fourier transform of a 1D sequence at arbitrary equally spaced points. Since the pseudopolar Fourier transform is computed using only 1D operations, it is suitable for real-time implementations. Although the algorithm produces irregular samples in the Fourier domain, it does not use any interpolations, and the computed values have machine accuracy. The pseudopolar Fourier transform is invertible. Invertibility is of major importance from a practical point of view, as many real-life problems can be formulated as the recovery of image samples from frequency samples (e.g., medical imagery reconstruction algorithms). Invertibility assures that no information about the physical phenomenon is lost due to the transformation. The inversion algorithms for the pseudopolar Fourier transform face several numerical and computational difficulties, as the transform is ill-conditioned and not selfadjoint. We consider two inversion algorithms for the pseudopolar Fourier transform: iterative and direct. The iterative algorithm is based on the application of the conjugate-gradient method to the Gram operator of the pseudopolar Fourier transform. Since both the forward pseudopolar Fourier transform and its adjoint can be computed in $O(N^2 \log N)$ and $O(3^2 \log N)$ operations, where $N \times N$ and $N \times N \times N$ are the sizes of the input

[1] It is also called scaled DFT in the terminology of Chapter 3 in this book.

images, respectively, the Gram operator can also be computed in the same complexity. However, since the transform is ill-conditioned, we introduce a preconditioner, which significantly accelerates the convergence. In addition, we develop a direct inversion algorithm, which resamples the pseudopolar grid to a Cartesian frequency grid, and then, recovers the image from the Cartesian frequency grid. The algorithm is based on an "onion-peeling" procedure that at each step recovers two rows/columns of the Cartesian frequency grid, from the outermost rows/columns to the origin, by using columns/rows recovered in previous steps. The Cartesian samples of each row/column are recovered using trigonometric interpolation that is based on a fast multipole method (FMM). Finally, the original image is recovered from the Cartesian frequency samples, which are not the standard DFT samples, by using a fast Toeplitz solver. Then, we use the pseudopolar Fourier transform to construct the 2D discrete Radon transform.

The Radon transform is a fundamental tool in many areas, for example, in reconstruction of an image from its projections (CT imaging). Although it is situated in the core of many modern physical computations, the Radon transform lacks a coherent discrete definition for 2D discrete images, which is algebraically exact, invertible, and rapidly computable. We define a 2D discrete Radon transform for discrete 2D images, which is based on summation along lines with absolute slopes less than 1. Values at nongrid locations are defined using trigonometric interpolation on a zero-padded grid where shearing is the underlying transform. The discrete 2D definition of the Radon transform is shown to be geometrically faithful as the lines used for summation exhibit no wraparound effects. We show that our discrete Radon transform satisfies the Fourier slice theorem, which states that the 1D Fourier transform of the discrete Radon transform is equal to the samples of the pseudopolar Fourier transform of the underlying image that lie along a ray. We show that the discrete Radon transform converges to the continuous Radon transform, as the discretization step goes to zero. This property is of major theoretical and computational importance since it shows that the discrete transform is indeed an approximation of the continuous transform, and thus can be used to replace the continuous transform in digital implementations. We utilize the same concepts used in the construction of the 2D discrete Radon transform to derive a 3D discrete X-ray transform. The analysis of 3D discrete volumetric data becomes increasingly important as computation power increases. 3D analysis and visualization applications are expected to be especially relevant in areas like medical imaging and nondestructive testing where extensive continuous theory exists. However, this theory is not directly applicable to discrete datasets. Therefore, we have to establish theoretical foundations for discrete analysis tools that will replace the existing inexact discretizations, which are based on the continuous theory. We want the discrete theory to preserve the concepts, properties, and main results of the continuous theory. Then, we develop a discretization of the continuous X-ray transform that operates on 3D discrete images. The suggested transform preserves summation along true geometric lines without using arbitrary interpolation schemes. As for the 2D discrete Radon transform, the 3D discrete X-ray transform satisfies the Fourier slice theorem, which relates the Fourier transform of a

3D object to the Fourier transform of its discrete X-ray transform. Specifically, we prove that the samples of the Fourier transform of a 3D object on a certain family of planes is equal to the 2D Fourier transform of the 3D discrete X-ray transform. We then derive a fast algorithm that computes the discrete X-ray transform that is based on the Fourier slice theorem. The algorithm resamples the 3D Fourier transform of the input object over the required set of planes by using the 1D chirp Z-transform, and then, applies to each plane the 2D inverse Fourier transform. The resampling of the Fourier transform of the input object over the set of planes is accurate, involves no interpolations, and requires only 1D operations. Finally, we show that the discrete X-ray transform is invertible.

Radon and X-ray transformations are closely related. However, there is a fundamental difference between them. The Radon transform in n-dimensions decomposes an n-dimensional object into a set of integrals over hyperplanes of dimension n^2. On the other hand, an n-dimensional X-ray transform decomposes an object into its set of line integrals. The two transforms coincide in the 2D case. The 2D discrete Radon transform together with the 3D discrete Radon transform and the 3D discrete X-ray transform provide a complete framework for defining n-dimensional Radon and X-ray transforms for discrete objects, as extensions from 3D to n-dimensions are straightforward. These transforms are algebraically accurate and geometrically faithful, as they do not involve arbitrary interpolations and preserve summation along lines/hyperplanes. These transforms are invertible, can be computed using fast algorithms, and parallel with the continuum theory, as they preserve, for example, the Fourier slice theorem.

A discrete transform that results in a different sampling geometry is the discrete diffraction transform. The continuous diffraction transform is the mathematical model that underlies, for example, ultrasound imaging. The discrete diffraction transform is defined as a collection of discrete diffracted projections taken at a specific set of angles along a specific set of lines. A discrete diffracted projection is a discrete transform whose properties are shown to be similar to the properties of the continuous diffracted projection. We prove that when the discrete diffraction transform is applied on samples of a continuous object, it approximates a set of continuous vertical diffracted projections of a horizontally sheared object and a set of continuous horizontal diffracted projections of a vertically sheared object. Also, we prove that the discrete transform satisfies the Fourier diffraction theorem, is rapidly computable, and is invertible.

Recently, we proposed two algorithms for the reconstruction of a 2D object from its continuous projections. The first algorithm operates on parallel projection data, while the second uses the more practical model of fan-beam projections. Both algorithms are based on the discrete Radon transform, which extends the continuous Radon transform to discrete data. The discrete Radon transform and its inverse can be computed in a complexity comparable with the 2D FFT, and are shown to accurately model the continuum as the number of samples increases. Numerical results demonstrate high quality reconstructions for both parallel and fan-beam acquisition geometries. The sane idea is currently extended to process a 3D object.

4.1.1. CT processing

An important problem in image processing is to reconstruct a cross section of an object from several images of its projections. A projection is a shadowgram obtained by illuminating an object by penetrating radiation. Figure 4.1(a) shows a typical method for obtaining projections. Each horizontal line shown in this figure is a one-dimensional projection of a horizontal slice of the object. Each pixel of the projected image represents the total absorption of the X-ray along its path from the source to the detector. By rotating the source-detector assembly around the object, projections for several different angles can be obtained. The goal of *image reconstruction from projections* is to obtain an image of a cross section of the object from these projections. Imaging systems that generate such slice views are called CT (computerized tomography) scanners.

The Radon transform is the underlying fundamental concept used for CT scanning, as well as for a wide range of other disciplines, including radar imaging, geophysical imaging, nondestructive testing, and medical imaging [1–4].

4.1.1.1. 2D continuous Radon transform

For the 2D case, the Radon transform of a function $f(x, y)$, denoted as $\Re f(\theta, s)$, is defined as its line integral along a line L inclined at an angle θ and at distance s from the origin (see Figure 4.1(b)). Formally,

$$\Re f(\theta, s) = \int_L f(x, y)du = \iint_{-\infty}^{\infty} f(x, y)\delta(x\cos\theta + y\sin\theta - s)dx\,dy, \qquad (4.1)$$

where $\delta(x)$ is Dirac's delta function. The Radon transform maps the spatial domain (x, y) to the domain (θ, s). Each point in the (θ, s) space corresponds to a line in the spatial domain (x, y).

In the paper, we will use the following version of the *continuous direct and inverse Fourier transform*:

$$\hat{f}(w) = \int_{-\infty}^{\infty} f(x)e^{-2\pi\imath wx}dx, \qquad f(x) = \int_{-\infty}^{\infty} \hat{f}(w)e^{2\pi\imath wx}dw. \qquad (4.2)$$

4.1.1.2. The Fourier slice theorem

There is a fundamental relationship between the 2D Fourier transform of a function and the 1D Fourier transform of its Radon transform. The result is summarized in the following theorem.

Theorem 4.1.1 (Fourier slice theorem). *The 1D Fourier transform with respect to s of the projection $\Re f(\theta, s)$ is equal to a central slice, at angle θ, of the 2D Fourier*

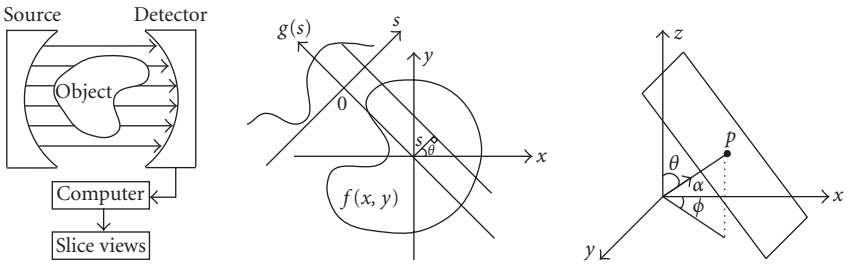


FIGURE 4.1. (a) An X-ray CT scanning system. (b) 2D spatial domain. (c) 3D projection geometry.

transform of the function $f(x, y)$. *That is,*

$$\widehat{\Re f}(\theta, \xi) = \hat{f}(\xi \cos\theta, \xi \sin\theta), \tag{4.3}$$

where $\hat{f}(\xi_1, \xi_2) = \iint_{-\infty}^{\infty} f(x, y) e^{-2\pi\imath(x\xi_1 + y\xi_2)} dx\, dy$ *is the 2D Fourier transform of* $f(x, y)$.

4.1.1.3. 3D continuous Radon transform

The 2D Radon transform, which is obtained using 1D line integrals, can be generalized to 3D using integrals on planes. The 3D Radon transform is defined using 1D projections of a 3D object $f(x, y, z)$ where these projections were obtained by integrating $f(x, y, z)$ on a plane, whose orientation can be described by a unit vector $\vec{\alpha}$ (see Figure 4.1(c)). Formally, we have the following.

Definition 4.1.2 (3D continuous Radon transform). Given a 3D function $f(\vec{x}) \triangleq f(x, y, z)$, and a plane (whose representation is given using the normal $\vec{\alpha}$ and the distance from the origin s), the Radon transform for this plane is defined by

$$\begin{aligned} \Re f(\vec{\alpha}, s) &= \iiint_{-\infty}^{\infty} f(\vec{x}) \delta(\vec{x}^T \vec{\alpha} - s) d\vec{x} \\ &= \iiint_{-\infty}^{\infty} f(x, y, z) \delta(x \sin\theta \cos\phi + y \sin\theta \sin\phi + z \cos\theta - s) dx\, dy\, dz, \end{aligned} \tag{4.4}$$

where $\vec{x} = [x, y, z]^T$, $\vec{\alpha} = [\sin\theta \cos\phi, \sin\theta \sin\phi, \cos\theta]^T$, and δ is Dirac's delta function.

The Fourier slice theorem for the 3D discrete Radon transform is the following.

Theorem 4.1.3 (3D Fourier slice theorem). *The 1D Fourier transform with respect to s of the function* $\Re f(\vec{\alpha}, s)$ *is equal to a central slice at direction* $\vec{\alpha}$ *of the 3D Fourier transform of the function* $f(\vec{x})$*. That is,*

$$\widehat{\Re f}(\vec{\alpha}, \xi) = \hat{f}(\xi\vec{\alpha}) = \hat{f}(\xi \sin\theta\cos\phi, \xi \sin\theta \sin\phi, \xi\cos\theta), \tag{4.5}$$

where

$$\hat{f}(\xi_1, \xi_2, \xi_3) = \iiint_{-\infty}^{\infty} f(\vec{x}) e^{-2\pi\imath(\vec{x}^T \cdot \vec{\xi})} d\vec{x}, \quad \vec{\xi} = [\xi_1, \xi_2, \xi_3]^T, \ \vec{x} = [x, y, z]^T. \tag{4.6}$$

4.1.2. Discretization of the Radon transform

For modern applications it is important to have a discrete analogues of $\Re f$ for digital images $I = (I(u, v) : -n/2 \le u, v < n/2)$ for the 2D case and $I = (I(u, v, w) : -n/2 \le u, v, w < n/2)$ for the 3D case. As guidelines for developing the notion of the discrete Radon transform, we define the following set of properties which should be satisfied by any definition of the discrete Radon transform: algebraic exactness, geometric fidelity, be rapidly computable, for example, admit an $O(N \log N)$ algorithm where N is the size of the data in I ($N = n^2$ in the 2D case, $N = n^3$ in the 3D case), invertible and parallels with continuum theory.

The structure of the paper is the following: in Section 4.2 the relationship between earlier works and the processed results in this chapter is discussed. The 2D pseudopolar Fourier transform is described in Section 4.3, which is the basis for the rest of the developments. The 2D discrete Radon transform following the notion in [6, 7] is described in Section 4.4. In Section 4.5, we give a definition of the 3D discrete Radon transform for 3D discrete images using slopes, and explain its relation to conventional plane representation using normals. We present a 3D Fourier slice theorem for the 3D discrete Radon transform and define the 3D pseudopolar Fourier transform. Finally, for a special set of planes, we show that the 3D discrete Radon transform is rapidly computable. The 3D discrete X-ray transform is described in Section 4.6. In the summary section we mentioned some applications that are based on this methodology.

4.2. Related works

We describe here the relationship between earlier works and the processed results in this chapter. Over the years, many attempts were made to define the notion of the discrete Radon transform. For a survey of these approaches see [7, 8]. All these approaches failed to meet the requirements in Section 4.1.2 simultaneously, but recently [5–8, 10] established a coherent methodology for the definition of the discrete Radon transform that satisfies these requirements. In this chapter, we sketch this approach for the 2D and 3D cases. The full constructions and proofs can be found in [5–7, 11] for the 2D case and in [8, 10] for the 3D case. A combined description appears in [18].

Bailey and Swarztrauber [21] described the fractional Fourier transform (FRFFT). It lacks the discussion of the chirp-Z transform, dating from twenty years earlier [22], which is in fact more general than the fractional Fourier transform as introduced in [21]. Bailey and Swarztrauber consider the problem of summing along a family of lines in an array, and suggest the idea like in our case of trigonometric interpolation. However, FFRT does not explicitly define or defend a notion "Radon transform." Moreover, they do not proceed as above, that is, establishing a projection-slice theorem which relates sums along lines to Fourier coefficients on a pseudopolar grid. Another difference is that their definition of summing along lines can be shown equivalent to ours using interpolating kernel D_m for $m = n$ rather than $m = 2n + 1$. For us, this is a crucial difference, because of the wraparound artifacts that it causes. It shows that coefficients in the $m = n$ case are indeed formed by summing along wrapped lines. FRFFT deserve a great deal of credit for an important series of algorithmic insights. Our contributions are (1) to insist on a specific definition of a full Radon transform, using a specific set of offsets and angles, and not merely a convenient algorithm for general families of sums; (2) to insist on $m = 2n + 1$ and so on geometric fidelity of the corresponding family of lines; (3) to formalize a pseudopolar FFT and recognize a projection-slice theorem relating the pseudopolar FFT to the Radon; (4) to establish properties for the Radon transform, including injectivity and inversion algorithms; and (5) to put the various results in scholarly context. In a survey paper (see [23]) a non-Cartesian grid in the 2D Fourier plane was introduced. This is a pseudopolar grid of the type we have described here in the paper, except for the degree of radial sampling. It is called there the concentric squares grid. The authors of that paper assumed that data on a continuum object were gathered in unequally spaced projections chosen so that the 1D Fourier transform corresponded to the concentric squares grid. They considered the problem of reconstructing a discrete array of n^2 pixels from such Fourier domain data, and developed an algorithm based on interpolating from the data given in the concentric squares grid to the Cartesian grid. They used simple 1-dimensional interpolation based on linear interpolation in rows/columns. In short, a key organizational tool, a trapezoidal grid for Fourier space, has been known since 1974, under the name concentric squares grid. In fact, this grid has since been rediscovered numerous times. The authors in [23] seem to be the real inventors of this concept. In comparison to our work, (1) [23]'s definition samples half as frequently as that in the radial direction. This can be shown to be exactly the grid which would arise if we had developed our original Radon definition for the $m = n$ case. Hence, the original concentric squares grid involves wraparound of the underlying lines; (2) [23]'s methodology is about reconstruction from data given about a continuum object; the authors do not attempt to define a Radon transform on digital data, or establish the invertibility and conditioning of such a transform; and (3) their methodology is approximate; they do not obtain an exact conversion between concentric squares and Cartesian grids.

Another important set of papers in the literature of computed tomography are both medical tomography [24–26] and synthetic aperture radar imaging [27].

Like [23], these authors are concerned with image reconstruction; effectively they assume that one is given data in the Fourier domain on a concentric squares grid.

Reference [24], unpublished work, which is known among tomography experts through a citation in Natterer's book [3], showed in 1980 that by the given data on a pseudopolar grid in Fourier space, one could calculate a collection of n^2 sums which, using the notation, we can write as

$$\sum c_{k,l}^{s} \exp\left\{i(u,v)'\zeta_{k,l}^{s}\right\}, \quad -\frac{n}{2} \le u,v < \frac{n}{2}, \tag{4.7}$$

where the $\zeta_{k,l}^{s}$ are points in the concentric squares grid. (Reference [24] makes no reference to [23].) Pasciak in [24] studied this calculation, which is essentially the calculation of adj P for a variant of P based on $m = n$ rather than $m = 2n + 1$, and showed it may be done in order $n^2 \log n$ time. His key insight was to use the chirp-Z transform to calculate Fourier-like sums with exponents different from the usual $2\pi/nkt$ by a factor α.

The authors in [25, 26] develop the linogram, with a very similar point of view. They assume that data on a continuum object have been gathered by what we have called the continuous Slant Stack, at a set of projections which are equispaced in $\tan(\theta)$ rather than θ. By digitally sampling each constant θ projection and making a 1D discrete Fourier transform of the resulting samples, they argue that they are essentially given data on a concentric squares grid in Fourier space, (making no reference to [23] or [24]). They are concerned with reconstruction and consider the sum (4.7) and derive a fast algorithm—the same as [24], using again the chirp-Z transform.

References [25, 27] develop the so called Polar Fourier transform for Synthetic Aperture Radar (SAR) imagery. They introduce a concentric squares grid, assuming that SAR data are essentially given on such a concentric squares grid in Fourier space, and consider the problem of rapidly reconstructing an image from such data. They consider the sum (4.7) and derive a fast algorithm using again the chirp-Z transform. They refer to [23]. These authors deserve major credit for identifying an important algorithmic idea use of chirp-Z techniques to resample data from Cartesian to concentric squares grids which obviously is the same idea we use in our fast algorithms.

In comparison to our work, (1) this methodology is about reconstruction only, assuming that data are gathered about a continuum object by a physical device, and (2) the algorithmic problem they consider is equivalent to rapidly computing (4.7).

4.3. 2D pseudopolar Fourier transform

The pseudopolar representation is the basis for the rest of the developments. Therefore, we provide here a detailed description for its construction. Given an

image I of size $n \times n$, its 2D Fourier transform, denoted by $\hat{I}(\omega_x, \omega_y)$, is given by

$$\hat{I}(\omega_x, \omega_y) = \sum_{u,v=-n/2}^{n/2-1} I(u,v) e^{-(2\pi\imath/m)(u\omega_x + v\omega_y)}, \quad \omega_x, \omega_y \in \mathbb{R}, \tag{4.8}$$

Fourier transform of I where $m \geq n$ is an arbitrary integer. We assume for simplicity that the image I has equal dimensions in the x- and y- directions and that n is even. Equation (4.8) uses continuous frequencies ω_x and ω_y. For practical applications we need to evaluate $\hat{I}$ on discrete sets. Given a set Ω, we denote the samples of $\hat{I}$ on Ω by $\hat{I}_\Omega$. For example, let Ω_c be the Cartesian grid of size $m \times m$ given by

$$\Omega_c = \left\{(k,l),\ k,l = \frac{-m}{2}, \ldots, \frac{m}{2-1}\right\}. \tag{4.9}$$

Then, the Fourier transform in (4.8) has the form

$$\hat{I}_{\Omega_c}(k,l) \triangleq \hat{I}(k,l) = \sum_{u,v=-n/2}^{n/2-1} I(u,v) e^{-(2\pi\imath/m)(uk+vl)}, \tag{4.10}$$

$k,l = -m/2, \ldots, m/2-1$, which is usually referred to as the 2D DFT of the image I. It is well known that $\hat{I}_{\Omega_c}$, given by (4.10), can be computed in $O(m^2 \log m)$ operations using the FFT algorithm.

4.3.1. Definition

Definition 4.3.1 (2D pseudopolar grid). The 2D pseudopolar grid, denoted by Ω_{pp}, is given by

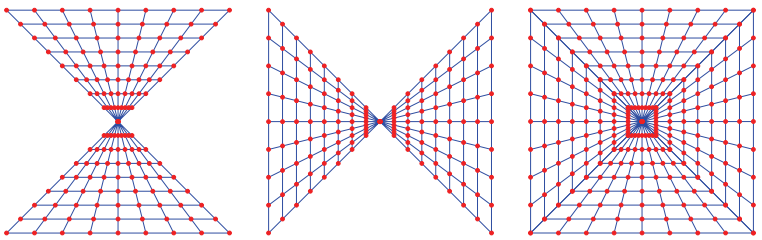
$$\Omega_{\text{pp}} \triangleq \Omega_{\text{pp}}^1 \cup \Omega_{\text{pp}}^2, \tag{4.11}$$

where

$$\begin{aligned} \Omega_{\text{pp}}^1 &\triangleq \left\{\left(-\frac{2l}{n}k, k\right) \mid \frac{-n}{2} \leq l \leq \frac{n}{2},\ -n \leq k \leq n\right\}, \\ \Omega_{\text{pp}}^2 &\triangleq \left\{\left(k, -\frac{2l}{n}k\right) \mid \frac{-n}{2} \leq l \leq \frac{n}{2},\ -n \leq k \leq n\right\}. \end{aligned} \tag{4.12}$$

We denote a specific point in Ω_{pp}^1 and Ω_{pp}^2 by

$$\begin{aligned} \Omega_{\text{pp}}^1(k,l) \triangleq \left(-\frac{2l}{n}k, k\right), \qquad \Omega_{\text{pp}}^2(k,l) \triangleq \left(k, -\frac{2l}{n}k\right), \\ k = -n, \ldots, n, \quad l = \frac{-n}{2}, \ldots, \frac{n}{2}. \end{aligned} \tag{4.13}$$



(a) (b) (c)

FIGURE 4.2. (a) Pseudopolar sector Ω^1_{pp}. (b) Pseudopolar sector Ω^2_{pp}. (c) Pseudopolar grid $\Omega_{\text{pp}} = \Omega^1_{\text{pp}} \cup \Omega^2_{\text{pp}}$.

See Figure 4.2 for an illustration of Ω^1_{pp}, Ω^2_{pp}, and Ω_{pp}, respectively. As can be seen from the figures, k serves as a "pseudo-radius" and l serves as a "pseudo-angle."

The resolution of the pseudopolar grid, given by Definition 4.3.1, is $n+1$ in the angular direction and $m = 2n+1$ in the radial direction. In polar coordinates, the pseudopolar grid is given by

$$\Omega^1_{\text{pp}}(k,l) = (r^1_k, \theta^1_l), \qquad \Omega^2_{\text{pp}}(k,l) = (r^2_k, \theta^2_l), \tag{4.14}$$

where

$$\begin{aligned} r^1_k &= k\sqrt{4\left(\frac{l}{n}\right)^2 + 1}, & r^2_k &= k\sqrt{4\left(\frac{l}{n}\right)^2 + 1}, \\ \theta^1_l &= \frac{\pi}{2} - \arctan\left(\frac{2l}{n}\right), & \theta^2_l &= \arctan\left(\frac{2l}{n}\right), \end{aligned} \tag{4.15}$$

$k = -n, \ldots, n$ and $l = -n/2, \ldots, n/2$.

As we can see in Figure 4.2(c), for each fixed angle l, the samples of the pseudopolar grid are equally spaced in the radial direction. However, this spacing is different for different angles. Also, the grid is not equally spaced in the angular direction, but has equally spaced slopes. Formally,

$$\begin{aligned} \Delta r^1_k &\triangleq r^1_{k+1} - r^1_k = \sqrt{4\left(\frac{l}{n}\right)^2 + 1}, & \Delta r^2_k &\triangleq r^2_{k+1} - r^2_k = \sqrt{4\left(\frac{l}{n}\right)^2 + 1}, \\ \Delta \tan\theta^1_{\text{pp}}(l) &\triangleq \cot\theta^1_{l+1} - \cot\theta^1_l = \frac{2}{n}, & \Delta \tan\theta^2_{\text{pp}}(l) &\triangleq \tan\theta^2_{l+1} - \tan\theta^2_l = \frac{2}{n}, \end{aligned} \tag{4.16}$$

where r^1_k, r^2_k, θ^1_l, and θ^2_l are given by (4.15).

The pseudopolar Fourier transform is defined by resampling $\widehat{I}$, given by (4.8), on the pseudopolar grid Ω_{pp}, given by (4.11)-(4.12).

Definition 4.3.2 (pseudopolar Fourier transform). The pseudopolar Fourier transform $\hat{I}_{\Omega_{\text{pp}}^j}$ $(j = 1, 2)$ is a linear transformation that is defined for $k = -n, \ldots, n$ and $l = -n/2, \ldots, n/2$ as

$$\hat{I}_{\Omega_{\text{pp}}^1}(k,l) = \hat{I}\Big(-\frac{2l}{n}k, k\Big), \qquad \hat{I}_{\Omega_{\text{pp}}^2}(k,l) = \hat{I}\Big(k, -\frac{2l}{n}k\Big) \tag{4.17}$$

on the set S where $\hat{I}$ is given by (4.8). Equivalently, the more compact notation $\hat{I}_{\Omega_{\text{pp}}}(s,k,l) \triangleq \hat{I}_{\Omega_{\text{pp}}^s}(k,l)$, where $s = 0, 1$, $k = -n, \ldots, n$, and $l = -n/2, \ldots, n/2$, is used.

Using operator notation, we denote the pseudopolar Fourier transform of an image I as $\mathcal{F}_{\text{pp}}I$, where

$$(\mathcal{F}_{\text{pp}}I)(s,k,l) \triangleq \hat{I}_{\Omega_{\text{pp}}^s}(k,l), \tag{4.18}$$

$s = 0, 1$, $k = -n, \ldots, n$, $l = -n/2, \ldots, n/2$, and $\hat{I}_{\Omega_{\text{pp}}^s}$ is given by (4.17). This notation is used whenever operator notation is more convenient.

4.3.2. Fast forward transform

In this section, we present a fast algorithm that efficiently computes the pseudopolar Fourier transform of an image I. The idea behind the algorithm is to evaluate $\hat{I}$, given by (4.8), on a Cartesian grid, by using the 2D FFT algorithm, and then, resampling the Cartesian frequency grid to the pseudopolar grid. The operator behind this algorithm is the *fractional Fourier transform*.

Definition 4.3.3 (fractional Fourier transform). Let $c \in \mathbb{C}^{n+1}$ be a vector of length $n+1$, $c = (c(u),\ u = -n/2, \ldots, n/2)$, and let $\alpha \in \mathbb{R}$. The fractional Fourier transform, denoted by $(F_{n+1}^{\alpha}c)(k)$, is given by

$$(F_{n+1}^{\alpha}c)(k) = \sum_{u=-n/2}^{n/2} c(u)e^{-2\pi\imath\alpha ku/(n+1)}, \quad k = \frac{-n}{2}, \ldots, \frac{n}{2}. \tag{4.19}$$

An important property of the fractional Fourier transform is that given a vector c of length $n+1$, the sequence $(F_{n+1}^{\alpha}c)(k)$, $k = -n/2, \ldots, n/2$, can be computed using $O(n \log n)$ operations for any $\alpha \in \mathbb{R}$ (see [21]).

Definition 4.3.3 is usually referred to as the unaliased fractional Fourier transform. It differs from the usual definition of the fractional Fourier transform given in [21]. According to the definition in [21], for a vector $c = (c(u),\ u = 0, \ldots, n)$ and an arbitrary $\alpha \in \mathbb{R}$, the fractional Fourier transform is defined as

$$(F_n^{\alpha}c)(k) = \sum_{u=0}^{n-1} c(u)e^{-2\pi\imath\alpha ku}, \quad k = 0, \ldots, n-1. \tag{4.20}$$

The algorithm that computes the unaliased fractional Fourier transform (Definition 4.3.3) is very similar to the algorithm in [21], and is therefore omitted.

The algorithm that computes the pseudopolar Fourier transform uses the following notation.

(i) E: padding operator. $E(I, m, n)$ accepts an image I of size $n \times n$ and symmetrically zero-pads it to size $m \times n$.
(ii) F_1^{-1}: 1D inverse DFT.
(iii) $\widetilde{F}_m^{\alpha}$: fractional Fourier transform with factor α. The operator accepts a sequence of length n, symmetrically pads it to length $m = 2n+1$, applies to it the fractional Fourier transform with factor α, and returns the $n+1$ central elements.
(iv) F_2: 2D DFT.
(v) $G_{k,n}$: resampling operator given by

$$G_{k,n} = \widetilde{F}_m^{\alpha} \circ F_1^{-1}, \quad \alpha = \frac{2k}{n}. \tag{4.21}$$

Using this notation, the algorithm that computes the pseudopolar Fourier transform of an image I is given by Algorithms 4.1 and 4.2.

In words, Algorithm 4.1 computes Res_1 as follows.

(i) Zero-pad both ends of the y-direction of the image I (to size m) and compute the 2D DFT of the padded image. The result is stored in $\widehat{I}_d$.
(ii) Apply the 1D inverse Fourier transform on each row of $\widehat{I}_d$.
(iii) Resample each row k in the resulting array using the 1D fractional Fourier transform with $\alpha = 2k/n$.
(iv) Flip each row around its center.

The next theorem proves that Algorithms 4.1 and 4.2 indeed compute the pseudopolar Fourier transform.

Theorem 4.3.4 (correctness of Algorithms 4.1 and 4.2 [5]). *Upon termination of Algorithms 4.1 and 4.2, the following hold:*

$$\begin{aligned} \mathrm{Res}_1(k,l) &= \widehat{I}_{\Omega_{\mathrm{pp}}^1}(k,l), \\ \mathrm{Res}_2(k,l) &= \widehat{I}_{\Omega_{\mathrm{pp}}^2}(k,l), \end{aligned} \tag{4.22}$$

where $k = -n, \ldots, n$, $l = -n/2, \ldots, n/2$, *and* $\widehat{I}_{\Omega_{\mathrm{pp}}^1}$ *and* $\widehat{I}_{\Omega_{\mathrm{pp}}^2}$ *are given by* (4.17).

Next, we analyze the complexity of Algorithm 4.1. Step (2) can be executed in $O(n^2 \log n)$ operations by using successive applications of 1D FFT. Each call to $G_{k,n}$ in step (5) involves the application of 1D inverse Fourier transform ($O(n \log n)$ operations) followed by the computation of the fractional Fourier transform ($O(n \log n)$ operations). Thus, the computation of $G_{k,n}$ requires $O(n \log n)$ operations. Step (5) computes $G_{k,n}$ for each row k ($2n + 1$ rows), and thus, step (5) requires a total of $O(n^2 \log n)$ operations. Step (6) involves flipping $2n + 1$ vectors

Input: image I of size $n \times n$
Output: array Res_1 with $n+1$ rows and $m = 2n+1$ columns that contains the samples of $\hat{I}_{\Omega^1_{\text{pp}}}$
(1) $m \leftarrow 2n+1$
(2) $\hat{I}_d \leftarrow F_2(E(I,m,n))$
(3) **for** $k = -n,\ldots,n$ **do**
(4) $\quad q \leftarrow \hat{I}_d(\cdot,k)$
(5) $\quad w_k \leftarrow G_{k,n}q, \qquad w_k \in \mathbb{C}^{n+1}$
(6) $\quad \text{Res}_1(k,l) \leftarrow w_k(-l)$
(7) **end for**

Algorithm 4.1. Computing the pseudopolar Fourier transform $\hat{I}_{\Omega^1_{\text{pp}}}$ (4.17).

Input: image I of size $n \times n$
Output: array Res_2 with $n+1$ rows and $m = 2n+1$ columns that contains the samples of $\hat{I}_{\Omega^2_{\text{pp}}}$
(1) $m \leftarrow 2n+1$
(2) $\hat{I}_d \leftarrow F_2(E(I,n,m))$
(3) **for** $k = -n,\ldots,n$ **do**
(4) $\quad z \leftarrow \hat{I}_d(k,\cdot)$
(5) $\quad w_k \leftarrow G_{k,n}z, \qquad w_k \in \mathbb{C}^{n+1}$
(6) $\quad \text{Res}_2(k,l) \leftarrow w_k(-l)$
(7) **end for**

Algorithm 4.2. Computing the pseudopolar Fourier transform $\hat{I}_{\Omega^2_{\text{pp}}}$ (4.17).

of length $n+1$. Flipping a single vector requires $O(n)$ operations. This gives a total of $O(n^2)$ operations for flipping all $2n+1$ vectors. Thus, the total complexity of Algorithm 4.1 is $O(n^2 \log n)$ operations. The complexity of Algorithm 4.2 is the same. Thus, given an image I of size $n \times n$, its pseudopolar Fourier transform can be computed in $O(n^2 \log n)$ operations.

4.3.3. Invertibility

The 2D pseudopolar Fourier transform is invertible; see [5, 18].

4.3.3.1. Iterative inverse algorithm

We want to solve

$$\mathcal{F}_{\text{pp}}x = y, \tag{4.23}$$

where $\mathcal{F}_{\text{pp}}$ is the 2D pseudopolar Fourier transform, given by (4.18). Since y is not necessarily in the range of the pseudopolar Fourier transform, due to, for example, roundoff errors, we would like to solve

$$\min_{x \in R(\mathcal{F}_{\text{pp}})} ||\mathcal{F}_{\text{pp}}x - y||_2 \tag{4.24}$$

instead of (4.23), where $R(\mathcal{F}_{\mathrm{pp}})$ is the range of the pseudopolar Fourier transform. Solving (4.24) is equivalent to solving the normal equations

$$\mathcal{F}_{\mathrm{pp}}{}^{*}\mathcal{F}_{\mathrm{pp}}x = \mathcal{F}_{\mathrm{pp}}{}^{*}y, \tag{4.25}$$

where $\mathcal{F}_{\mathrm{pp}}^{*}$ is the adjoint pseudopolar Fourier transform. Since $\mathcal{F}_{\mathrm{pp}}{}^{*}\mathcal{F}_{\mathrm{pp}}$ is symmetric and positive definite, we can use the conjugate-gradient method [31] to solve (4.25). When using the conjugate-gradient method, we never explicitly form the matrices that correspond to $\mathcal{F}_{\mathrm{pp}}$ and $\mathcal{F}_{\mathrm{pp}}{}^{*}$ (which are huge), since only the application of the pseudopolar Fourier transform and its adjoint are required.

The number of iterations required by the conjugate-gradient method depends on the condition number of the transform. To accelerate the convergence, we construct a preconditioner M and solve

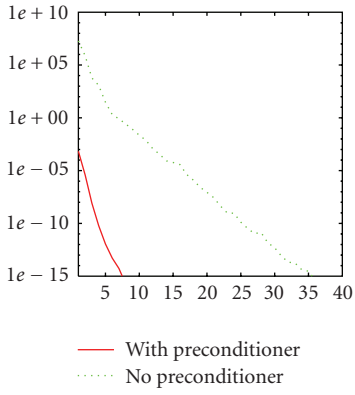
$$\mathcal{F}_{\mathrm{pp}}{}^{*}M\mathcal{F}_{\mathrm{pp}}x = M\mathcal{F}_{\mathrm{pp}}{}^{*}y. \tag{4.26}$$

M is usually designed such that the condition number of the operator $\mathcal{F}_{\mathrm{pp}}{}^{*}M\mathcal{F}_{\mathrm{pp}}$ is much smaller than the condition number of $\mathcal{F}_{\mathrm{pp}}{}^{*}\mathcal{F}_{\mathrm{pp}}$, or, such that the eigenvalues of $\mathcal{F}_{\mathrm{pp}}{}^{*}M\mathcal{F}_{\mathrm{pp}}$ are well clustered. We define the preconditioner M to be

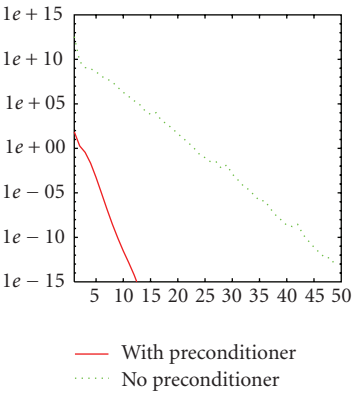
$$M(k,l) = \begin{cases} \dfrac{1}{m^2}, & k = 0, \\ \dfrac{2(n+1)|k|}{nm} & \text{otherwise}, \end{cases} \tag{4.27}$$

where $k = -n,\ldots,n$, $l = -n/2,\ldots,n/2$ and $m = 2n+1$. The preconditioner is applied to each of the pseudopolar sectors, where each pseudopolar sector is of size $m \times (n+1)$. The efficiency of the preconditioner M (4.27) is demonstrated in Figure 4.3. Each graph presents the residual error as a function of the iteration number. In Figure 4.3(a), the original image is a 2D Gaussian of size 512×512 with $\mu_x = \mu_y = 0$ and $\sigma_x = \sigma_y = 512/6$. In Figure 4.3(b), the original image is a random image of size 512×512, where the entries are uniformly distributed between 0 and 1. In Figure 4.3(d), the original image is Barbara of size 512×512, shown in Figure 4.3(c). As we can see from Figures 4.3(a)–4.3(d), the suggested preconditioner significantly accelerates the convergence. Without the preconditioner, the conjugate-gradient algorithm converges slowly. With the preconditioner, only a few iterations are required, and the number of iterations weakly depends on the reconstructed image.

As we can see from the tables in [5], very few iterations are required to invert the pseudopolar Fourier transform with high accuracy. The total complexity of the algorithm that inverts the pseudopolar Fourier transform is $O(\rho(\varepsilon)n^2 \log n)$, where $\rho(\varepsilon)$ is the number of iterations required to achieve accuracy ε. The value of $\rho(\varepsilon)$ depends very weakly on the size of the reconstructed image, and in any case $\rho(10^{-7}) \leq 10$.

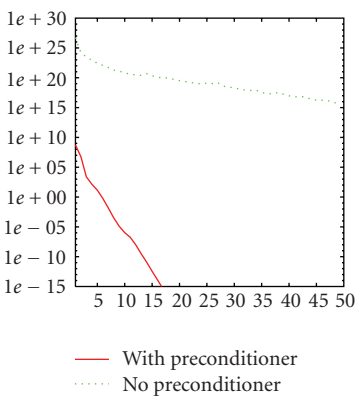


(a) Reconstruction of a random image 512×512



(b) Reconstruction of a Gaussian image 512×512

(c) Barbara 512×512



(d) Reconstruction of Barbara 512×512

FIGURE 4.3. The effect of using the preconditioner in (4.27).

4.3.3.2. Direct inverse algorithm

In this section, we describe a direct inversion algorithm for the pseudopolar Fourier transform. The algorithm consists of two phases. The first phase resamples the pseudopolar Fourier transform into a Cartesian frequency grid. The second phase recovers the image from these Cartesian frequency samples. Resampling from the pseudopolar to a Cartesian frequency grid is based on an "onion-peeling" procedure, which recovers a single row/column of the Cartesian grid in each iteration, from the outermost row/column to the origin. Recovering each row/column is based on a fast algorithm that resamples trigonometric polynomials from one set on frequencies to another set of frequencies (for more detail see [5, 18]).

TABLE 4.1. Inverting the pseudopolar Fourier transform of a Gaussian image with $\varepsilon = 10^{-7}$. The first column corresponds to n, where $n \times n$ is the size of the original image. The second and third columns correspond to the E_2 and E_∞ reconstruction errors, respectively. The fourth column, denoted by t_F, corresponds to the time (in seconds) required to compute the forward pseudopolar Fourier transform. The fifth column, denoted by t_I, corresponds to the time (in seconds) required to invert the pseudopolar Fourier transform using the proposed algorithm.

n	E_2	E_∞	t_F	t_I
8	8.85306e-16	7.75742e-16	1.68498e-01	3.53113e-01
16	6.33498e-16	7.78284e-16	9.19520e-02	9.89350e-02
32	1.07588e-15	1.42958e-15	1.67678e-01	1.74619e-01
64	8.62082e-15	6.83852e-15	3.32671e-01	5.65795e-01
128	1.15638e-14	7.68190e-15	9.46146e-01	2.62962e+00
256	6.81762e-15	4.07823e-15	2.68635e+00	1.86460e+01
512	3.83615e-14	2.52678e-14	1.02859e+01	1.33362e+02

4.3.3.3. Numerical results

The algorithm was implemented in Matlab and was applied to two test images of various sizes. The first image is a Gaussian image of size $n \times n$ with mean $\mu_x = \mu_y = 0$ and standard deviation $\sigma_x = \sigma_y = n/6$. The second test image is a random image whose entries are uniformly distributed in $[0, 1]$. In all tests we choose $\varepsilon = 10^{-7}$. For each test image we compute its pseudopolar Fourier transform followed by the application of the inverse pseudopolar Fourier transform. The reconstructed image is then compared to the original image. The results are summarized in Tables 4.1 and 4.2. Table 4.1 presents the results of inverting the pseudopolar Fourier transform of a Gaussian image. Table 4.2 presents the results of inverting the pseudopolar Fourier transform of a random image whose entries are uniformly distributed in $[0, 1]$.

As we can see from Tables 4.1 and 4.2, the inversion algorithm achieves very high accuracy for both types of images, while its execution time is comparable with the execution time of the optimized pseudopolar Fourier transform. The obtained accuracy is higher than the prescribed accuracy ε.

4.4. 2D discrete Radon transform

The importance of reconstruction of a cross section of an object from several images of its projections was explained in Section 4.1.1. A projection is a shadowgram obtained by illuminating an object with a penetrating radiation. Figure 4.1(a) shows a typical method for obtaining projections. Each horizontal line in Figure 4.1(a) is a one-dimensional projection of a horizontal slice of the object. Each pixel of the projected image represents the total absorption of X-ray along the ray's path from the source to the detector. By rotating the source-detector assembly around the object, projections for different angles are obtained. The goal of *image reconstruction from projections* is to obtain an image of a cross section of the object from these projections. Imaging systems that generate such slice views are called CT (computerized tomography) scanners.

TABLE 4.2. Inverting the pseudopolar Fourier transform of a random image with $\varepsilon = 10^{-7}$. The first column corresponds to n, where $n \times n$ is the size of the original image. The second and third columns correspond to the E_2 and E_∞ reconstruction errors, respectively. The fourth column, denoted by t_F, corresponds to the time (in seconds) required to compute the forward pseudopolar Fourier transform. The fifth column, denoted by t_I, corresponds to the time (in seconds) required to invert the pseudopolar Fourier transform using the proposed algorithm.

n	E_2	E_∞	t_F	t_I
8	1.12371e-15	1.40236e-15	3.03960e-02	4.83120e-02
16	1.54226e-15	1.98263e-15	6.56960e-02	8.04270e-02
32	4.68305e-15	8.27006e-15	1.50652e-01	1.85239e-01
64	1.56620e-14	2.50608e-14	3.19637e-01	5.45564e-01
128	3.56283e-14	6.96984e-14	9.66602e-01	2.57069e+00
256	7.45050e-14	1.59613e-13	2.78531e+00	1.69190e+01
512	3.15213e-13	6.38815e-13	9.19018e+00	1.30085e+02

The Radon transform is the underlying mathematical tool used for CT scanning, as well as for a wide range of other disciplines, including radar imaging, geophysical imaging, nondestructive testing, and medical imaging [1].

For the 2D case, the Radon transform of a function $f(x, y)$, denoted as $\Re f(\theta, s)$, is defined as the line integral of f along a line L inclined at an angle θ and at distance s from the origin (see Figure 4.1(b)). Formally, it is described by (4.1) and it is illustrated in Figure 4.1(c). There is a fundamental relationship between the 2D Fourier transform of a function and the 1D Fourier transform of its Radon transform. The result is summarized in Theorem 4.6.1.

For modern applications it is important to have a discrete analog of $\Re f$ for 2D digital images $I = (I(u, v) : u, v = -n/2, \ldots, n/2 - 1)$. The proposed notion of digital Radon transform satisfies the requirements in Section 4.1.2.

4.4.1. Definition

The discrete Radon transform is defined by summing the samples of $I(u, v)$, $u, v = -n/2, \ldots, n/2-1$, along lines. The two key issues of the construction are the following. How to process lines of the discrete transform when they do not pass through grid points? How to choose the set of lines which we sum, in order to achieve geometric fidelity, rapid computation, and invertibility?

The discrete Radon transform is defined by summing the discrete samples of the image $I(u, v)$ along lines with absolute slope less than 1. For lines of the form $y = sx + t$ ($|s| \leq 1$), we traverse each line by unit horizontal steps $x = -n/2, \ldots, n/2 - 1$, and for each x, we interpolate the image sample at position (x, y) by using trigonometric interpolation along the corresponding image column. For lines of the form $y = sx + t$ ($|s| \geq 1$), we rephrase the line equation as $x = s'y + t'$ ($|s'| \leq 1$). In this case, we traverse the line by unit vertical steps, and for each integer y, we interpolate the value at the x coordinate $x = s'y + t'$ by using trigonometric interpolation along the corresponding row. The requirement for slopes less than 1 induces two families of lines.

(i) *Basically horizontal line* is a line of the form $y = sx + t$, where $|s| \leq 1$.
(ii) *Basically vertical line* is a line of the form $x = sy + t$, where $|s| \leq 1$.

Using these line families we define the 2D Radon transform for discrete images as the following.

Definition 4.4.1 (2D Radon transform for discrete images). Let $I(u, v)$, $u, v = -n/2, \ldots, n/2 - 1$, be an $n \times n$ array. Let s be a slope such that $|s| \leq 1$, and let t be an intercept such that $t = -n, \ldots, n$. Then,

$$\begin{aligned} \text{Radon}(\{y = sx + t\}, I) &= \sum_{u=-n/2}^{n/2-1} \widetilde{I}^1(u, su + t), \\ \text{Radon}(\{x = sy + t\}, I) &= \sum_{v=-n/2}^{n/2-1} \widetilde{I}^2(sv + t, v), \end{aligned} \tag{4.28}$$

where

$$\begin{aligned} \widetilde{I}^1(u, y) &= \sum_{v=-n/2}^{n/2-1} I(u, v) D_m(y - v), \quad u = \frac{-n}{2}, \ldots, \frac{n}{2-1}, \; y \in \mathbb{R}, \\ \widetilde{I}^2(x, v) &= \sum_{u=-n/2}^{n/2-1} I(u, v) D_m(x - u), \quad v = \frac{-n}{2}, \ldots, \frac{n}{2-1}, \; x \in \mathbb{R}, \end{aligned} \tag{4.29}$$

$$D_m(t) = \frac{\sin(\pi t)}{m \sin(\pi t/m)}, \quad m = 2n + 1. \tag{4.30}$$

Moreover, for an arbitrary line l with slope s and intercept t, such that $|s| \leq 1$ and $t = -n, \ldots, n$, the Radon transform is given by

$$(RI)(s, t) = \begin{cases} \text{Radon}(\{y = sx + t\}, I), & l \text{ is a basically horizontal line,} \\ \text{Radon}(\{x = sy + t\}, I), & l \text{ is a basically vertical line.} \end{cases} \tag{4.31}$$

$\widetilde{I}^1$ and $\widetilde{I}^2$ in (4.29) are column-wise and row-wise interpolated versions of I, respectively, using the *Dirichlet kernel* D_m with $m = 2n + 1$.

The selection of the parameters s and t in Definition 4.4.1 is explained in [6].

4.4.2. Representation of lines using angle

In order to derive the 2D Radon transform (Definition 4.4.1) in a more natural way, we rephrase it using angles instead of slopes. For a basically horizontal line $y = sx+t$ with $|s| \leq 1$, we express s as $s = \tan\theta$ with $\theta \in [-\pi/4, \pi/4]$, where θ is the angle between the line and the positive direction of the x-axis. Using this notation, a basically horizontal line has the form $y = (\tan\theta)x + t$ with $\theta \in [-\pi/4, \pi/4]$. Given a basically vertical line $x = sy + t$ with $|s| \leq 1$, we express s as $s = \cot\theta$ with $\theta \in [\pi/4, 3\pi/4]$, where θ is again the angle between the line and the positive

direction of the x-axis. Hence, a basically vertical line has the form $x = (\cot\theta)y + t\theta \in [\pi/4, 3\pi/4]$.

Using this parametric representation we rephrase the definition of the Radon transform (Definition 4.4.1) as the following.

Definition 4.4.2. Let $I(u,v)$, $u,v = -n/2,\ldots,n/2-1$, be a $n\times n$ array. Let $\theta \in [-\pi/4, 3\pi/4]$, and let t be an intercept such that $t = -n,\ldots,n$. Then,

$$(RI)(\theta,t) = \begin{cases} \text{Radon}\big(\{y = (\tan\theta)x + t\}, I\big), & \theta \in \left[\dfrac{-\pi}{4}, \dfrac{\pi}{4}\right], \\ \text{Radon}\big(\{x = (\cot\theta)y + t\}, I\big), & \theta \in \left[\dfrac{\pi}{4}, \dfrac{3\pi}{4}\right], \end{cases} \tag{4.32}$$

where

$$\begin{aligned} \text{Radon}\big(\{y = sx + t\}, I\big) &= \sum_{u=-n/2}^{n/2-1} \widetilde{I}^{1}(u, su + t), \\ \text{Radon}\big(\{x = sy + t\}, I\big) &= \sum_{v=-n/2}^{n/2-1} \widetilde{I}^{2}(sv + t, v), \end{aligned} \tag{4.33}$$

and $\widetilde{I}^{1}$, $\widetilde{I}^{2}$, and D_m are given by (4.29), (4.30), respectively.

The Radon transform in Definition 4.4.2 operates on discrete images $I(u,v)$ while θ is continuous. We have to discretize the parameter θ in order to have a "discrete Radon transform" that satisfies properties in Section 4.1.2.

4.4.3. Fourier slice theorem

The Fourier slice theorem for the 2D continuous Radon transform defines a relation between the continuous Radon transform $\Re f$ of a function $f(x,y)$ and the 2D Fourier transform of f along some radial line:

$$\widehat{\Re f}(\theta,\xi) = \hat{f}(\xi\cos\theta, \xi\sin\theta), \tag{4.34}$$

where $\widehat{\Re f}$ is the 1D Fourier transform of $\Re f$ (4.2). Equation (4.34) allows us to evaluate the continuous Radon transform using the 2D Fourier transform of the function f. We are looking for a similar relation between the discrete Radon transform and the discrete Fourier transform of the image. We then use such a relation to compute the discrete Radon transform.

To construct such a relation we define some auxiliary operators which enable us to rephrase the definition of the discrete Radon transform (Definition 4.4.2) in a more convenient way. Throughout this section we denote by $\mathbb{C}^m$ the set of complex-valued vectors of length m, indexed from $-\lfloor m/2\rfloor$ to $\lfloor (m-1)/2\rfloor$. Also, we denote by $\mathbb{C}^{k\times l}$ the set of 2D complex-valued images of dimensions $k\times l$. Each

image $I \in \mathbb{C}^{k\times l}$ is indexed as $I(u,v)$, where $u = -\lfloor k/2\rfloor,\ldots,\lfloor (k-1)/2\rfloor$ and $v = -\lfloor l/2\rfloor,\ldots,\lfloor (l-1)/2\rfloor$. k is along the x-axis and l is along the y-axis.

Definition 4.4.3 (translation operator). Let $\alpha \in \mathbb{C}^m$ and $\tau \in \mathbb{R}$. The *translation* operator $T_\tau : \mathbb{C}^m \to \mathbb{C}^m$ is given by

$$\left(T_\tau \alpha\right)_u = \sum_{i=-n}^{n} \alpha_i D_m(u-i-\tau), \quad m = 2n+1, \tag{4.35}$$

where $u = -n,\ldots,n$ and D_m is given by (4.30).

The translation operator T_τ takes a vector of length m and translates it by τ by using trigonometric interpolation .

Lemma 4.4.4 (see [6]). *Let T_τ be the translation operator given by Definition 4.4.3. Then,*

$$\operatorname{adj} T_\tau = T_{-\tau}. \tag{4.36}$$

An important property of the translation operator T_τ is that the translation of exponentials is algebraically exact. In other words, translating a vector of samples of the exponential $e^{2\pi ikx/m}$ is the same as resampling the exponential at the translated points. This observation is of great importance for proving the Fourier slice theorem.

Lemma 4.4.5 (see [6]). *Let $m = 2n+1$, $\varphi(x) = e^{2\pi ikx/m}$. Define the vector $\phi \in \mathbb{C}^m$ as $\phi = (\varphi(t) : t = -n,\ldots,n)$. Then, for arbitrary $\tau \in \mathbb{R}$,*

$$\left(T_\tau \phi\right)_t = \varphi(t-\tau), \quad t = -n,\ldots,n. \tag{4.37}$$

Definition 4.4.6. The *extension* operators $E^1 : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times m}$ and $E^2 : \mathbb{C}^{n\times n} \to \mathbb{C}^{m\times n}$ are given by

$$E^i I(u,v) = \begin{cases} I(u,v), & u,v = \dfrac{-n}{2},\ldots,\dfrac{n}{2-1}, \\ 0 & \text{otherwise}, \end{cases} \tag{4.38}$$

where $i = 1,2$, and $m = 2n+1$.

E^1 is an operator that takes an array of size $n\times n$ and produces an array of size $(2n+1)\times n$ ($2n+1$ rows and n columns) by adding $n/2+1$ zero rows at the top of the array and $n/2$ zero rows at the bottom of the array (see Figure 4.4). Similarly, E^2 corresponds to padding the array I with $n+1$ zero columns, $n/2+1$ at the right and $n/2$ at the left.

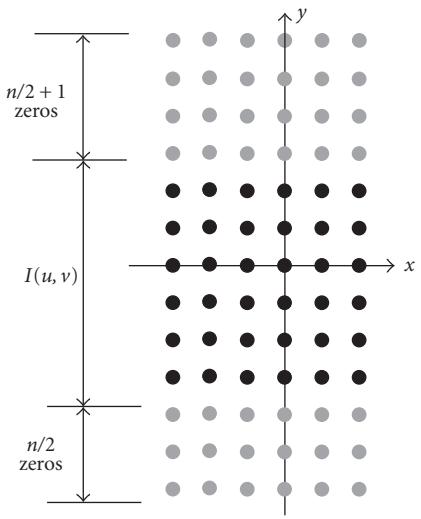


FIGURE 4.4. Applying the padding operator E^1 to an array of size $n \times n$.

Definition 4.4.7. The *truncation* operators $U^1 : \mathbb{C}^{n\times m} \to \mathbb{C}^{n\times n}$ and $U^2 : \mathbb{C}^{m\times n} \to \mathbb{C}^{n\times n}$ are given by

$$U^i I(u,v) = I(u,v), \quad u,v = \frac{-n}{2},\ldots,\frac{n}{2-1}, \tag{4.39}$$

where $i = 1,2$ and $m = 2n+1$.

The operator U^1 removes $n/2+1$ rows from the top of the array and $n/2$ rows from the bottom of the array, recovering an $n \times n$ image. Similarly, U^2 removes $n/2+1$ columns from the right and $n/2$ columns from the left of the array.

Lemma 4.4.8 (see [6]). *Let E^1 and U^1 be the extension and truncation operators, given by Definitions 4.4.6 and 4.4.7, respectively. Then,*

$$\operatorname{adj} E^1 = U^1. \tag{4.40}$$

In an exactly analogous way we show that $U^2 = \operatorname{adj} E^2$.

Definition 4.4.9. Let $I(u,v)$, $u,v = -n/2,\ldots,n/2-1$, be an $n \times n$ image. The *shearing operators* $S^1 : \mathbb{C}^{n\times m} \to \mathbb{C}^{n\times m}$ and $S^2 : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$, $m = 2n+1$, are given by

$$\begin{aligned} \left(S^1_\theta I\right)(u,v) &= \left(T_{-u\tan\theta} I(u,\cdot)\right)_v, \quad \theta \in \left[\frac{-\pi}{4},\frac{\pi}{4}\right], \\ \left(S^2_\theta I\right)(u,v) &= \left(T_{-v\cot\theta} I(\cdot,v)\right)_u, \quad \theta \in \left[\frac{\pi}{4},\frac{3\pi}{4}\right]. \end{aligned} \tag{4.41}$$

Lemma 4.4.10 (see [6]). *For $t = -n, \ldots, n$,*

$$RI(\theta, t) = \begin{cases} \sum_{u=-n/2}^{n/2-1} (S_\theta^1(E^1 I))(u, t), & \theta \in \left[\frac{-\pi}{4}, \frac{\pi}{4}\right], \\ \sum_{v=-n/2}^{n/2-1} (S_\theta^2(E^2 I))(t, v), & \theta \in \left[\frac{\pi}{4}, \frac{3\pi}{4}\right]. \end{cases} \tag{4.42}$$

Definition 4.4.11. Let $\psi \in \mathbb{C}^m$ and $m = 2n + 1$. The backprojection operators $B_\theta^1 : \mathbb{C}^m \to \mathbb{C}^{n \times m}$ and $B_\theta^2 : \mathbb{C}^m \to \mathbb{C}^{m \times n}$ are given by

$$\begin{aligned} (B_\theta^1 \psi)(u, \cdot) &= T_{u \tan\theta} \psi, \quad \theta \in \left[\frac{-\pi}{4}, \frac{\pi}{4}\right], \\ (B_\theta^2 \psi)(\cdot, v) &= T_{v \cot\theta} \psi, \quad \theta \in \left[\frac{\pi}{4}, \frac{3\pi}{4}\right]. \end{aligned} \tag{4.43}$$

Lemma 4.4.12 (see [6]).

$$\begin{aligned} \operatorname{adj} B_\theta^1 &= \sum_u S_\theta^1, \\ \operatorname{adj} B_\theta^2 &= \sum_v S_\theta^2. \end{aligned} \tag{4.44}$$

We can now give an explicit formula for the adjoint Radon transform.

Theorem 4.4.13 (see [6]). *The adjoint Radon transform* $\operatorname{adj} R_\theta : \mathbb{C}^m \to \mathbb{C}^{n \times n}$ *is given by*

$$\operatorname{adj} R_\theta = \begin{cases} U^1 \circ B_\theta^1, & \theta \in \left[\frac{-\pi}{4}, \frac{\pi}{4}\right], \\ U^2 \circ B_\theta^2, & \theta \in \left[\frac{\pi}{4}, \frac{3\pi}{4}\right], \end{cases} \tag{4.45}$$

where U^i, $i = 1, 2$, is given by Definition 4.4.7 and B_θ^i, $i = 1, 2$, is given by Definition 4.4.11.

We next examine how the adjoint Radon transform operates on the vector $\phi^{(k)} = (\varphi^{(k)}(t) : t = -n, \ldots, n)$, where $\varphi^{(k)}(t) = e^{(2\pi\imath/m)kt}$. For $\theta \in [-\pi/4, \pi/4]$ and $u, v = -n/2, \ldots, n/2 - 1$.

Theorem 4.4.14 (Fourier slice theorem). *Let $I(u, v)$ be an $n \times n$ image, $u, v = -n/2, \ldots, n/2 - 1$, and let $m = 2n + 1$. Then,*

$$(\widehat{R_\theta I})(k) = \begin{cases} \widehat{I}(-s_1 k, k), & s_1 = \tan\theta,\ \theta \in \left[\frac{-\pi}{4}, \frac{\pi}{4}\right], \\ \widehat{I}(k, -s_2 k), & s_2 = \cot\theta,\ \theta \in \left[\frac{\pi}{4}, \frac{3\pi}{4}\right], \end{cases} \tag{4.46}$$

where $\widehat{R_\theta I}(k)$ *is the 1D DFT of the discrete Radon transform, given by Definition 4.4.2, with respect to the parameter* t, $k = -n, \ldots, n$, *and* $\hat{I}$ *is the trigonometric polynomial*

$$\hat{I}(\xi_1, \xi_2) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} I(u, v) e^{-(2\pi\imath/m)(\xi_1 u + \xi_2 v)}. \tag{4.47}$$

4.4.4. Discretization and fast algorithms

The Radon transform, given by Definition 4.4.2, and the Fourier slice theorem, given by Theorem 4.4.14, were defined for discrete images and a continuous set of lines. Specifically, $R_\theta I(t)$ operates on any angle in the range $[-\pi/4, 3\pi/4]$. For our transform to be fully discrete, we must discretize the set of angles. We denote such a discrete set of angles by Θ. By using the discrete set of intercepts

$$T \triangleq \{-n, \ldots, n\}, \tag{4.48}$$

we define the discrete Radon transform as

$$RI = \{R_\theta I(t) \mid \theta \in \Theta,\ t \in T\}. \tag{4.49}$$

We define Θ to be the set of angles induced by lines with equally spaced slopes. Specifically, we define two sets of angles:

$$\Theta_2 = \left\{ \arctan\left(\frac{2l}{n}\right) \mid l = \frac{-n}{2}, \ldots, \frac{n}{2} \right\}, \tag{4.50}$$

$$\Theta_1 = \left\{ \frac{\pi}{2} - \arctan\left(\frac{2l}{n}\right) \mid l = \frac{-n}{2}, \ldots, \frac{n}{2} \right\}. \tag{4.51}$$

If we take an arbitrary element from Θ_2, that is, $\theta_2^l = \arctan(2l/n)$, then, the slope of the line corresponding to θ_2^l is $s = \tan\theta_2^l = 2l/n$. By inspecting two elements θ_2^l and θ_2^{l+1} we can see that the difference between the slopes that correspond to the two angles is

$$s_2 - s_1 = \tan\theta_2^{l+1} - \tan\theta_2^l = \frac{2(l+1)}{n} - \frac{2l}{n} = \frac{2}{n}, \tag{4.52}$$

which means that our angles define a set of equally spaced slopes (see Figures 4.2(a) and 4.2(b)). Figure 4.2(a) is the slopes that correspond to angles in Θ_2 (4.50) and Figure 4.2(c) is the slopes that correspond to angles in Θ_1 (4.51).

We define the set of angles Θ to be

$$\Theta \triangleq \Theta_1 \cup \Theta_2, \tag{4.53}$$

where Θ_1 and Θ_2 are given by (4.51) and (4.50), respectively. Using the set Θ we define the discrete Radon transform for discrete images and a discrete set of parameters.

Definition 4.4.15 (2D discrete Radon transform). Let $\theta \in \Theta$, $t \in T$, where Θ and T are given by (4.53) and (4.48), respectively. Then,

$$R_\theta I(t) \triangleq \begin{cases} \text{Radon } (y = (\tan\theta)x + t), & \theta \in \Theta_2, \\ \text{Radon } (x = (\cot\theta)y + t), & \theta \in \Theta_1. \end{cases} \tag{4.54}$$

Definition 4.4.15 defines a transform that takes an image I of size $n \times n$ into an array (θ, t) of size $2 \times (2n+1) \times (n+1)$.

The Fourier slice theorem (Theorem 4.4.14) holds also for the discrete set Θ. For $\theta \in \Theta^2$ (4.50) we have from Theorem 4.4.14 that $\widehat{R_\theta I}(k) = \hat{I}(-s_1 k, k)$, where $s_1 = \tan\theta$. Since $\theta \in \Theta^2$ has the form $\theta = \arctan(2l/n)$, it follows that $s_1 = \tan(\arctan(2l/n)) = 2l/n$ and

$$\widehat{R_\theta I}(k) = \hat{I}\Big(-\frac{2l}{n}k, k\Big), \quad k = -n, \ldots, n. \tag{4.55}$$

For $\theta \in \Theta^1$ (4.51), we have from Theorem 4.4.14 that $\widehat{R_\theta I}(k) = \hat{I}(k, -s_2 k)$, where $s_2 = \cot\theta$. Since $\theta \in \Theta^1$ has the form $\theta = \pi/2 - \arctan(2l/n)$, it follows that $s_2 = \cot(\pi/2 - \arctan(2l/n)) = \tan(\arctan(2l/n)) = 2l/n$ and

$$\widehat{R_\theta I}(k) = \hat{I}\Big(k, -\frac{2l}{n}k\Big), \quad k = -n, \ldots, n. \tag{4.56}$$

Equations (4.55) and (4.56) show that $R_\theta I$ is obtained by resampling the trigonometric polynomial $\hat{I}$, given by (4.8), on the pseudopolar grid Ω_{pp}, defined in Section 4.3.1. Specifically,

$$\begin{aligned} \widehat{R_\theta I}(k) &= \hat{I}_{\Omega^1_{\text{pp}}}(k,l), \quad \theta \in \Theta^2, \\ \widehat{R_\theta I}(k) &= \hat{I}_{\Omega^2_{\text{pp}}}(k,l), \quad \theta \in \Theta^1, \end{aligned} \tag{4.57}$$

where $\hat{I}_{\Omega^1_{\text{pp}}}$ and $\hat{I}_{\Omega^2_{\text{pp}}}$ are given by (4.17). From (4.57) we derive the relation

$$\begin{aligned} R_\theta I &= F_k^{-1} \circ \hat{I}_{\Omega^1_{\text{pp}}}, \quad \theta \in \Theta^2, \\ R_\theta I &= F_k^{-1} \circ \hat{I}_{\Omega^2_{\text{pp}}}, \quad \theta \in \Theta^1, \end{aligned} \tag{4.58}$$

where F_k^{-1} is the 1D inverse Fourier transform along each row of the array $\hat{I}_{\Omega^s_{\text{pp}}}$, $s = 1, 2$ (along the parameter k).

We denote by RI the array that corresponds to the discrete Radon transform of an image I of size $n \times n$. By using (4.58), we define RI as

$$RI = \{R_\theta I \mid \theta \in \Theta\}, \tag{4.59}$$

where Θ is given by (4.53). The fast algorithm that computes the pseudopolar Fourier transform, given in Section 4.3.2, immediately gives an algorithm for computing the discrete Radon transform. To see this, consider a row in RI, which corresponds to a constant θ. The values of the discrete Radon transform that correspond to θ are computed by applying F_k^{-1} to a row of $\widehat{I}_{\Omega^s_{\text{pp}}}$, $s = 1, 2$. Thus, the computation of RI requires applying F_k^{-1} to all rows of $\widehat{I}_{\Omega^s_{\text{pp}}}$, $s = 1, 2$ ($2n + 2$ rows). Since each application of F_k^{-1} operates on a vector of length $2n + 1$ (a row of $\widehat{I}_{\Omega^1_{\text{pp}}}$ or $\widehat{I}_{\Omega^2_{\text{pp}}}$), it requires $O(n \log n)$ operations for a single application, and a total of $O(n^2 \log n)$ operations for the required $2n + 2$ calls to F_k^{-1}. Thus, once we compute the arrays $\widehat{I}_{\Omega^s_{\text{pp}}}$, $s = 1, 2$, it requires $O(n^2 \log n)$ operations to compute RI. Since computing $\widehat{I}_{\Omega^s_{\text{pp}}}$, $s = 1, 2$, requires $O(n^2 \log n)$ operations, the total complexity of computing the values of RI, given by (4.59), is $O(n^2 \log n)$ operations.

Table 4.3 presents numerical results for the computation of the 2D discrete Radon transform.

Invertibility of the 2D discrete Radon transform RI (4.59) also follows from (4.58). F_k^{-1} is invertible and can be rapidly inverted by using the inverse fast Fourier transform. $\widehat{I}_{\Omega_{\text{pp}}}$, which is the union of $\widehat{I}_{\Omega^1_{\text{pp}}}$ and $\widehat{I}_{\Omega^2_{\text{pp}}}$ (4.17), is invertible (Section 4.3.3) and can be rapidly inverted as described in Sections 4.3.3.1 and 4.3.3.2. Thus, the discrete Radon transform is invertible, and can be inverted by applying the inverse FFT on each row of RI ($O(n^2 \log n)$ operations), followed by an inversion of $\widehat{I}_{\Omega_{\text{pp}}}$ ($O(n^2 \log n)$ operations). Hence, inverting the discrete Radon transform requires $O(n^2 \log n)$ operations.

4.4.4.1. Convergence

The discrete Radon transform converges to the continuous Radon transform as the number of samples in the discretization goes to infinity (see [6, 18, 19]).

4.5. 3D discrete Radon transform

Following the 2D construction of the discrete Radon transform, we give a definition for the 3D discrete Radon transform that satisfies, as in the 2D case, the properties in Section 4.1.2. This construction can generalize the construction and the properties of the 2D discrete Radon transform to 3D. Throughout the rest of the paper we refer to "3D image/array" as a 3D discrete volume of size $n \times n \times n$. This section is based on [8].

We first define a 3D Radon transform that is defined on discrete 3D arrays and on all planes in $\mathbb{R}^3$. Then we show that there exists a discrete set of planes for which the 3D discrete Radon transform is invertible and rapidly computable.

TABLE 4.3. Numerical results for the discrete Radon transform. Each test matrix is a random matrix with uniformly distributed elements in the range $[0, 1]$. The timings are given in seconds. Column 1 corresponds to n. Column 2 corresponds to the time required to compute the 2D discrete Radon transform of an $n \times n$ image using the Fourier slice theorem. Column 3 presents the time required to compute the 2D discrete Radon transform directly according to its definition (Definition 4.4.15). Column 4 presents the relative L^2 error between the output from the fast algorithm and the direct computation.

n	T_{fast}	T_{direct}	error
8	0.047	0.734	2.4922e-016
16	0.063	10.3	3.1364e-016
32	0.094	165	3.6785e-016
64	0.282	2.7e+003	4.5775e-016
128	1.91	3.9e+004	5.7779e-016
256	2.62	—	—
512	10.3	—	—
1024	43.5	—	—
2048	261	—	—

4.5.1. Definition of the 3D Radon transform

Inspired by the definition of the 2D discrete Radon transform, the 3D discrete Radon transform is defined by summing the interpolated samples of a discrete array $I(u, v, w)$ that lie on planes which satisfy certain constraints.

We define three types of planes, namely, "x-planes," "y-planes," and "z-planes."

Definition 4.5.1. (i) A plane of the form

$$x = s_1 y + s_2 z + t, \tag{4.60}$$

where

$$|s_1| \leq 1, \qquad |s_2| \leq 1, \tag{4.61}$$

is called an *x-plane*.

(ii) A plane of the form

$$y = s_1 x + s_2 z + t, \tag{4.62}$$

where

$$|s_1| \leq 1, \qquad |s_2| \leq 1, \tag{4.63}$$

is called a *y-plane*.

(iii) A plane of the form

$$z = s_1 x + s_2 y + t, \tag{4.64}$$

where

$$|s_1| \le 1, \qquad |s_2| \le 1, \tag{4.65}$$

is called a *z-plane*.

Lemma 4.5.2. *Each plane p in $\mathbb{R}^3$ can be expressed as one of the plane types from Definition 4.5.1 (x-plane, y-plane, or z-plane).*

We define three summation operators, one for each type of plane (x-plane, y-plane, z-plane defined in Definition 4.5.1). Each summation operator takes a plane and an image I and calculates the sum of the interpolated samples of I on the plane.

Definition 4.5.3 (summation operators). Let I be a discrete image of size $n \times n \times n$.

(i) For x-plane,

$$\text{Radon}\left(\{x = s_1 y + s_2 z + t\}, I\right) = \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} \widetilde{I}^1(s_1 v + s_2 w + t, v, w), \tag{4.66}$$

where

$$\widetilde{I}^1(x, v, w) = \sum_{u=-n/2}^{n/2-1} I(u, v, w) D_m(x - u), \quad v, w \in \left\{\frac{-n}{2}, \ldots, \frac{n}{2-1}\right\}, \; x \in \mathbb{R}. \tag{4.67}$$

(ii) For y-plane,

$$\text{Radon}\left(\{y = s_1 x + s_2 z + t\}, I\right) = \sum_{u=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} \widetilde{I}^2(u, s_1 u + s_2 w + t, w), \tag{4.68}$$

where

$$\widetilde{I}^2(u, y, w) = \sum_{v=-n/2}^{n/2-1} I(u, v, w) D_m(y - v), \quad u, w \in \left\{\frac{-n}{2}, \ldots, \frac{n}{2-1}\right\}, \; y \in \mathbb{R}. \tag{4.69}$$

(iii) For z-plane,

$$\text{Radon}\left(\{z = s_1 x + s_2 y + t\}, I\right) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} \widetilde{I}^3(u, v, s_1 u + s_2 v + t), \tag{4.70}$$

where

$$\widetilde{I}^{3}(u,v,z) = \sum_{w=-n/2}^{n/2-1} I(u,v,w)D_m(z-w), \quad u,v \in \left\{\frac{-n}{2},\ldots,\frac{n}{2-1}\right\},\ z \in \mathbb{R}. \tag{4.71}$$

In (4.66)–(4.70), t satisfies $t \in \mathbb{Z}$, $-3n/2 \le t \le 3n/2$ and D_m is defined by

$$D_m(t) = \frac{\sin(\pi t)}{m\sin(\pi t/m)}, \quad m = 3n+1. \tag{4.72}$$

The selection of t and m is critical and explained in [8].

Using the summation operators given in (4.66)–(4.70), we define three operators R_iI $(i = 1,2,3)$ for a 3D image (volume) I as follows.

(i) For x-plane $x = s_1y + s_2z + t$ define

$$R_1I(s_1,s_2,t) \triangleq \text{Radon}\,(x = s_1y + s_2z + t, I). \tag{4.73}$$

(ii) For y-plane $y = s_1x + s_2z + t$ define

$$R_2I(s_1,s_2,t) \triangleq \text{Radon}\,(y = s_1x + s_2z + t, I). \tag{4.74}$$

(iii) For z-plane $z = s_1x + s_2y + t$ define

$$R_3I(s_1,s_2,t) \triangleq \text{Radon}\,(z = s_1x + s_2y + t, I). \tag{4.75}$$

For a given plane p we can classify it by Lemma 4.5.2 as either an x-plane, y-plane, or z-plane with slopes s_1 and s_2. We define the "canonization transform" that takes an arbitrary plane p and transforms it into one of the plane types that were defined in Definition 4.5.1.

Definition 4.5.4 (canonization transform). Given a plane p whose equation is given by

$$p : \alpha x + \beta y + \gamma z + t = 0 \tag{4.76}$$

we denote by P the set of all planes p in $\mathbb{R}^3$. Let $C : P \to \mathbb{R}^4$ be the transformation that takes a plane $p \in P$ and transforms it into one of the plane types: x-plane, y-plane, or z-plane. For $p \in P$,

$$C(p) \triangleq (q, s_1, s_2, t'), \tag{4.77}$$

where

(i) $q = 1$ if $C(p)$ is an x-plane $x = s_1 y + s_2 z + t'$ with $-1 \le s_1, s_2 \le 1$,
(ii) $q = 2$ if $C(p)$ is a y-plane $y = s_1 x + s_2 z + t'$ with $-1 \le s_1, s_2 \le 1$,
(iii) $q = 3$ if $C(p)$ is a z-plane $z = s_1 x + s_2 y + t'$ with $-1 \le s_1, s_2 \le 1$.

Definition 4.5.5 (3D Radon transform). Assume that I is a discrete image of size $n \times n \times n$ and the plane p is determined by $C(p)$ from Definition 4.5.4. Then the 3D Radon transform of I on p is defined by

$$RI(p, I) = RI(s_1, s_2, t) \triangleq \begin{cases} R_1 I(s_1, s_2, t), & q = 1, \\ R_2 I(s_1, s_2, t), & q = 2, \\ R_3 I(s_1, s_2, t), & q = 3, \end{cases} \tag{4.78}$$

where R_1, R_2, and R_3 were defined in (4.73)–(4.75) and q, s_1, s_2, and t are determined from $C(p)$ according to Definition 4.5.4.

4.5.2. Traditional (ϕ, θ) representation of planes

The definition of the 3D Radon transform uses slopes and intercepts to designate a specific plane. The notation of a plane using slopes is less common and should be further explained. Usually, a plane in $\mathbb{R}^3$ is defined using a unit normal vector $\vec{n}$ and the distance of the plane from the origin. Formally, $\langle \vec{n}, (x, y, z) \rangle = t$ where $\vec{n}$ can be represented using the angles (ϕ, θ) as

$$\vec{n} = (\cos\phi \sin\theta, \sin\phi \sin\theta, \cos\theta). \tag{4.79}$$

We would like to find a correlation between the (ϕ, θ, t) representation of a plane and the explicit plane representation (using slopes).

We start by inspecting the explicit equation of a z-plane $z = s_1 x + s_2 y + t$ where $|s_1| \le 1$ and $|s_2| \le 1$. This can be rewritten as

$$\langle (-s_1, -s_2, 1), (x, y, z) \rangle = t. \tag{4.80}$$

We define

$$s = ||(-s_1, -s_2, 1)|| = \sqrt{s_1^2 + s_2^2 + 1} \tag{4.81}$$

and by normalizing the normal vector, we obtain

$$\left\langle \left(-\frac{s_1}{s}, -\frac{s_2}{s}, \frac{1}{s} \right), (x, y, z) \right\rangle = \frac{t}{s}, \tag{4.82}$$

where $\vec{n} = (-s_1/s, -s_2/s, 1/s)$ is the unit normal vector to the plane.

By expressing (ϕ, θ) of the vector $\vec{n}$ using s_1 and s_2 we obtain

$$\begin{aligned} \tan\phi &= \frac{-s_2/s}{-s_1/s} = \frac{s_2}{s_1}, \\ \tan\theta &= \pm\frac{\sqrt{s_1^2/s^2 + s_2^2/s^2}}{1/s} = \pm\sqrt{s_1^2 + s_2^2}. \end{aligned} \tag{4.83}$$

The range of (ϕ, θ), which corresponds to z-planes, is

$$\tan\phi = \frac{s_2}{s_1}, \qquad \tan\theta = \pm\sqrt{s_1^2 + s_2^2}, \tag{4.84}$$

where $|s_1| \leq 1$ and $|s_2| \leq 1$.

Resolving for s_1 and s_2, we obtain

$$\begin{aligned} s_1^2 &= \frac{\tan^2\theta}{1 + \tan^2\phi}, \\ s_2^2 &= \frac{\tan^2\theta\tan^2\phi}{1 + \tan^2\phi}. \end{aligned} \tag{4.85}$$

Since $|s_1| \leq 1$ and $|s_2| \leq 1$, it follows that

$$\begin{aligned} \frac{\tan^2\theta}{1 + \tan^2\phi} &\leq 1, \\ \frac{\tan^2\theta\tan^2\phi}{1 + \tan^2\phi} &\leq 1. \end{aligned} \tag{4.86}$$

We conclude that the range of (ϕ, θ), which defines all z-planes, satisfies (4.86). Equation (4.86) does not define a domain that is simple to describe and therefore it is less convenient than the slopes notation used to define our notion of 3D Radon transform (Definition 4.5.5).

Inspecting the relations between x-planes and y-planes and the (ϕ, θ) notation yields similar results and will not be detailed here.

Equations (4.85) and (4.86) allow us to compute the 3D Radon transform given a pair of angles (ϕ, θ) by using transformation to explicit plane representation with s_1 and s_2.

4.5.3. 3D discrete Fourier slice theorem

Similar to the 2D case, we can derive a Fourier slice theorem for the 3D case. The 3D Fourier slice theorem associates the 1D discrete Fourier transform of the 3D discrete Radon transform with the discrete Fourier transform of the image I. The 3D Fourier slice theorem is summarized in Theorem 4.5.6.

Theorem 4.5.6 (3D Fourier slice theorem). *Given a 3D image* $I \in \mathbb{I}_{n\times n\times n}$ *and a plane* p *with* $C(p) = (q, s_1, s_2, t)$, $|s_1| \le 1$, $|s_2| \le 1$, $t \in \{-3n/2, \ldots, 3n/2\}$ *(defined in Definition 4.5.4),*

$$\widehat{RI}(s_1, s_2, k) = \begin{cases} \hat{I}(k, -s_1 k, -s_2 k), & q = 1, \\ \hat{I}(-s_1 k, k, -s_2 k), & q = 2, \\ \hat{I}(-s_1 k, -s_2 k, k), & q = 3, \end{cases} \tag{4.87}$$

where $\widehat{RI}$ *is the 1D Fourier of* I *along the parameter* t,

$$\widehat{RI}(s_1, s_2, k) = \sum_{t=-3n/2}^{3n/2} RI(s_1, s_2, t) e^{-2\pi \imath k t/m}, \quad k \in \mathbb{Z}, \ \frac{-3n}{2} \le k \le \frac{3n}{2}, \tag{4.88}$$

$$\hat{I}(\xi_1, \xi_2, \xi_3) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u, v, w) e^{(-2\pi \imath/m)(\xi_1 u + \xi_2 v + \xi_3 w)}. \tag{4.89}$$

See [8] for the construction and proof of Theorem 4.5.6.

4.5.4. 3D pseudopolar grid

As for now, we defined RI for continuous slopes (s_1, s_2) in $[-1, 1] \times [-1, 1]$ while I is discrete. We would like to discretize s_1 and s_2 while satisfying the properties in Section 4.1.2.

We define three sets

$$\begin{aligned} S_1 &\triangleq \left\{ \frac{l}{(n/2)} \mid l = \frac{-n}{2}, \ldots, \frac{n}{2} \right\}, \\ S_2 &\triangleq \left\{ \frac{j}{(n/2)} \mid j = \frac{-n}{2}, \ldots, \frac{n}{2} \right\}, \\ T &\triangleq \left\{ t \in \mathbb{Z} \mid \frac{-3n}{2} \le t \le \frac{3n}{2} \right\}. \end{aligned} \tag{4.90}$$

The 3D discrete Radon transform will be defined as the restriction of RI (Definition 4.5.5) to the discrete set of slopes $(s_1, s_2) \in S_1 \times S_2$ with $t \in T$.

Definition 4.5.7 (3D discrete Radon transform). Given a plane p. Let $C(p) = (q, s_1, s_2, t)$ be the canonized form of p with slopes $(s_1, s_2) \in S_1 \times S_2$, $t \in T$, then,

$$RI(s_1, s_2, t) \triangleq \begin{cases} R_1 I(s_1, s_2, t), & q = 1, \\ R_2 I(s_1, s_2, t), & q = 2, \\ R_3 I(s_1, s_2, t), & q = 3, \end{cases} \tag{4.91}$$

where $R_1 I$, $R_2 I$, and $R_3 I$ are defined in Definition 4.5.5.

RI is not defined for every plane p, but only for planes p such that for $C(p) = (q, s_1, s_2, t)$ it holds that $(s_1, s_2) \in S_1 \times S_2$ and $t \in T$. The difference between Definitions 4.5.5 and 4.5.7 is in the discrete set of slopes $S_1 \times S_2$, which replaces the continuous set of slopes $[-1, 1] \times [-1, 1]$. Definition 4.5.7 describes a transform that takes an image I of size n^3 into an array of size $3 \times (3n + 1) \times (n + 1)^2$.

The Fourier slice theorem (Theorem 4.5.6) holds also for the discrete set of slopes $S_1 \times S_2$ for each of the plane types.

(i) For an x-plane,

$$\widehat{RI}\left(\frac{2l}{n}, \frac{2j}{n}, k\right) = \hat{I}\left(k, -\frac{2l}{n}k, -\frac{2j}{n}k\right). \tag{4.92}$$

(ii) For a y-plane,

$$\widehat{RI}\left(\frac{2l}{n}, \frac{2j}{n}, k\right) = \hat{I}\left(-\frac{2l}{n}k, k, -\frac{2j}{n}k\right). \tag{4.93}$$

(iii) For a z-plane,

$$\widehat{RI}\left(\frac{2l}{n}, \frac{2j}{n}, k\right) = \hat{I}\left(-\frac{2l}{n}k, -\frac{2j}{n}k, k\right), \tag{4.94}$$

where $l, j \in \{-n/2, \ldots, n/2\}$, $k \in \{-3n/2, \ldots, 3n/2\}$ and $\hat{I}$ is the trigonometric polynomial given by (4.89).

Next we describe how the samples of $\hat{I}$ in (4.92)–(4.94) are scattered in $\mathbb{R}^3$. In order to obtain $\widehat{RI}$ over all x-planes, by (4.92) we sample $\hat{I}$ at

$$P_1 \triangleq \left\{\left(k, -\frac{2l}{n}k, -\frac{2j}{n}k\right) \mid l, j \in \left\{\frac{-n}{2}, \ldots, \frac{n}{2}\right\},\ k \in \left\{\frac{-3n}{2}, \ldots, \frac{3n}{2}\right\}\right\}. \tag{4.95}$$

Similarly, to obtain $\widehat{RI}$ over all y-planes, by (4.93) we sample $\hat{I}$ at

$$P_2 \triangleq \left\{\left(-\frac{2l}{n}k, k, -\frac{2j}{n}k\right) \mid l, j \in \left\{\frac{-n}{2}, \ldots, \frac{n}{2}\right\},\ k \in \left\{\frac{-3n}{2}, \ldots, \frac{3n}{2}\right\}\right\} \tag{4.96}$$

and finally, to obtain $\widehat{RI}$ over all z-planes, by (4.94) we sample $\hat{I}$ at

$$P_3 \triangleq \left\{\left(-\frac{2l}{n}k, -\frac{2j}{n}k, k\right) \mid l, j \in \left\{\frac{-n}{2}, \ldots, \frac{n}{2}\right\},\ k \in \left\{\frac{-3n}{2}, \ldots, \frac{3n}{2}\right\}\right\}. \tag{4.97}$$

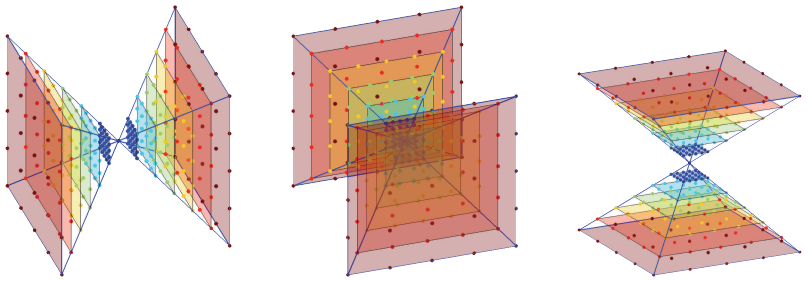
The set

$$P \triangleq P_1 \cup P_2 \cup P_3 \tag{4.98}$$

is called the *pseudopolar grid*.

See Figures 4.5(a)–4.5(c) for illustrations of the sets P_1, P_2, and P_3.

The pseudopolar Fourier transform is defined by sampling the trigonometric polynomial $\hat{I}$ on the pseudopolar grid P.



(a) The sector P_1 (b) The sector P_2 (c) The sector P_3

FIGURE 4.5. The pseudopolar grid.

Definition 4.5.8 (pseudopolar Fourier transform). The pseudopolar Fourier transform PP_i $(i = 1, 2, 3)$ is a linear transformation from 3D images $I \in \mathbb{I}_{n\times n\times n}$ defined by

$$\begin{aligned} PP_1 I(k,l,j) &= \hat{I}\left(k, -\frac{2l}{n}k, -\frac{2j}{n}k\right), \\ PP_2 I(k,l,j) &= \hat{I}\left(-\frac{2l}{n}k, k, -\frac{2j}{n}k\right), \\ PP_3 I(k,l,j) &= \hat{I}\left(-\frac{2l}{n}k, -\frac{2j}{n}k, k\right), \end{aligned} \tag{4.99}$$

where

$$\begin{aligned} &l, j \in \left\{\frac{-n}{2}, \ldots, \frac{n}{2}\right\}, \qquad k \in \left\{\frac{-3n}{2}, \ldots, \frac{3n}{2}\right\}, \qquad m = 3n+1, \\ &\hat{I}(\xi_1, \xi_2, \xi_3) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{(-2\pi\imath/m)(\xi_1 u + \xi_2 v + \xi_3 w)}. \end{aligned} \tag{4.100}$$

Using the pseudopolar Fourier transform we can express RI as

$$R_i I = F^{-1} \circ PP_i I \quad (i = 1, 2, 3), \tag{4.101}$$

where F^{-1} is the inverse Fourier transform.

We can regard s_1 and s_2 as "pseudoangles" and k as a "pseudoradius." This corresponds to the (r, ϕ, θ) representation of the polar grid.

4.5.5. Rapid computation of the 3D discrete Radon transform

We sketch an $O(n^3 \log n)$ algorithm for the computation of the 3D discrete Radon transform.

Equation (4.101) states that we can compute $R_i I$ by calling the 1D inverse Fourier transform $(n+1)^2$ times for each $PP_i I$. Each vector in $PP_i I$ corresponds

to fixed slopes l, j. Hence, all applications of F^{-1} take $O(n^3 \log n)$ operations. It remains to show that the 3D pseudopolar Fourier transform PP_iI $(i = 1, 2, 3)$ can be computed using $O(n^3 \log n)$ operations.

From Definition 4.5.8 it follows that given an image $I \in \mathbb{I}_{n\times n\times n}$, the pseudopolar Fourier transform of I is defined by

$$PPI(s,k,l,j) = \begin{cases} \hat{I}\left(k, -\dfrac{2l}{n}k, -\dfrac{2j}{n}k\right), & s = 1, \\ \hat{I}\left(-\dfrac{2l}{n}k, k, -\dfrac{2j}{n}k\right), & s = 2, \\ \hat{I}\left(-\dfrac{2l}{n}k, -\dfrac{2j}{n}k, k\right), & s = 3, \end{cases} = \begin{cases} PP_1I(k,l,j), \\ PP_2I(k,l,j), \\ PP_3I(k,l,j), \end{cases} \tag{4.102}$$

where

$$\begin{aligned} &l, j \in \left\{\frac{-n}{2}, \ldots, \frac{n}{2}\right\}, \qquad k \in \left\{\frac{-3n}{2}, \ldots, \frac{3n}{2}\right\}, \qquad m = 3n+1, \\ &\hat{I}(\xi_1, \xi_2, \xi_3) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{(-2\pi\imath/m)(\xi_1 u + \xi_2 v + \xi_3 w)}. \end{aligned} \tag{4.103}$$

The main idea of the algorithm is to use the samples of the equally spaced 3D DFT (which can be computed in $O(n^3 \log n)$), and resample them rapidly on the pseudopolar grid points, using the fractional Fourier transform (defined in Definition 4.3.3).

4.5.5.1. Algorithm description for the rapid computation of the 3D pseudopolar Fourier transform (pseudocode)

Throughout this section we will use the following notation.

Notation 4.5.9.

(i) F_n^{-1}: 1D inverse DFT.
(ii) F_m^{α}: fractional Fourier transform with factor α. The operator accepts a sequence of length n, pads it symmetrically to length $m = 3n+1$, applies to it the fractional Fourier transform with factor α, and returns the $n+1$ central elements.
(iii) F_3: 3D DFT.
(iv) $G_{k,n}$: consecutive application of F_n^{-1} followed by $F_m^{2k/n}$:

$$G_{k,n} = F_m^{2k/n} \circ F_n^{-1}. \tag{4.104}$$

We illustrate the operations of Algorithm 4.3 on Res_1.

Input: image I of size $n \times n \times n$.
Output: three arrays: Res_1, Res_2, Res_3 of size $m \times (n+1) \times (n+1)$.
Working arrays: six auxiliary arrays T_1, T_2, T_3, T_1', T_2', T_3':

(i) T_1 is of size $m \times n \times (n+1)$,
(ii) T_2 is of size $(n+1) \times m \times n$,
(iii) T_3 is of size $n \times (n+1) \times m$,
(iv) T_1', T_2', T_3' are of size $m \times (n+1) \times (n+1)$.

Process: Res_1 **computation**.

(1) Pad I symmetrically along the x-axis to length $3n+1$. Denote the result $\widetilde{I}_1$.
(2) $\hat{I}_d \leftarrow F_3(\widetilde{I}_1)$.
(3) For each k and l,

$$T_1(k,l,\cdot) = G_{k,n}(\hat{I}_d(k,l,\cdot)). \tag{4.105}$$

(4) For each k and j,

$$T_1'(k,\cdot,j) = G_{k,n}(T_1(k,\cdot,j)). \tag{4.106}$$

(5) For each k, l, j,

$$\mathrm{Res}_1(k,l,j) = T_1'(k,-l,-j). \tag{4.107}$$

Res_2 **computation**.

(6) Pad I symmetrically along the y-axis to length $3n+1$. Denote the result $\widetilde{I}_2$.
(7) $\hat{I}_d \leftarrow F_3(\widetilde{I}_2)$.
(8) For each k and l,

$$T_2(\cdot,k,j) = G_{k,n}(\hat{I}_d(\cdot,k,j)). \tag{4.108}$$

(9) For each k and j,

$$T_2'(k,l,\cdot) = G_{k,n}(T_2(l,k,\cdot)). \tag{4.109}$$

(10) For each k, l, j,

$$\mathrm{Res}_2(k,l,j) = T_2'(k,-l,-j). \tag{4.110}$$

Res_3 **computation**.

(11) Pad I symmetrically along the z-axis to length $3n+1$. Denote the result $\widetilde{I}_3$.
(12) $\hat{I}_d \leftarrow F_3(\widetilde{I}_3)$.
(13) For each k and l,

$$T_3(l,\cdot,k) = G_{k,n}(\hat{I}_d(l,\cdot,k)). \tag{4.111}$$

(14) For each k and j,

$$T_3'(k,\cdot,j) = G_{k,n}(T_3(\cdot,j,k)). \tag{4.112}$$

(15) For each k, l, j,

$$\mathrm{Res}_3(k,l,j) = T_3'(k,-l,-j). \tag{4.113}$$

Algorithm 4.3

(1) Both ends of the x-direction of the image I are zero-padded. The 3D DFT of the padded image is computed. The results are placed in $\widehat{I}_d$.
(2) Each vector from $\widehat{I}_d$, which corresponds to fixed x and y, is taken. Apply on it the 1D inverse DFT and resample it using 1D fractional Fourier transform where $\alpha = 2x/n$. The result 3D array is denoted by T_1.
(3) Each vector from T_1, which corresponds to fixed x and z, is taken. Apply on it 1D inverse DFT and resample it using 1D fractional Fourier transform where $\alpha = 2x/n$. The result array is T_1'.
(4) Flip T_1' along the y- and z-axes.

Theorem 4.5.10 (algorithm correctness of Algorithm 4.3).
(i) $\mathrm{Res}_1(k,l,j) = PPI(1,k,l,j)$.
(ii) $\mathrm{Res}_2(k,l,j) = PPI(2,k,l,j)$.
(iii) $\mathrm{Res}_3(k,l,j) = PPI(3,k,l,j)$.

See [8] for a proof of Theorem 4.5.10 and a complexity analysis of Algorithm 4.3.

4.5.6. Complexity

We first compute the complexity of calculating Res_1. Each application of $G_{k,n}$ involves the application of a 1D inverse Fourier transform followed by the computation of the fractional Fourier transform. Both the computation of the 1D inverse Fourier transform and the computation of the fractional Fourier transform require $O(n\log n)$ operations. Thus, the computation of $G_{k,n}$ requires $O(n\log n)$ operations. The complexity of steps (2) and (3) in Algorithm 4.3 is of $(3n+1)\times n$ and $(3n+1)\times(n+1)$ applications of $G_{k,n}$, respectively. Since each application costs $O(n\log n)$ operations, this gives a total of $O(n^3\log n)$ operations for each of steps (2) and (3). The complexity of the 3D DFT in step (1) is $O(n^3\log n)$ and the complexity of step (4) (flipping T_1') is $O(n^3)$. This gives a total of $O(n^3\log n)$ for the computation of Res_1. The complexity of calculating Res_2 and Res_3 is identical to the complexity of calculating Res_1, and therefore the total complexity of the algorithm is $3\cdot O(n^3\log n)$, namely, $O(n^3\log n)$ operations.

4.5.7. Invertibility of the 3D discrete Radon transform

We show that our definition of the 3D discrete Radon transform is invertible. Given the 3D discrete Radon transform RI, we show that it is possible to uniquely recover I from RI.

From (4.101)

$$R_iI = F^{-1}\circ PP_iI \quad (i=1,2,3), \tag{4.114}$$

where F^{-1} (the 1D inverse Fourier transform) is invertible, and therefore left to show that I can be recovered from PP_iI, $i=1,2,3$.

Given the values of PPI (4.102), we take a vector of samples from PP_1I, which corresponds to some $k_0 \neq 0$. From Definition 4.5.8,

$$PP_1I(k_0, l, j) = \hat{I}\left(k_0, -\frac{2lk_0}{n}, -\frac{2jk_0}{n}\right), \quad l, j \in \left\{\frac{-n}{2}, \ldots, \frac{n}{2}\right\}. \tag{4.115}$$

By expanding $\hat{I}$ using (4.89) at $(k_0, -2lk_0/n, -2jk_0/n)$, we have

$$\begin{aligned}&\hat{I}\left(k_0, -\frac{2lk_0}{n}, -\frac{2jk_0}{n}\right)\\ &\quad= \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u, v, w) e^{-2\pi\imath k_0 u/m} e^{-2\pi\imath(-2lk_0/n)v/m} e^{-2\pi\imath(-2jk_0/n)w/m}.\end{aligned} \tag{4.116}$$

Rewriting (4.116), we have

$$\begin{aligned}&\hat{I}\left(k_0, -\frac{2lk_0}{n}, -\frac{2jk_0}{n}\right)\\ &\quad= \sum_{w=-n/2}^{n/2-1} \left(\sum_{v=-n/2}^{n/2-1} \sum_{u=-n/2}^{n/2-1} I(u, v, w) e^{-2\pi\imath k_0 u/m} e^{-2\pi\imath(-2lk_0/n)v/m} \right) e^{-2\pi\imath(-2jk_0/n)w/m}\\ &\quad= \sum_{w=-n/2}^{n/2-1} c_{k_0,l}(w) e^{-2\pi\imath(-2jk_0/n)w/m},\end{aligned} \tag{4.117}$$

where

$$\begin{aligned}&c_{k_0,l}(w)\\ &\quad= \sum_{v=-n/2}^{n/2-1} \sum_{u=-n/2}^{n/2-1} I(u, v, w) e^{-2\pi\imath k_0 u/m} e^{-2\pi\imath(-2lk_0/n)v/m}, \quad w \in \left\{\frac{-n}{2}, \ldots, \frac{n}{2-1}\right\}.\end{aligned} \tag{4.118}$$

For fixed k_0, l and variable j denote

$$T_{k_0,l}\left(\frac{-2jk_0}{n}\right) \triangleq \hat{I}\left(k_0, \frac{-2lk_0}{n}, \frac{-2jk_0}{n}\right). \tag{4.119}$$

Then, from (4.117) we have

$$T_{k_0,l}\left(\frac{-2jk_0}{n}\right) = \sum_{w=-n/2}^{n/2-1} c_{k_0,l}(w) e^{-2\pi\imath(-2jk_0/n)w/m}. \tag{4.120}$$

In other words, $T_{k_0,l}(-2jk_0/n)$, $j \in \mathbb{Z}$, $-n/2 \le j \le n/2$, are the values of the polynomial

$$T_{k_0,l}(x) = \sum_{w=-n/2}^{n/2-1} c_{k_0,l}(w) e^{-2\pi \imath x w/m} \tag{4.121}$$

at $\{-2jk_0/n\}$. Since we have the values of $T_{k_0,l}(x)$ at $n+1$ distinct points $\{-2jk_0/n\}$ (and thus we required $k_0 \neq 0$), we can uniquely determine $\{c_{k_0,l}(w)\}$ and $T_{k_0,l}(x)$, and therefore we can compute $T_{k_0,l}(x)$ for any x.

By computing $T_{k_0,l}(x)$ at integer points we can recover

$$T_{k_0,l}(j) = \sum_{w=-n/2}^{n/2-1} c_{k_0,l}(w) e^{-2\pi \imath j w/m} \tag{4.122}$$

for every k_0, l. Substituting $c_{k_0,l}(w)$ (4.118) in (4.122) we obtain

$$\begin{aligned} H_{k_0,j_0}\left(\frac{-2lk_0}{n}\right) &\triangleq T_{k_0,l}(j_0) \\ &= \sum_{w=-n/2}^{n/2-1} \left(\sum_{v=-n/2}^{n/2-1} \sum_{u=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi \imath k_0 u/m} e^{-2\pi \imath(-2lk_0/n)v/m} \right) e^{-2\pi \imath j_0 w/m} \\ &= \sum_{v=-n/2}^{n/2-1} \left(\sum_{u=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi \imath k_0 u/m} e^{-2\pi \imath j_0 w/m} \right) e^{-2\pi \imath(-2lk_0/n)v/m} \\ &= \sum_{v=-n/2}^{n/2-1} c'_{k_0,j_0}(v) e^{-2\pi \imath(-2lk_0/n)v/m}, \end{aligned} \tag{4.123}$$

where

$$c'_{k_0,j_0}(v) = \sum_{u=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi \imath k_0 u/m} e^{-2\pi \imath j_0 w/m}, \quad v \in \left\{ \frac{-n}{2}, \ldots, \frac{n}{2-1} \right\}. \tag{4.124}$$

From (4.123),

$$H_{k_0,j_0}\left(\frac{-2lk_0}{n}\right) = \sum_{v=-n/2}^{n/2-1} c'_{k_0,j_0}(v) e^{-2\pi \imath(-2lk_0/n)v/m} \tag{4.125}$$

and we have that $H_{k_0,j_0}(-2lk_0/n)$ are the samples of the trigonometric polynomial

$$H_{k_0,j_0}(x) = \sum_{v=-n/2}^{n/2-1} c'_{k_0,j_0}(v) e^{-2\pi \imath x v/m} \tag{4.126}$$

at $\{-2lk_0/n\}$. Again, since we have the values $H_{k_0,j_0}(x)$ at $n+1$ distinct points (for $n+1$ distinct values of l), we can uniquely determine $\{c'_{k_0,j_0}(v)\}$ and the underlying trigonometric polynomial $H_{k_0,j_0}(x)$ given by (4.126).

Evaluating $H_{k_0,j_0}(x)$ for integer points, we obtain using (4.124)

$$\begin{aligned} H_{k_0,j_0}(l) &= \sum_{v=-n/2}^{n/2-1} c'_{k_0,j_0}(v) e^{-2\pi \imath l v/m} \\ &= \sum_{v=-n/2}^{n/2-1} \sum_{u=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi \imath k_0 u/m} e^{-2\pi \imath j_0 w/m} e^{-2\pi \imath l v/m} \\ &= \hat{I}(k_0, l, j_0). \end{aligned} \tag{4.127}$$

Equation (4.127) states that we have recovered $\hat{I}$ at integer grid points for every $k_0 \neq 0$ (the entire discrete grid except the plane $\xi_1 = 0$). Remains to evaluate $\hat{I}$ on the plane $\xi_1 = 0$, or, in other words, the values $\hat{I}(0, l, j)$.

As before, by taking a sequence of samples from PP_2I, which corresponds to some $k_0 \neq 0$, we have from Definition 4.5.8

$$PP_2I(k_0, l, j) = \hat{I}\Big(-\frac{2lk_0}{n}, k_0, -\frac{2jk_0}{n} \Big). \tag{4.128}$$

By repeating exactly the same arguments as above we have using (4.89)

$$\begin{aligned} T'_{k_0,l}\Big(\frac{-2jk_0}{n}\Big) &\triangleq \hat{I}\Big(-\frac{2lk_0}{n}, k_0, -\frac{2jk_0}{n} \Big) \\ &= \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi \imath(-2lk_0/n)u/m} e^{-2\pi \imath k_0 v/m} e^{-2\pi \imath(-2jk_0/n)w/m} \\ &= \sum_{w=-n/2}^{n/2-1} \Bigg(\sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi \imath(-2lk_0/n)u/m} e^{-2\pi \imath k_0 v/m} \Bigg) e^{-2\pi \imath(-2jk_0/n)w/m} \\ &= \sum_{w=-n/2}^{n/2-1} c_{k_0,l}(w) e^{-2\pi \imath(-2jk_0/n)w/m}, \end{aligned} \tag{4.129}$$

where

$$c_{k_0,l}(w) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi\imath(-2lk_0/n)u/m} e^{-2\pi\imath k_0 v/m}, \quad w \in \left\{ \frac{-n}{2}, \ldots, \frac{n}{2-1} \right\}. \tag{4.130}$$

Using $n+1$ distinct samples $\{T'_{k_0,l}(-2jk_0/n)\}$ at $n+1$ distinct points $\{-2jk_0/n\}$, $j = -n/2, \ldots, n/2$, we can compute $\{c_{k_0,l}(w)\}$ and the trigonometric polynomial

$$T'_{k_0,l}(x) = \sum_{w=-n/2}^{n/2-1} c_{k_0,l}(w) e^{-2\pi\imath xw/m}. \tag{4.131}$$

For every l, we evaluate $T'_{k_0,l}(x)$ at integer points and by using (4.130) we obtain

$$\begin{aligned} H'_{k_0,j_0}\left(\frac{-2lk_0}{n}\right) &\triangleq T'_{k_0,l}(j_0) \\ &= \sum_{w=-n/2}^{n/2-1} c_{k_0,l}(w) e^{-2\pi\imath j_0 w/m} \\ &= \sum_{w=-n/2}^{n/2-1} \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi\imath(-2lk_0/n)u/m} e^{-2\pi\imath k_0 v/m} e^{-2\pi\imath j_0 w/m} \\ &= \sum_{u=-n/2}^{n/2-1} \left(\sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi\imath k_0 v/m} e^{-2\pi\imath j_0 w/m} \right) e^{-2\pi\imath(-2lk_0/n)u/m} \\ &= \sum_{u=-n/2}^{n/2-1} c'_{k_0,j_0}(u) e^{-2\pi\imath(-2lk_0/n)u/m}, \end{aligned} \tag{4.132}$$

where

$$c'_{k_0,j_0}(u) = \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi\imath k_0 v/m} e^{-2\pi\imath j_0 w/m}, \quad u \in \left\{ \frac{-n}{2}, \ldots, \frac{n}{2-1} \right\}. \tag{4.133}$$

Using $n+1$ distinct samples $\{H'_{k_0,j_0}(-2lk_0/n)\}$ from (4.132) at $n+1$ distinct points ($n+1$ distinct values of l), $\{c'_{k_0,j_0}(u)\}$ is uniquely determined and

$$H'_{k_0,j_0}(x) = \sum_{u=-n/2}^{n/2-1} c'_{k_0,j_0}(u) e^{-2\pi\imath xu/m}. \tag{4.134}$$

Evaluating $H'_{k_0,j_0}(x)$ at integer points and using (4.133), we obtain

$$\begin{aligned} H'_{k_0,j_0}(l) &= \sum_{u=-n/2}^{n/2-1} c'_{k_0,j_0}(u) e^{-2\pi \imath lu/m} \\ &= \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) e^{-2\pi \imath lu/m} e^{-2\pi \imath k_0 v/m} e^{-2\pi \imath j_0 w/m} \\ &= \hat{I}(l,k_0,j_0). \end{aligned} \tag{4.135}$$

We evaluated $\hat{I}(l,k_0,j)$ at grid points where $k_0 \neq 0$. Specifically, we computed $\hat{I}(0,k_0,j)$.

Finally, we have to compute $\hat{I}$ on the line $\xi_1 = 0, \xi_2 = 0$. By using exactly the same arguments on PP_3I as above, we evaluate $\hat{I}$ on all grid points except $\xi_3 = 0$. Thus, we have the values of $\hat{I}$ on all grid points except the origin $\hat{I}(0,0,0)$. Since $PP_1I(0,0,0) = \hat{I}(0,0,0)$, we have the values of $\hat{I}$ on the entire grid.

Since we can recover $\hat{I}$ (the 3D DFT of I) from PPI, and trivially recover I from $\hat{I}$, it follows that we can recover I from PPI, and thus PP in invertible.

4.6. 3D discrete X-ray transform

The X-ray transform is an important practical tool in many scientific and industrial areas. An example of such area is computerized tomography (CT) scanning where the X-ray transform plays a major role in the derivation and implementation of various tomographic methods (see [4]). The work presented here is based on [10] and it is a continuation of the works in [5, 6, 8].

4.6.1. The continuous X-ray transform

The continuous X-ray transform of a 3D function $f(x,y,z)$, denoted by $\mathcal{P}f$, is defined by the set of all line integrals of f. For a line L, defined by a unit vector θ and a point x on L, we express L as

$$L(t) = x + t\theta, \quad t \in \mathbb{R}. \tag{4.136}$$

The X-ray transform of f on L is defined as

$$\mathcal{P}f(L) \triangleq \int_{-\infty}^{\infty} f(x+t\theta)dt. \tag{4.137}$$

The X-ray transform maps each line L in $\mathbb{R}^3$ to a real value that represents the projection of the function f along the line L. For convenience, (4.137) is sometimes

written with the notation

$$\mathcal{P}_\theta f(x) \triangleq \mathcal{P} f(L), \tag{4.138}$$

where L is given by (4.136).

The X-ray transform is closely related to the Radon transform. However, while the 3D X-ray transform is defined using line integrals of a function f, we define the 3D Radon transform as integrals of f over all planes in $\mathbb{R}^3$. Note that in the 2D case, the X-ray transform coincides with the Radon transform [1–3].

The Fourier slice theorem connects the continuous X-ray transform, defined by (4.137), with the Fourier transform. For a given 3D function f, it defines the relation between the 2D Fourier transform of the X-ray transform of f and the 3D Fourier transform of f. The Fourier slice theorem is summarized in the following theorem.

Theorem 4.6.1. *For a function $f(x, y, z)$ and a family of lines in $\mathbb{R}^3$, whose direction is given by the unit vector θ, it holds that*

$$\widehat{\mathcal{P}_\theta f}(\xi) = \hat{f}(\xi), \tag{4.139}$$

where $\xi \in \theta^\perp$ and $\theta^\perp$ is the subspace perpendicular to θ.

4.6.2. Discretization guidelines

We define a 3D $n \times n \times n$ image as the set

$$I = \left\{ I(u, v, w) : \frac{-n}{2} \le u, v, w \le \frac{n}{2-1} \right\}. \tag{4.140}$$

Note that we define I in (4.140) as a cube of voxels with an even side of length n. We refer to the image I as a cube of size $n \times n \times n$ to simplify the formulation of the discrete transform. The entire formulation can be repeated for an image I with arbitrary dimensions $n_1 \times n_2 \times n_3$.

We present here a discrete definition of the 3D X-ray transform for discrete images, which satisfies the properties in Section 4.1.2. We prove the Fourier slice theorem that relates our definition of the X-ray transform with the Fourier transform of the image I and develop a rapid computational algorithm, which is based on the Fourier slice theorem. We also show that our discrete X-ray transform is invertible.

The present work is based on [5, 6, 8]. However, there are important differences between them and the present work. References [6, 8] establish a framework for surface integrals decomposition of discrete objects. The present work derives a framework for line integrals decomposition of discrete objects. The two frameworks coincide for the 2D case, but for higher dimensions there are some fundamental differences between them. Although both frameworks follow the same

guidelines and use the same building blocks, they require different discretizations of the continuous space because of the difference between the underlying continuous transforms. This results in a different frequency domain geometry, a different relation between the space domain and the frequency domain, and a different numerical computation algorithm.

4.6.3. Semidiscrete transform definition

We parameterize a line in $\mathbb{R}^3$ as the intersection of two planes. Using this parameterization, we define three families of lines, which we call *x-lines*, *y-lines*, and *z-lines*. Formally, an x-line is defined as

$$l_x(\alpha,\beta,c_1,c_2) = \begin{cases} y = \alpha x + c_1, \\ z = \beta x + c_2, \end{cases} \quad |\alpha| \le 1,\ |\beta| \le 1,\ c_1, c_2 \in \{-n,\dots,n\}. \tag{4.141}$$

Figure 4.6 is a 3D illustration of the family of x-lines that corresponds to $c_1 = c_2 = 0$, together with its projections on different axes. Similarly, a y-line and a z-line are defined as

$$l_y(\alpha,\beta,c_1,c_2) = \begin{cases} x = \alpha y + c_1, \\ z = \beta y + c_2, \end{cases} \quad |\alpha| \le 1,\ |\beta| \le 1,\ c_1, c_2 \in \{-n,\dots,n\},$$

$$l_z(\alpha,\beta,c_1,c_2) = \begin{cases} x = \alpha z + c_1, \\ y = \beta z + c_2, \end{cases} \quad |\alpha| \le 1,\ |\beta| \le 1,\ c_1, c_2 \in \{-n,\dots,n\}. \tag{4.142}$$

We denote the sets of all x-lines, y-lines, and z-lines in $\mathbb{R}^3$ by $\mathcal{L}_x$, $\mathcal{L}_y$, and $\mathcal{L}_z$, respectively. Also, we denote the family of lines that corresponds to a fixed direction (α,β) and variable intercepts (c_1,c_2), by $l_x(\alpha,\beta)$, $l_y(\alpha,\beta)$, and $l_z(\alpha,\beta)$ for a family of x-lines, y-lines, and z-lines, respectively. See Figure 4.7 for an illustration of the different families of lines for $c_1 = c_2 = 0$.

Each line in $\mathbb{R}^3$ can be expressed as either an x-line, y-line, or z-line. In other words, each line in $\mathbb{R}^3$ belongs to $\mathcal{L}_x$, $\mathcal{L}_y$, or $\mathcal{L}_z$. Note that the sets $\mathcal{L}_x$, $\mathcal{L}_y$, and $\mathcal{L}_z$ are not disjoint.

For a discrete image I of size $n \times n \times n$ we define three continuous extensions of I, which we denote by I_x, I_y, and I_z. Each of the extensions I_x, I_y and I_z is a continuous function in the directions perpendicular to its index. This means, for example, that I_x is a continuous function in the y- and z-directions. Formally, we

define these three extensions by

$$I_x(u,y,z) = \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) D_m(y-v) D_m(z-w), \quad u \in \left\{ \frac{-n}{2}, \ldots, \frac{n}{2-1} \right\}, \quad y,z \in \mathbb{R}, \tag{4.143}$$

$$I_y(x,v,z) = \sum_{u=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u,v,w) D_m(x-u) D_m(z-w), \quad v \in \left\{ \frac{-n}{2}, \ldots, \frac{n}{2-1} \right\}, \quad x,z \in \mathbb{R}, \tag{4.144}$$

$$I_z(x,y,w) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} I(u,v,w) D_m(x-u) D_m(y-v), \quad w \in \left\{ \frac{-n}{2}, \ldots, \frac{n}{2-1} \right\}, \quad x,y \in \mathbb{R}, \tag{4.145}$$

where D_m is the Dirichlet kernel of length $m = 2n+1$ given by

$$D_m(t) = \frac{\sin \pi t}{m \sin(\pi t/m)}. \tag{4.146}$$

Next, we use I_x, I_y, and I_z to define the discrete X-ray transform. For an x-line $l_x(\alpha,\beta,c_1,c_2) \in \mathcal{L}_x$, given by (4.141), we define the discrete X-ray transform $P_x I(\alpha,\beta,c_1,c_2)$ as

$$P_x I(\alpha,\beta,c_1,c_2) = \sum_{u=-n/2}^{n/2-1} I_x(u, \alpha u + c_1, \beta u + c_2), \quad |\alpha| \le 1,\ |\beta| \le 1,\ c_1,c_2 \in \{-n,\ldots,n\}, \tag{4.147}$$

where I_x is given by (4.143). The transformation $P_x : \mathcal{L}_x \to \mathbb{R}$ is obtained by traversing the line l_x with unit steps in the x-direction, and for each integer u the value of the image I at the point $(u, \alpha u + c_1, \beta u + c_2)$ is summed.

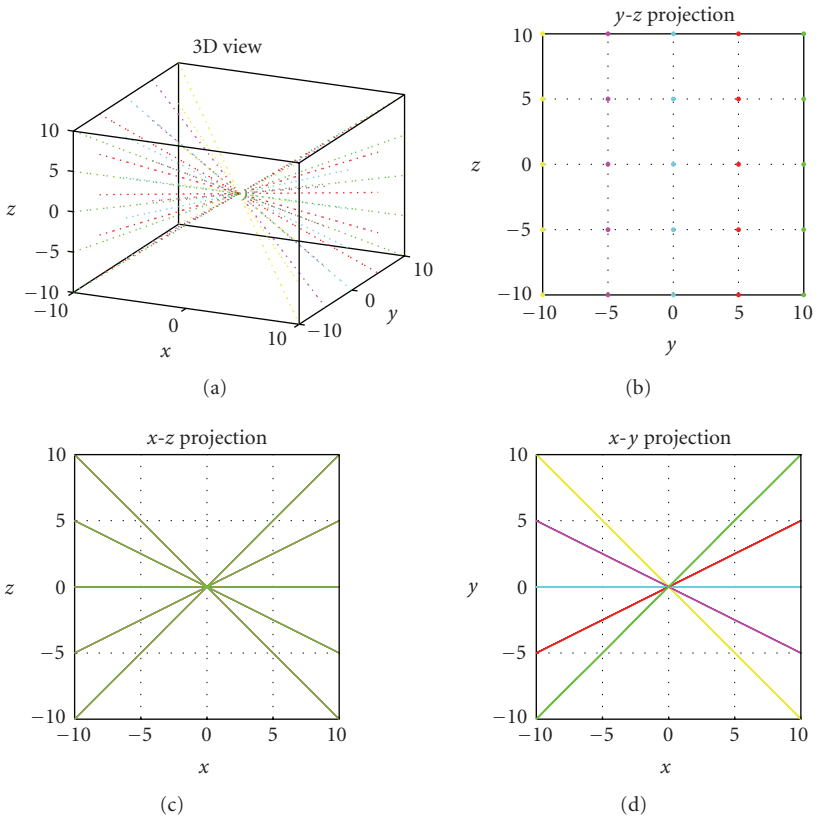


FIGURE 4.6. The set of x-lines $\mathcal{L}_x$.

Similarly to (4.147), we define the discrete X-ray transform $P_y I(\alpha, \beta, c_1, c_2)$ for the y-line $l_y(\alpha, \beta, c_1, c_2) \in \mathcal{L}_y$ as

$$P_y I(\alpha, \beta, c_1, c_2) = \sum_{v=-n/2}^{n/2-1} I_y(\alpha v + c_1, v, \beta v + c_2), \tag{4.148}$$

where I_y is given by (4.144). Finally, we define the discrete X-ray transform $P_z I(\alpha, \beta, c_1, c_2)$ for a z-line $l_z(\alpha, \beta, c_1, c_2) \in \mathcal{L}_z$ as

$$P_z I(\alpha, \beta, c_1, c_2) = \sum_{w=-n/2}^{n/2-1} I_z(\alpha w + c_1, \beta w + c_2, w), \tag{4.149}$$

where I_z is given by (4.145).

Equations (4.147)–(4.149) define the X-ray transform for x-lines, y-lines, and z-lines, respectively. Since each line in $\mathbb{R}^3$ can be expressed as either an x-line, y-line, or z-line, then, given a line l, we express it as either an x-line, y-line, or z-line

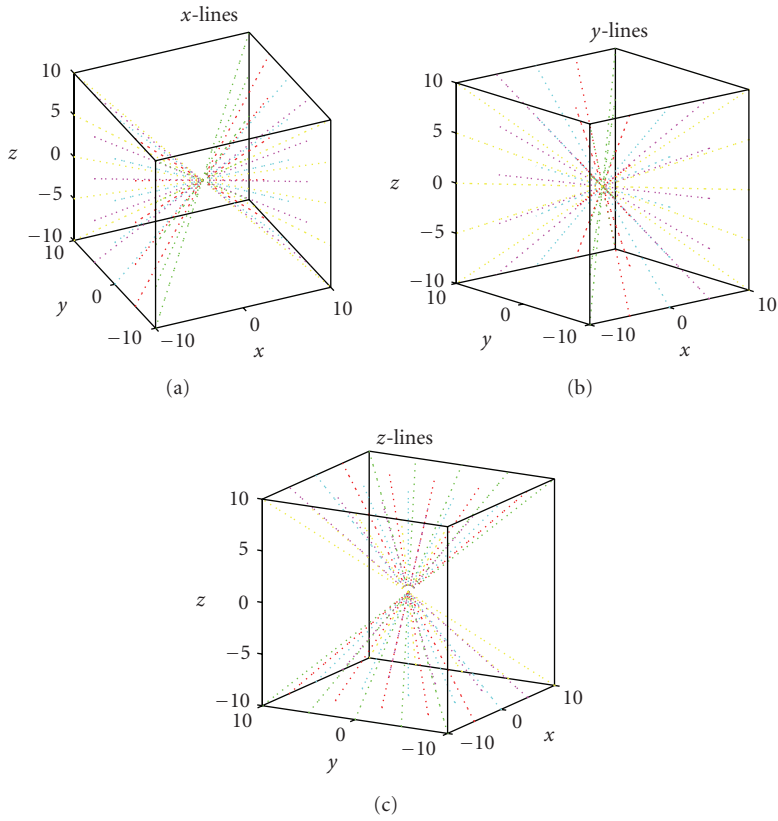


FIGURE 4.7. The line families $\mathcal{L}_x$, $\mathcal{L}_y$, and $\mathcal{L}_z$.

and apply on it our definition of the X-ray transform. Hence, for a given image I and a line l, we define the discrete X-ray transform of I for the line l as the following.

Definition 4.6.2.

$$PI(l) = \begin{cases} P_x I(l), & l \in \mathcal{L}_x, \\ P_y I(l), & l \in \mathcal{L}_y, \\ P_z I(l), & l \in \mathcal{L}_z. \end{cases} \tag{4.150}$$

There is a subtle difference between the continuous X-ray transform, given in (4.137), and the discrete X-ray transform, given in (4.150). The continuous X-ray transform assigns to each line the integral of the object along the line, where the value of the integral is independent of the parameterization of the line. The discrete X-ray transform assigns a value to a specific parameterization of the line. This means that if the same line is written in two different ways in (4.150), then,

it may receive two different values. This problem occurs only for lines that have an angle of 45° in some direction. We can eliminate this problem by using a more subtle construction in (4.150). However, since this issue does not pose any computational problems, we simply ignore it.

The X-ray transform, given by Definition 4.6.2, is defined for a set of lines in $\mathbb{R}^3$ with a discrete set of intercepts (c_1, c_2) and a continuous set of slopes (α, β). Therefore, Definition 4.6.2 is what we call "semidiscrete," since it is discrete with respect to the image I and the set of intercepts (c_1, c_2), but it uses a continuous set of slopes (α, β). In Section 4.6.5 we show how to discretize the set (α, β) to have a fully discrete definition of the X-ray transform, which is rapidly computable and invertible.

4.6.4. Discrete Fourier slice theorem for the discrete X-ray transform

As we showed in (4.139), the continuous X-ray transform satisfies the Fourier slice theorem, which associates the continuous X-ray transform of a function f with the Fourier transform of f. This relation is very useful for both the computation and analysis of the continuous X-ray transform. We will therefore derive a similar relation for the discrete X-ray transform, and we will later utilize it to rapidly compute the discrete X-ray transform.

For a 2D array X of size $m \times m$ ($m = 2n+1$), the 2D discrete Fourier transform (DFT) of X, denoted by $\hat{X}$, is defined by

$$\hat{X}(k,l) = \sum_{u=-n}^{n} \sum_{v=-n}^{n} X(u,v) e^{-2\pi \imath ku/m} e^{-2\pi \imath lv/m}, \quad k,l = -n,\ldots,n. \tag{4.151}$$

The 2D inverse discrete Fourier transform is given by

$$X(u,v) = \sum_{k=-n}^{n} \sum_{l=-n}^{n} \hat{X}(k,l) e^{2\pi \imath ku/m} e^{2\pi \imath lv/m}, \quad u,v = -n,\ldots,n. \tag{4.152}$$

Given a family of x-lines $l_x(\alpha, \beta)$, we denote the X-ray transform of the image I for this family of lines by $P_{x(\alpha,\beta)}I$. Formally,

$$P_{x(\alpha,\beta)}I \triangleq P_x I(\alpha, \beta, c_1, c_2), \quad c_1, c_2 = -n,\ldots,n, \tag{4.153}$$

or for specific c_1 and c_2,

$$P_{x(\alpha,\beta)}I(c_1, c_2) \triangleq P_x I(\alpha, \beta, c_1, c_2), \tag{4.154}$$

where $P_x I(\alpha, \beta, c_1, c_2)$ is given by (4.147). $P_{x(\alpha,\beta)}I$ is the 2D array generated by applying the X-ray transform to a family of x-lines with a fixed direction (α, β).

Theorem 4.6.3 (x-lines Fourier slice theorem [10]). *For a given family of x-lines $l_x(\alpha,\beta)$ with fixed slopes (α,β) and variable intercepts (c_1,c_2), take the 2D array of projections $P_{x(\alpha,\beta)}I$. Then,*

$$\hat{P}_{x(\alpha,\beta)}I(k,l) = \hat{I}(-\alpha k - \beta l, k, l), \tag{4.155}$$

where $\hat{I}$ is given by

$$\hat{I}(\xi_1,\xi_2,\xi_3) = \sum_{u=-n/2}^{n/2-1}\sum_{v=-n/2}^{n/2-1}\sum_{w=-n/2}^{n/2-1} I(u,v,w)e^{-2\pi\iota\xi_1 u/m}e^{-2\pi\iota\xi_2 v/m}e^{-2\pi\iota\xi_3 w/m} \tag{4.156}$$

and $\hat{P}_{x(\alpha,\beta)}I(k,l)$ is the 2D DFT of the array $P_{x(\alpha,\beta)}I$.

Similar theorems hold for y-lines and z-lines.

Geometrically, Theorem 4.6.3 states that the 2D DFT of the discrete X-ray transform over a family of x-lines with fixed slopes (α,β) is equal to the samples of the trigonometric polynomial $\hat{I}$ on the plane defined by the points $(-\alpha k - \beta l, k, l)$. Explicitly, we need to sample $\hat{I}$ on the plane given by the equation $x = -\alpha y - \beta z$. Figure 4.8 depicts these x-planes for various values of α and β. Theorem 4.6.3 is very important for the rapid computation of the discrete X-ray transform, as it relates the discrete X-ray transform of the image I to the 3D DFT of I.

4.6.5. Discretization of the X-ray transform

Definition 4.6.2 defines the X-ray transform over the continuous line sets $\mathcal{L}_x$, $\mathcal{L}_y$, and $\mathcal{L}_z$. These line sets are comprised from lines that have discrete intercepts and continuous slopes. In this section, we define the discrete X-ray transform for a discrete set of lines, which are discrete in both their slopes and intercepts.

Consider the set defined by

$$S = \left\{\frac{2p}{n}\right\}, \quad p = \frac{-n}{2},\ldots,\frac{n}{2}. \tag{4.157}$$

We define the *discrete X-ray transform* as the restriction of Definition 4.6.2 to the set of slopes $S \times S$. Formally, we define three discrete sets of lines:

(i) discrete x-lines

$$\mathcal{L}_x^d = \{l_x(\alpha,\beta,c_1,c_2) \in \mathcal{L}_x \mid \alpha \in S,\ \beta \in S\}; \tag{4.158}$$

(ii) discrete y-lines

$$\mathcal{L}_y^d = \{l_y(\alpha,\beta,c_1,c_2) \in \mathcal{L}_y \mid \alpha \in S,\ \beta \in S\}; \tag{4.159}$$

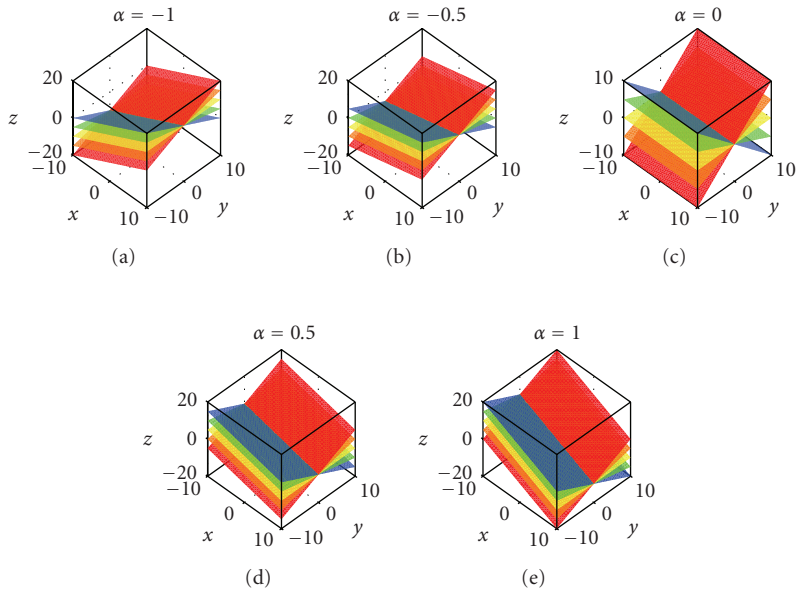


FIGURE 4.8. x-planes for various values of α and β.

(iii) discrete z-lines

$$\mathcal{L}_z^d = \{l_z(\alpha,\beta,c_1,c_2) \in \mathcal{L}_z \mid \alpha \in S,\ \beta \in S\}. \tag{4.160}$$

The set $\mathcal{L}^d$ is defined by

$$\mathcal{L}^d = \mathcal{L}_x^d \cup \mathcal{L}_y^d \cup \mathcal{L}_z^d. \tag{4.161}$$

By using the lines in $\mathcal{L}^d$ we define the discrete X-ray transform for discrete images as the following.

Definition 4.6.4. For an image I and a line $l(\alpha,\beta,c_1,c_2) \in \mathcal{L}^d$, the discrete X-ray transform is given by

$$PI(l) = \begin{cases} P_x I(l), & l \in \mathcal{L}_x^d, \\ P_y I(l), & l \in \mathcal{L}_y^d, \\ P_z I(l), & l \in \mathcal{L}_z^d, \end{cases} \tag{4.162}$$

where P_x, P_y, and P_z are defined by (4.147)–(4.149), respectively.

Definition 4.6.4 defines the discrete X-ray transform for discrete images by using a discrete set of lines. This transform is not defined for all lines in $\mathbb{R}^3$, but only for lines in $\mathcal{L}^d$. We will show in Section 4.6.6 that for the set of lines $\mathcal{L}^d$ the discrete X-ray transform can be computed using a fast algorithm. Moreover, we will show in Section 4.5.7 that the discrete X-ray transform is invertible.

Since the Fourier slice theorem (Theorem 4.6.3) holds for continuous slopes (α, β), it holds in particular for the discrete set of slopes defined by (4.157). Substituting the discrete set of slopes, given by (4.157), into Theorem 4.6.3 gives the discrete Fourier slice theorem, which is defined for both discrete images and a discrete set of directions.

Corollary 4.6.5 (discrete Fourier slice theorem). *Let S be the set given in* (4.157) *and let $\hat{I}$ be the trigonometric polynomial defined by*

$$\hat{I}(\xi_1, \xi_2, \xi_3) = \sum_{u=-n/2}^{n/2-1} \sum_{v=-n/2}^{n/2-1} \sum_{w=-n/2}^{n/2-1} I(u, v, w) e^{-2\pi\iota\xi_1 u/m} e^{-2\pi\iota\xi_2 v/m} e^{-2\pi\iota\xi_3 w/m}. \tag{4.163}$$

Then,

(i) *for a given family of x-lines $l_x(\alpha, \beta)$, $\alpha = 2p/n \in S_1$, $\beta = 2q/n \in S_2$,*

$$\hat{P}_{x(2p/n,2q/n)} I(k, l) = \hat{I}\left(\frac{-2pk}{n} - \frac{2ql}{n}, k, l\right), \tag{4.164}$$

(ii) *for a given family of y-lines $l_y(\alpha, \beta)$, $\alpha = 2p/n \in S_1$, $\beta = 2q/n \in S_2$,*

$$\hat{P}_{y(2p/n,2q/n)} I(k, l) = \hat{I}\left(k, \frac{-2pk}{n} - \frac{2ql}{n}, l\right), \tag{4.165}$$

(iii) *for a given family of z-lines $l_z(\alpha, \beta)$, $\alpha = 2p/n \in S_1$, $\beta = 2q/n \in S_2$,*

$$\hat{P}_{z(2p/n,2q/n)} I(k, l) = \hat{I}\left(k, l, \frac{-2pk}{n} - \frac{2ql}{n}\right). \tag{4.166}$$

4.6.6. Computing the discrete X-ray transform

Equation (4.162) shows that direct computation of the discrete X-ray transform according to its definition requires $O(n^7)$ operations. As we will shortly see, by utilizing the frequency domain relations between the samples of the discrete X-ray transform, it is possible to compute it in $O(n^4 \log n)$ operations without sacrificing the accuracy. This is quite a remarkable result if we consider the fact that the lower bound for such a computation is $\Omega(n^4)$ operations, since there are four independent parameters $(\alpha, \beta, c_1, c_2)$ to consider.

We consider only the computation of the discrete X-ray transform for $\mathcal{L}_x^d$, given by (4.158). The algorithm for computing the discrete X-ray transform for

$\mathcal{L}_y^d$ and $\mathcal{L}_z^d$ is similar. The discrete Fourier slice theorem for an x-line l_x (4.164) is given by

$$\hat{P}_{x(2p/n,2q/n)}I(k,l) = \hat{I}\left(\frac{-2pk}{n} - \frac{2ql}{n}, k, l\right), \quad p,q \in \left\{\frac{n}{2}, \ldots, \frac{n}{2}\right\}. \tag{4.167}$$

If we can rapidly sample the trigonometric polynomial $\hat{I}$, given by $\hat{I}(\xi_1,\xi_2,\xi_3) = \sum_{u=-n/2}^{n/2-1}\sum_{v=-n/2}^{n/2-1}\sum_{w=-n/2}^{n/2-1} I(u,v,w)e^{-2\pi\imath\xi_1 u/m}e^{-2\pi\imath\xi_2 v/m}e^{-2\pi\imath\xi_3 w/m}$, at the points $(-2pk/n - 2ql/n, k, l)$ for some fixed p and q, then, by the 2D inverse DFT we can recover the values of $P_{x(2p/n,2q/n)}I(c_1,c_2)$ for fixed p and q. Hence, once we compute the samples $\hat{P}_{x(2p/n,2q/n)}I(k,l)$ for all possible values of p, q, k and l, it requires $(n+1)^2$ applications of the 2D inverse DFT (one for each pair of p and q) to recover P_xI for all x-lines $l_x \in \mathcal{L}_x^d$. This results in a total of $O(n^4 \log n)$ operations to recover P_xI from $\hat{P}_xI$. Therefore, it remains to show that we can compute $\hat{P}_{x(2p/n,2q/n)}I(k,l)$ for all p, q, k, and l using $O(n^4 \log n)$ operations.

We take some fixed slope $\alpha = 2p/n \in S$ and denote

$$\hat{I}_p(q,k,l) = \hat{I}\left(\frac{-2pk}{n} - \frac{2ql}{n}, k, l\right). \tag{4.168}$$

By expanding (4.168) we obtain

$$\hat{I}_p(q,k,l) = \hat{I}\left(\frac{-2pk}{n} - \frac{2ql}{n}, k, l\right) \tag{4.169}$$

$$= \sum_{u,v,w=-n/2}^{n/2-1} I(u,v,w)e^{-2\pi\imath u(-2pk/n-2ql/n)/m}e^{-2\pi\imath kv/m}e^{-2\pi\imath lw/m}. \tag{4.170}$$

Denote

$$\hat{I}_{yz}(u,k,l) = \sum_{v=-n/2}^{n/2-1}\sum_{w=-n/2}^{n/2-1} I(u,v,w)e^{-2\pi\imath kv/m}e^{-2\pi\imath lw/m}. \tag{4.171}$$

By substituting (4.171) into (4.170) we obtain

$$\hat{I}_p(q,k,l) = \sum_{u=-n/2}^{n/2-1} \hat{I}_{yz}(u,k,l)e^{-2\pi\imath u(-2pk/n-2ql/n)/m} \tag{4.172}$$

or

$$\hat{I}_p(q,k,l) = \sum_{u=-n/2}^{n/2-1} \hat{I}_{yz}(u,k,l)e^{-2\pi\imath u\omega_q}, \tag{4.173}$$

where

$$\Delta\omega = \frac{-2l}{(nm)},$$
$$\omega_0 = \frac{-2pk}{(nm)}, \tag{4.174}$$
$$\omega_q = \omega_0 + q\Delta\omega.$$

If we compute $\hat{I}_p(q,k,l)$, given in (4.173), for all values of q, k, and l, then, by (4.167) and (4.168) we can compute $\hat{P}_{x(2p/n,2q/n)}I(k,l)$ for a fixed direction p. Repeating the process for all possible values of p produces the values of $\hat{P}_{x(2p/n,2q/n)}I(k,l)$ for all possible p, q, k, and l. Hence, rapid evaluation of (4.173) enables rapid computation of the discrete X-ray transform.

Equations (4.173) and (4.174) reveal a special relation between the samples of $\hat{I}_p(q,k,l)$. As we will see in Section 4.6.6.1, we can utilize this relation to rapidly compute the values of $\hat{I}_p(q,k,l)$ by using the chirp Z-transform. We first introduce the chirp Z-transform, and later show how to use it to rapidly compute the discrete X-ray transform.

4.6.6.1. The chirp Z-transform

Given a sequence $x(j)$, $j = -n/2,\ldots,n/2-1$, its Z-transform is defined by

$$X(z) = \sum_{j=-n/2}^{n/2-1} x(j)z^{-j}. \tag{4.175}$$

The chirp Z-transform, first discussed in [28], rapidly computes $X(z_k)$ for points $z_k = AW^{-k}$, where $A, W \in \mathbb{C}$. Specifically, the chirp Z-transform allows to compute $X(z_k)$ along contours of the form

$$z_k = e^{2\pi\imath w_k}, \quad \omega_k = \omega_0 + k\Delta\omega, \quad k = \frac{-n}{2},\ldots,\frac{n}{2}, \tag{4.176}$$

where ω_0 is an arbitrary starting frequency and $\Delta\omega$ is an arbitrary frequency increment. For the contour defined by (4.176), the chirp Z-transform has the form

$$X\left(e^{2\pi\imath\omega_k}\right) = \sum_{j=-n/2}^{n/2-1} x(j)e^{-2\pi\imath j\omega_k}, \quad k = \frac{-n}{2},\ldots,\frac{n}{2}. \tag{4.177}$$

For the case where $\omega_0 = 0$ and $\Delta\omega = 1/n$, the chirp Z-transform in (4.177) computes the discrete Fourier transform of the sequence $x(j)$.

The algorithm described in [29, 30] computes the chirp Z-transform of a sequence $x(j)$ of length n and arbitrary ω_0 and $\Delta\omega$ using $O(n\log n)$ operations.

Equations (4.173) and (4.174) state that for fixed k and l, $\hat{I}_p(q,k,l)$ can be rapidly computed by setting

$$\begin{aligned} x(j) &= \hat{I}_{yz}(j,k,l), \\ \omega_0 &= \frac{-2pk}{(nm)}, \\ \Delta\omega &= \frac{-2l}{(nm)} \end{aligned} \tag{4.178}$$

and using the chirp Z-transform. These settings are used in the next section for the rapid computation of the discrete X-ray transform.

4.6.7. Fast algorithm for the computation of the discrete X-ray transform

We will use the chirp Z-transform algorithm from Section 4.6.6.1 to rapidly compute $\hat{P}I(l)$ for all $l \in \mathcal{L}^d$, where $\mathcal{L}^d$ is defined in (4.161). The algorithm consists of three phases, which compute $\hat{P}_xI$, $\hat{P}_yI$, and $\hat{P}_zI$ for lines in $\mathcal{L}^d_x$, $\mathcal{L}^d_y$, and $\mathcal{L}^d_z$, respectively. We present only the algorithm for computing $\hat{P}_xI$. The algorithms for $\hat{P}_yI$ and $\hat{P}_zI$ are similar.

We use the following notation in the description of the algorithm.

(i) E_y, E_z—Extension operators, which symmetrically zero-pad the image I to length $2n+1$ along the y- and z-directions, respectively.

(ii) F_y, F_z—1D discrete Fourier transform (FFT) along the specified direction. For example, F_yI takes all the vectors $I(x,\cdot,z)$ and applies on them the 1D FFT.

(iii) $\mathrm{CZT}(x,\omega_0,\Delta\omega)$—the chirp Z-transform, defined in Section 4.6.6.1, with parameters ω_0 and $\Delta\omega$. Specifically, $\mathrm{CZT}(x,\omega_0,\Delta\omega)$ is defined by

$$\mathrm{CZT}(x,\omega_0,\Delta\omega)_k = \sum_{j=-n/2}^{n/2-1} x(j)e^{-2\pi\imath j\omega_k}, \quad \omega_k = \omega_0 + k\Delta\omega,\ k = \frac{-n}{2},\ldots,\frac{n}{2}. \tag{4.179}$$

4.6.7.1. Algorithm description

Algorithm 4.4 is applied. The output of Algorithm 4.4 is stored in the array Res_x of size $(n+1)\times(n+1)\times m\times m$, $(m=2n+1)$.

4.6.7.2. Correctness of Algorithm 4.4

Theorem 4.6.6. *Upon termination of Algorithm 4.4 the following holds:*

$$\mathrm{Res}_x(p,q,k,l) = \hat{P}_{x(2p/n,2q/n)}I(k,l). \tag{4.180}$$

• Computing $\widehat{P}_x I$:
(1) $\dot{I} = E_y E_z I$
(2) $\widetilde{I} = F_y F_z \dot{I}$
(3) foreach p in $-n/2, \ldots, n/2$
(4) foreach k, l in $-n, \ldots, n$
(5) $x_{k,l} \leftarrow \widetilde{I}(\cdot, k, l)$ ($x_{k,l}$ is a sequence of length n)
(6) $\omega_0 \leftarrow -2pk/(nm)$, $\Delta\omega \leftarrow -2l/(nm)$
(7) $\mathrm{Res}_x(p, \cdot, k, l) = \mathrm{CZT}(x_{k,l}, \omega_0, \Delta\omega)$
(8) endfor
(9) endfor

Algorithm 4.4

4.6.7.3. Complexity of computing the discrete X-ray transform (Algorithm 4.4)

The complexity of computing $\widehat{P}_x I$ (Algorithm 4.4) is analyzed. The complexity of computing $\widehat{P}_y I$ and $\widehat{P}_z I$ is the same.

Step (1) of Algorithm 4.4 requires $O(n^3)$ operations as it doubles the size of a 3D image of size n^3 by zero-padding each direction. Step (2) requires the application of $O(n^2)$ 1D FFTs along the z-direction and $O(n^2)$ 1D FFTs along the y-direction. Since each FFT application requires $O(n \log n)$ operations, this accounts to a total of $O(n^3 \log n)$ operations.

Next, for fixed k, l, and p, steps (5)–(7) require $O(n \log n)$ operations, since the most expensive operation is to compute the CZT (chirp Z-transform) in step (7), which requires $O(n \log n)$ operations. This accounts to a total of $O(n^4 \log n)$ operations for the processing of all values of k, l, and p. Hence, computing $\widehat{P}_x I$ requires $O(n^4 \log n)$ operations.

Note that Algorithm 4.4 computes $\widehat{P}_x I$ for all directions p and q. If for some application not all directions are needed, they can be discarded from the computation, reducing the complexity of Algorithm 4.4.

4.7. Summary

As a summary, we want to mention some more algorithms and applications that we believe are important but were omitted because of space limitation. These algorithms and papers are based on the irregular sampling for multidimensional polar processing of integral transforms that is introduced in this chapter.

In the discrete diffraction transform paper [11, 19], a discrete analogue of the continuous diffracted projection is defined. It defines a discrete diffracted transform (DDT) as a collection of the discrete diffracted projections taken at specific set of angles along specific set of lines. The "discrete diffracted projection" is defined to be a discrete transform that is similar in its properties to the continuous diffracted projection. The paper proves that when the DDT is applied on a set of samples of a continuous object, it approximates a set of continuous vertical

diffracted projections of a horizontally sheared object and a set of continuous horizontal diffracted projections of a vertically sheared object. A similar statement, where diffracted projections are replaced by the X-ray projections, holds in the case of the discrete 2D Radon transform (DRT). Thus, the discrete diffraction transform is rapidly computable and invertible. Some of the underlying ideas came from the definition of DRT.

In [13], a fast high accuracy Polar FFT was developed. For a given two-dimensional signal of size $N \times N$, the proposed algorithm's complexity is $O(N^2 \log N)$, just like in a Cartesian 2D-FFT. A special feature of our approach is that it involves only 1D equispaced FFT's and 1D interpolations. A central tool in our approach is the pseudopolar FFT, an FFT where the evaluation frequencies lie in an oversampled set of nonangularly equispaced points. The pseudopolar FFT plays the role of a halfway point, a nearly polar system from which conversion to polar coordinates uses processes relying purely on 1D FFTs and interpolation operations.

Two algorithms for the reconstruction of a 2D object from its continuous projections are proposed. The first algorithm operates on parallel projection data, while the second uses the more practical model of fan-beam projections. Both algorithms are based on the discrete Radon transform [5], which extends the continuous Radon transform to discrete data. The discrete Radon transform and its inverse can be computed in a complexity comparable with the 2D FFT, and are shown to accurately model the continuum as the number of samples increases. Numerical results demonstrate high quality reconstructions for both parallel and fan-beam acquisition geometries. This is a work in progress [20]. This method is now being considered to be extended to 3D fan-beam and to discrete spiral Fourier transform.

Here is a short list of applications: image registration [14, 15], volume registration [16], 3D registration, 2D symmetry detection [17]. This list is far from being exhaustive.

Bibliography

[1] S. R. Deans, *The Radon Transform and Some of Its Applications*, John Wiley & Sons, New York, NY, USA, 1983.

[2] F. Natterer, *The Mathematics of Computerized Tomography*, John Wiley & Sons, New York, NY, USA, 1989.

[3] F. Natterer and F. Wubbeling, *Mathematical Methods in Image Reconstraction*, SIAM, Philadelphia, Pa, USA, 2001.

[4] A. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*, IEEE Press, New York, NY, USA, 1988.

[5] A. Averbuch, R. Coifman, D. L. Donoho, M. Israeli, and Y. Shkolnisky, "A framework for discrete integral transformations I - the pseudo-polar Fourier transform," submitted.

[6] A. Averbuch, R. Coifman, D. L. Donoho, M. Israeli, I. Sedelnikov, and Y. Shkolnisky, "A framework for discrete integral transformations II - the 2D discrete Radon transform," submitted.

[7] A. Averbuch, R. Coifman, D. L. Donoho, M. Israeli, and Y. Shkolnisky, "Fast slant stack: a notion of Radon transform for data in a cartesian grid which is rapidly computable, algebraically exact, geometrically faithful and invertible," Research Report, Stanford University, Stanford, Calif, USA, 2001.

[8] A. Averbuch and Y. Shkolinsky, "3D Fourier based discrete Radon transform," *Applied Computational Harmonic Analysis*, vol. 15, no. 1, pp. 33–69, 2003.

[9] A. Averbuch and Y. Shkolnisky, "Multidimensional discrete Radon transform," in *Wavelets and Their Applications*, M. Krishna, R. Radha, and S. Thangavelu, Eds., pp. 63–88, Allied, Anna University, Chennai, India, 2003.

[10] A. Averbuch and Y. Shkolnisky, "3D discrete X-ray transform," *Applied and Computational Harmonic Analysis*, vol. 17, no. 3, pp. 259–276, 2004.

[11] I. Sedelnikov, A. Averbuch, and Y. Shkolnisky, "The discrete diffraction transform," submitted.

[12] A. G. Flesia, H. Hel-Or, A. Averbuch, E. J. Candes, R. Coifman, and D. Donoho, "Digital implementation of ridgelet packets," in *Beyond Wavelets*, J. Stockler and G. Welland, Eds., pp. 31–60, Academic Press, San Diego, Calif, USA, 2003.

[13] A. Averbuch, R. Coifman, D. L. Donoho, M. Elad, and M. Israeli, "Fast and accurate polar Fourier transform," *Journal on Applied and Computational Harmonic Analysis*, vol. 21, pp. 145–167, 2006.

[14] Y. Keller, A. Averbuch, and M. Israeli, "Pseudo-polar-based estimation of large translations, rotations, and scalings in images," *IEEE Transactions on Image Processing*, vol. 14, no. 1, pp. 12–22, 2005.

[15] Y. Keller, Y. Shkolnisky, and A. Averbuch, "The angular difference function and its application to image registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 969–976, 2005.

[16] Y. Keller, Y. Shkolnisky, and A. Averbuch, "Volume registration using 3-D pseudo-polar Fourier transform," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4323–4331, 2006.

[17] Y. Keller and Y. Shkolnisky, "A signal processing approach to symmetry detection," *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2198–2207, 2006.

[18] Y. Shkolnisky, *Irregular sampling for multidimensional polar processing of integral transforms and prolate spheroidal wave functions*, Ph.D. thesis, Tel Aviv University, Tel Aviv, Israel, 2005.

[19] I. Sedelnikov, "The discrete diffraction transform," M.S. thesis, Tel Aviv University, Tel Aviv, Israel, 2003.

[20] A. Averbuch, I. Sedelnikov, and Y. Shkolnisky, "CT reconstraction from parallel and fan-beam projections by 2D discrete Radon transform," submitted.

[21] P. N. Swarztrauber and D. H. Bailey, "The fractional Fourier transform and applications," *SIAM Review*, vol. 33, no. 3, pp. 389–404, 1991.

[22] L. R. Rabiner, R. W. Schafer, and C. M. Rader, "The chirp z-transform algorithm and its applications," *Bell System Technical Journal*, vol. 48, no. 5, pp. 1249–1292, 1969.

[23] R. M. Mersereau and A. V. Oppenheim, "Digital reconstruction of multidimensional signals from their projections," *Proceedings of the IEEE*, vol. 62, no. 10, pp. 1319–1338, 1974.

[24] J. E. Pasciak, "A note on the Fourier algorithm for image reconstruction," preprint, AMD 896Applied Mathematics Department, Brookhaven National Laboratory, Upton, NY, USA, 1973, 1981.

[25] P. Edholm and G. T. Herman, "Linograms in image reconstraction from projections," *IEEE Transactions on Medical Imaging*, vol. 6, no. 4, pp. 301–307, 1987.

[26] P. R. Edholm, G. T. Herman, and D. A. Roberts, "Image reconstruction from linograms: implementation and evaluation," *IEEE Transactions on Medical Imaging*, vol. 7, no. 3, pp. 239–246, 1988.

[27] W. Lawton, "A new polar Fourier transform for computer-aided tomography and spotlight synthetic aperture radar," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 6, pp. 931–933, 1988.

[28] L. R. Rabiner, R. W. Schafer, and C. M. Rader, "The chirp z-transform algorithm," *IEEE Transcations on Audio and Electroacoustics*, vol. 17, no. 2, pp. 86–92, 1969.

[29] L. R. Rabiner and B. Gold, *Theory and Applications of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1975.

[30] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1998.

[31] G. H. Golub and C. F. Van Loan, *Matrix Computation*, Johns Hopkins Series in the Mathematical Sciences, The Johns Hopkins University Press, Baltimore, Md, USA, 1984.

A. Averbuch: The School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel

Email: amir@math.tau.ac.il

R. Coifman: Program of Applied Mathematics, Department of Mathematics, Yale University, P.O. Box 208283, New Haven, CT 06520-8283, USA

Email: coifman@math.yale.edu

M. Israeli: Faculty of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel

Email: israeli@cs.technion.ac.il

I. Sedelnikov: The School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel

Email: ilyas@post.tau.ac.il

Y. Shkolnisky: Program of Applied Mathematics, Department of Mathematics, Yale University, P.O. Box 208283, New Haven, CT 06520-8283, USA

Email: yoel.shkolnisky@yale.edu

5 Space-variant and adaptive transform domain image restoration methods

L. Yaroslavsky

A family of space-time variant local adaptive transform domain methods for signal, image, and video processing (denoising, deblurring, enhancement) is described. The methods work in a space-time moving window in the domain of an orthogonal transform, and in each position of the window, nonlinearly modify the signal transform coefficients to obtain an estimate of the window central pixel. For the design of the transform coefficient nonlinear processing algorithm, minimum RMS restoration error approach is applied. This leads to transform-domain adaptive empirical Wiener filtering in form of the coefficient "soft" or "hard" thresholding. It is shown that among different possible transforms, discrete cosine transform proved to be the primer candidate for using in image and video processing. Very good edge-preserving noise-suppression capability of sliding window DCT domain (SWDCT) image denoising algorithms was confirmed experimentally on test and real-life images and was compared to that of the local "ideal" Wiener filtering, which they approximate. Applications of the algorithms for image denoising, deblurring, and enhancement are illustrated by numerous examples.

Theoretically, sliding window transform-domain filtering methods can be treated as an implementation of image subband decomposition and nonlinear pointwise transformation of the subband components. Being considered in this way, they parallel wavelet denoising methods that exploit multiresolution property of wavelet transforms for enabling local adaptivity of the filtering. Image denoising capabilities of both families of image denoising methods are compared in experiments with test and real-life images and 1D signals, which revealed the superiority of the SWDCT filtering in this respect. As a way to efficiently implement parallel SWDCT processing in windows of different sizes and to solve in this way the problem of selecting appropriate window size in SWDCT filtering, combining wavelet image multiresolution decomposition and SWDCT filtering in a hybrid processing is considered, and further improvement of filtering performance is demonstrated in extensive simulation experiments.

5.1. Introduction

Image and video restoration (denoising and deblurring) is a fundamental image processing task. Its successful solution is critical in many applications and especially when images are to be analyzed by a human operator. In spite of many efforts, this task is still challenging researchers. The fundamental difficulty is space/time nonstationarity of images and videos that requires local space/time adaptivity of the restoration algorithms.

In this chapter, we describe a family of easy-to-use sliding window transform-domain image and video restoration algorithms capable of local adaptivity. The algorithms work in a sliding window in the domain of an orthogonal transform, and in each position of the window, nonlinearly modify the signal transform coefficients to obtain an estimate of the window central pixel. Being initially designed as DFT-domain filters (see [1]), they implement the old idea of space (time)-frequency signal representation (see [2, 3]) and were soon extended to the use of other transforms, and, first of all, of DCT (see [4–6]) and to multicomponent image processing (see [7, 8]).

More recently, another family of transform-domain denoising methods capable of spatial adaptivity emerged, which are known, after Donoho, as wavelet shrinkage methods (see [9–11]). Basis functions of wavelet transforms (wavelets) are formed by means of a combination of two basic methods of building transform basis functions, shifting, and scaling ones. Thanks to this, wavelets combine local sensitivity of shift basis functions and global properties of "scaled" basis functions and feature such attractive properties as multiresolution and good localization in both signal and transform domains.

Both running window transform domain and wavelet processing are different implementations of linear filtering in transform domain. This motivates an attempt to suggest their unified interpretation that would provide an insight into their similarities and dissimilarities. This issue is also addressed in the chapter. It is shown that both filter families can be treated as special cases of signal subband decomposition and empirical Wiener filtering of the subbands. We also compare noise suppression capability of these methods and introduce a hybrid method that combines advantages of both methods.

The chapter is arranged as follows. In Section 5.2, principles of empirical Wiener scalar linear filtering in transform domain and its application to signal and image restoration are described. In Section 5.3, local adaptive sliding window filtering is introduced, substantiated both in terms of signal restoration capability and computational complexity, and its efficiency in signal, image, and video denoising, deblurring, and enhancement is demonstrated. Then, in Section 5.4, wavelet signal denoising methods are briefly reviewed, in Section 5.4.1, sliding window and wavelet methods are compared in terms of their signal restoration capability and hybrid wavelet/sliding window filters are introduced that combine advantages of both methods. In Section 5.5, an approach is suggested to a unified representation and treatment of sliding window, wavelet, and hybrid filtering methods as implementations of signal subband decomposition processing. In

conclusion, the results are summarized and possible new developments are briefly discussed.

5.2. MSE optimal scalar linear filters for signal restoration

5.2.1. A theoretical framework

In this section, we remember basic principles of mean-square-error (MSE-) optimal scalar linear filtering in transform domain (see [12, 13]). For the generality, we will assume that signals to be processed are multicomponent and are represented by their sampled components (such as, e.g., R, G, and B components of color images or successive frames of video sequences).

Let $\mathbf{a} = \{a_k^{(m)}\}$ be component vectors of samples of a "true" signal, estimation of which is the processing goal, and $\hat{\mathbf{a}} = \{\hat{a}_k^{(m)}\}$ are their estimates obtained in the result of processing, $\{k\}$ and $\{m\}$ are image sample and component indices, M is the number of image components. Note that for the sake of simplicity, we use 1D denotation for image sample indices. Let us assume also that signal processing quality is evaluated in terms of mean square restoration error (MSE) computed as squared difference between true signal $\mathbf{a} = \{a_k^{(m)}\}$ and its estimation $\hat{\mathbf{a}} = \{\hat{a}_k^{(m)}\}$ averaged over the set of N signal samples and over random factors involved in the task such as for instance, random signals and noise:

$$\mathrm{MSE} = \mathrm{AV}\left\{\sum_{m=1}^{M}\sum_{k=0}^{N-1} \mid a_k^{(m)} - \hat{a}_k^{(m)} \mid^2\right\}, \tag{5.1}$$

where AV is the averaging operator. In this assumption, the processing is aimed at minimization of the MSE:

$$\{\hat{a}_k^{(m)}\} = \underset{\{b_k^{(m)}\}\Rightarrow\{a_k^{(m)}\}}{\arg\min}\ \mathrm{AV}\left\{\sum_{m=1}^{M}\sum_{k=0}^{N-1} \mid a_k^{(m)} - \hat{a}_k^{(m)} \mid^2\right\} \tag{5.2}$$

by means of appropriate selection of observation-to-estimation mappings $\{b_k^{(m)}\}\Rightarrow \{\hat{a}_k^{(m)}\}$, where $\{b_k^{(m)}\}$ are samples of signal under processing.

To make the design of the processing constructive, one should parameterize the observation-to-estimation mapping. To this goal, we will make two additional assumptions. First, we restrict ourselves with the processing implemented as signal linear filtering:

$$\hat{\mathbf{a}} = \mathbf{H}\mathbf{b}, \tag{5.3}$$

where $\mathbf{H}$ is a linear filter matrix and $\mathbf{b}$ is a vector of samples of the input signal. For a signal with N samples, the filter design requires, in general, specification of N^2 elements of matrix $\mathbf{H}$. The filter matrix specification is much simplified if the filter matrix is a diagonal one that requires specification of only N diagonal elements. We will call such an implementation of signal linear filtering *scalar filtering*.

Using scalar filters, one can approximate a general filter matrix $\mathbf{H}$ by a diagonal matrix $\mathbf{H}_d$ in the following way:

$$\mathbf{H} \approx \mathbf{T}^{-1}\mathbf{H}_d\mathbf{T}, \tag{5.4}$$

where $\mathbf{T}$ and $\mathbf{T}^{-1}$ are direct and inverse matrices of an orthogonal transform that approximates the transform that diagonalizes matrix $\mathbf{H}$. Such an approximation means that the filtering is carried out in the domain of the transform $\mathbf{T}$. This is the second assumption that we accept. This assumption not only lowers computational complexity of the filter design, but also simplifies incorporating in the filter design a priori knowledge regarding signals under processing. While formulation of this information in the signal domain is frequently problematic, it is much simplified in the domain of orthogonal transforms since signal spectra in the transform domain exhibit statistically much more regular behavior than that of the proper signal provided that the transform is appropriately selected (recall, e.g., the behavior of image DFT or DCT spectra that are known to more or less rapidly decay with the rise of the frequency index).

For scalar filtering, filter matrix is specified by its diagonal elements:

$$\mathbf{H} = \{\eta_{r,\mu}\}, \tag{5.5}$$

where r and μ are indices in the transform domain that correspond to signal sample indices $\{k\}$ and $\{m\}$. Thus, we will assume in what follows scalar filtering and will design filters that operate in the following 3 steps:

(1) computing spectral coefficients $\{\beta_{r,\mu}\}$ of the observed signal $\mathbf{b}$ over the chosen orthogonal transform $\mathbf{T}$;

(2) multiplication of the obtained transform coefficients $\{\beta_{r,\mu}\}$ by the filter coefficients $\{\eta_{r,\mu}\}$ to obtain estimates of the processed signal spectral coefficients $\{\hat{\alpha}_{r,\mu}\}$ as

$$\{\hat{\alpha}_{r,\mu} = \eta_{r,\mu}\beta_{r,\mu}\}; \tag{5.6}$$

(3) inverse transformation $\mathbf{T}^{-1}$ of the output signal spectral coefficients $\{\hat{\alpha}_{r,\mu}\}$ to obtain estimated samples $\{\hat{a}_k^{(m)}\} = \mathbf{T}^{-1}\{\hat{\alpha}_{r,\mu}\}$ of the signal.

With this approach, the synthesis of filters is reduced to the determination of $N \times M$ filter coefficients $\{\eta_{r,\mu}\}$. For the MSE-optimal filter design in the domain of an orthogonal transform, one can, by virtue of Parceval's relation, reformulate the criterion of (5.2) in terms of signal transform coefficients:

$$\mathrm{MSE} = \mathrm{AV}\left\{\sum_{m=1}^{M}\sum_{k=0}^{N-1} \left|\alpha_{r,\mu} - \eta_{r,\mu}\beta_{r,\mu}\right|^2\right\}. \tag{5.7}$$

By minimizing the MSE with respect to $\{\eta_{r,\mu}\}$, one can immediately find that the optimal values of the coefficients of the filter that minimize mean square filtering

error (5.7) are defined by the following equation:

$$\eta_{r,\mu} = \frac{\mathrm{AV}\{\alpha_{r,\mu}\beta^*_{r,\mu}\}}{\mathrm{AV}\{|\beta_{r,\mu}|^2\}} \tag{5.8}$$

with asterisk $*$ denoting complex conjugate. The design of the filter of (5.8) is therefore reduced to an estimation of power spectrum of the input signal $\mathrm{AV}\{|\beta_{r,\mu}|^2\}$ and its mutual spectrum $\mathrm{AV}\{\alpha_{r,\mu}\beta^*_{r,\mu}\}$ with the true signal.

5.2.2. Adaptive filters for signal restoration with empirical estimation of filter parameters

Assume that signal distortions can be modeled by the equation

$$\mathbf{b} = \mathbf{La} + \mathbf{n}, \tag{5.9}$$

where $\mathbf{L}$ is a linear operator of the imaging system and $\mathbf{n}$ is a random zero-mean signal independent random vector that models imaging system's noise. Assume also that the imaging system operator $\mathbf{L}$ is such that the distorted signal can be described, in the domain of the chosen orthogonal transform, by the following relationship:

$$\beta_{r,\mu} = \lambda_{r,\mu}\alpha_{r,\mu} + \nu_{r,\mu}, \tag{5.10}$$

where $\{\lambda_{r,\mu}\}$ are representation coefficients of the linear operator $\mathbf{L}$ in the domain of the orthogonal transform and $\{\nu_{r,\mu}\}$ are zero-mean spectral coefficients of the realization of the noise interference. Then one can obtain from (5.8) that optimal restoration filter coefficients are defined as

$$\eta_{r,\mu} = \frac{1}{\lambda_{r,\mu}} \frac{|\lambda_{r,\mu}|^2\,\mathrm{AV}\{|\alpha_{r,\mu}|^2\}}{\mathrm{AV}\{|\beta_{r,\mu}|^2\}} = \frac{1}{\lambda_{r,\mu}} \frac{|\lambda_{r,\mu}|^2\,\mathrm{AV}\{|\alpha_{r,\mu}|^2\}}{|\lambda_{r,\mu}|^2\,\mathrm{AV}\{|\alpha_{r,\mu}|^2\} + \mathrm{AV}\{|\nu_{r,\mu}|^2\}}. \tag{5.11}$$

Filter of (5.11) is referred to as the *Wiener scalar filter*. Its design assumes knowledge of statistical average (in terms of the averaging operator AV) power spectra $\mathrm{AV}\{|\alpha_{r,\mu}|^2\}$ and $\mathrm{AV}\{|\nu_{r,\mu}|^2\}$ of image and noise.

Noise power spectrum $\mathrm{AV}\{|\nu_{r,\mu}|^2\}$ can either be known from the imaging system design specification or it can be estimated from observed noisy images (see [4, 12, 13]). For evaluation of the true signal power spectrum $\mathrm{AV}\{|\alpha_{r,\mu}|^2\}$, there are two similar options: (i) it can be evaluated in advance from either an a priori image statistical model or from an image database and (ii) one can attempt to estimate it directly from the observed signal/image. The first option constitutes a conventional approach to signal restoration that dates back to classical works by N. Wiener. The second option leads to adaptive filters that provide the best restoration result for a particular image and particular realization of noise.

Such an empirical spectrum estimation can be carried out using the relationship

$$|\lambda_{r,\mu}|^2 \,\mathrm{AV}\{|\alpha_{r,\mu}|^2\} = \mathrm{AV}\{|\beta_{r,\mu}|^2\} - \mathrm{AV}\{|\nu_{r,\mu}|^2\} \tag{5.12}$$

that follows from the image and noise model of (5.10). In this relationship, $\mathrm{AV}\{|\beta_{r,\mu}|^2\}$ is the estimate of the observed image power spectrum, for which one can use one or another known procedure of spectra estimation by means of smoothing spectrum of the signal being processed.

As a zero-order approximation, the observed power signal/image spectrum $\{|\beta_{r,\mu}|^2\}$ can be used as an estimate of the averaged one $\mathrm{AV}\{|\beta_{r,\mu}|^2\}$. In this way, we arrive at the following implementation of filters for signal denoising and deblurring (see [6–8, 12, 13]):

$$\eta_{r,\mu} \cong \max\left\{0, \frac{1}{\lambda_{r,\mu}} \frac{\{|\beta_{r,\mu}|^2\} - \mathrm{AV}\{|\nu_{r,\mu}|^2\}}{\{|\beta_{r,\mu}|^2\}}\right\}, \tag{5.13}$$

where zero values are to be taken wherever difference between image and noise power spectra gets negative, due to spectra estimation errors. We will refer to this filter as the *empirical Wiener filter*.

The two following modifications of this filter are of the most practical interest:

(i) "rejective" filter.

$$\eta_{r,\mu} \cong \begin{cases} \dfrac{1}{\lambda_{r,\mu}} & \text{if } |\beta_{r,\mu}|^2 > \mathrm{THR}_{r,\mu},\ \lambda_{r,\mu} \neq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{5.14}$$

where the values of $\mathrm{THR}_{r,\mu}$ are preassigned and are associated with power spectrum $\mathrm{AV}\{|\nu_{r,\mu}|^2\}$ of the additive noise;

(ii) "fractional spectrum filter".

$$\eta_{r,\mu} \cong \begin{cases} g\dfrac{1}{\lambda_{r,\mu}} |\beta_{r,\mu}|^{P-1} & \text{if } |\beta_{r,\mu}|^2 > \mathrm{THR}_{r,\mu},\ \lambda_{r,\mu} \neq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{5.15}$$

with $P \leq 1$ as a spectrum enhancement parameter and g as a normalization parameter. When $P = 1$, the filter is equivalent to that of (5.14). Selection of $P < 1$ results in redistribution of signal energy in favor of less intensive (most frequently, higher frequency) spectrum components. The latter modification is useful for image blind deblurring and image enhancement.

An important feature of filters defined by (5.13)–(5.15) is their adaptivity, which is implied by the fact that filter parameters are defined by individual realizations of signals.

5.3. Sliding window local adaptive filters

5.3.1. "Local" versus "global" processing

Described filers can be used in two fashions: filtering can either be carried out over the entire set of available signal samples such as image frames (globally) or fragmentwise (locally).

At least for image processing, there are quite a number of arguments in favor of "local" processing versus "global" one.

(i) "Global" processing assumes signal spatial "stationarity" or homogeneity. One hardly can regard such signals as images "stationary." Figures 5.1(a) and 5.1(b) illustrate "global" and "local" approaches. One can easily see how much individual fragments of the test image differ one from another and how relatively simply organized are most of the blocks of which it consists.

(ii) Adaptive filter design assumes empirical evaluation of signal spectra. In global image spectra, spectra variations due to image nonstationarity are hidden and are difficult if not impossible to detect. This can be clearly seen in Figure 5.2 that illustrates variability DCT spectra of blocks of the test image of Figure 5.1 compared to its global spectrum. Therefore in global spectrum analysis, local image information will be neglected in favor of global one, which usually contradicts processing goals.

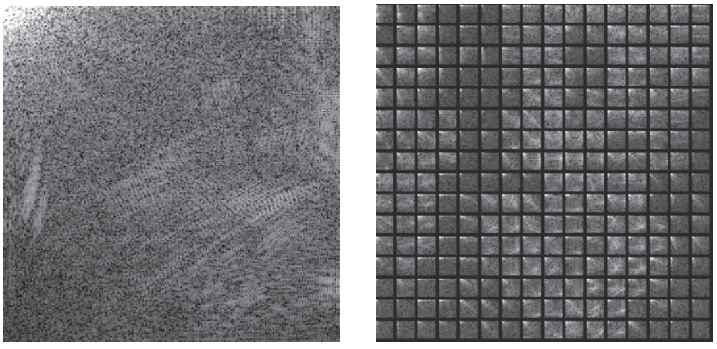
(iii) It is well known that when viewing image, human eye's optical axis permanently hops chaotically over the field of view and that the human visual acuity is very nonuniform over the field of view. The field of view of a man is about 30°. Resolving power of man's vision is about $1'$. However such a relatively high resolving power is concentrated only within a small fraction of the field of view that has size of about 2° (see [14]). Therefore, area of the acute vision is about 1/15th of the field of view. For images of 512 pixels, this means window of roughly 30 pixels.

(iv) Visual objects to be recognizable have to contain sufficiently large number of resolution cells (pixels). As an immediate illustration of this fact, one can recall that, for the representation of printed characters, one needs a matrix of at least 8×8 pixels. Even the smallest one-pixel-size object needs a neighborhood of 3×3 pixels to be detectable if not recognizable. The same and even to a greater degree holds for "texture" images. Texture identification is also possible only if texture area contains sufficiently large numbers of pixels. This means that image can be regarded as a composition of object domains with the linear size from several to several tens of resolution cells.

The most straightforward way to implement local filtering is to do it in a hopping window. This is exactly the way of processing implemented in most popular audio and image coding methods such as JPEG and MPEG. "Hopping window" processing being very attractive from the computational complexity point of view suffers however from "blocking effects"—artificial discontinuities at the edges of the hopping window. This motivated appearance of "lapped" transforms (see [15]) that avoid, though not completely, blocking effects by window hopping with overlap of half the window size. Obviously, the ultimate solution of the

(a) (b)

FIGURE 5.1. (a) A test 256×256 pixel image and (b) the same image divided into 32×32 pixel blocks shown by the grid.



(a) (b)

FIGURE 5.2. (a) Global and (b) blockwise DCT spectra of the test image of Figure 5.1.

"blocking effects" problem would be sample-by-sample processing in sliding window.

According to the above outlined principle of scalar filtering, in sliding window processing, one should, for each position of the filter window, compute transform of the signal vector within window, form on this base a filter, modify accordingly transform coefficients, and then compute inverse transform. With sliding window, inverse transform need not, in principle, be computed for all signal vectors within window since only central sample of the window has to be determined in order to form, pixel by pixel, the output signal. Figure 5.3 illustrates the described process.

Sliding window adaptive filtering can be theoretically justified if one admits that processing quality is evaluated by "local criteria" (see [12, 13]). According to

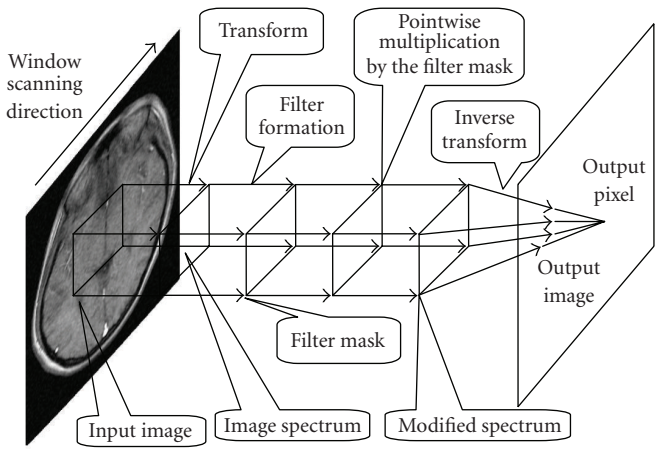


FIGURE 5.3. Flow diagram of sliding window processing in transform domain.

the local criteria, signal/image processing quality is evaluated for each individual image pixel by averaging of the "loss function," which measures deviations of the pixel estimates from desired ones, over a set of pixels that forms a neighborhood of the given pixel.

Within the MSE approach defined by (5.1), the local criterion determines MSE for the central sample (pixel) a_k of the sliding window in each of its kth position over the given set of signal samples (image pixels) as

$$\mathrm{MSE}(k) = \mathrm{AV}\left\{\sum_{m=1}^{M}\sum_{n\in \mathrm{Window}(k)} \left| a_n^{(m)} - \hat{a}_n^{(m)} \right|^2\right\}, \tag{5.16}$$

where Window(k) is a set of pixels in the window. Therefore, in the local processing, MSE-optimal filters generate estimates of the window central pixel taking the corresponding sample from estimates of the window samples that minimize mean-squared restoration error within the scope of the window. Being optimized individually for every particular window position, such local filters are space/time variant and locally adaptive.

5.3.2. Selection of transforms

Selection of orthogonal transforms for the implementation of the filters is governed by

(i) the admissible accuracy of approximation of the general linear filtering with scalar filtering;
(ii) the convenience of formulating a priori knowledge regarding image spectra in the chosen base;

(iii) the easiness of spectrum estimation from the observed data that is required for the filter design;

(iv) the computational complexity of the filter implementation.

By the present time, quite a number of orthogonal transforms have been suggested and have found their applications, mainly for signal and image compressions (see [16, 17]). Among these transforms, discrete cosine transform proved to be one of the most appropriate transforms for sliding window transform domain filtering. Arguments in favor of DCT are the following.

(i) DCT exhibits a very good energy compaction capability, which is a key feature for empirical spectra estimation and the efficiency of filters of (5.13)–(5.15).

(ii) Being advantageous to DFT in terms of energy compaction capability, DCT can also be regarded as a good substitute for DFT in image restoration tasks where imaging systems are specified in terms of their frequency responses. (for signal convolution in DCT domain, see Chapter 3, Section A.1.2).

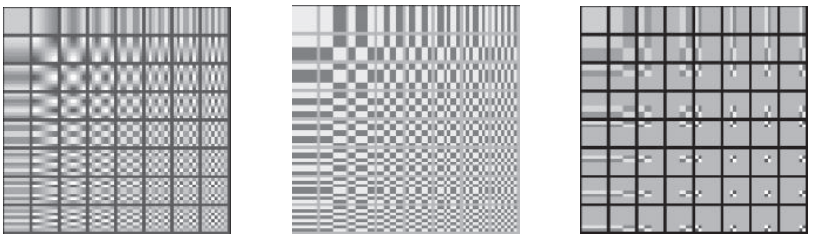
(iii) DCT is suitable for multicomponent signal/image processing and video processing, in which cases 3D transform over spatial 2D coordinates and componentwise/temporal coordinate should be considered.

(iv) Computing DCT in a sliding window (for 2D and 3D signals, a rectangular window) is possible with the use of recursive algorithms described in Chapter 3, Section A.1.2. Generally, computational complexity of DCT spectral analysis in sliding window is proportional to the window size. In sliding window filtering, it is most natural to select window dimensions to be odd number to secure window symmetry for its central pixel. As it is shown in the appendix, when window size is an odd number, for computing filtering output for the window central pixel by inverse DCT of the modified signal spectral coefficients, only coefficients with even indices are involved and therefore should be computed. This results in further almost 2, 4, or 8 times reduction of the computational complexity for 1D, 2D, and 3D signals, correspondingly. Note also that DCT spectral components with zero indices (local dc-components) can be recursively computed even faster with the complexity that does not depend on the window size (see [12, 13]).

Although DCT looks advantageous in many respects, other transforms can also be used for local adaptive filtering in transform domain. The closest candidates are transforms with binary basis functions such as Walsh-Hadamard and Haar transforms that are computationwisely the simplest ones and, in principle, allow recursive computation in sliding window (see [18]). For moderate window sizes, basis functions of DCT and Walsh-Hadamard transforms are quite similar as one can see from Figure 5.4. Yet another option is using signal/image polynomial expansion in sliding window considered in Chapter 6.

5.3.3. Selection of the window size and shape

Important parameters of the transform domain sliding window filtering are size and shape of the window. Obviously, the larger the window size, the higher is the noise suppression capability of the filtering in image "flat" areas is, where no significant signal changes occur. From the other side, filtering with a large window



(a) (b) (c)

FIGURE 5.4. Basis functions of 2D 8×8 DCT (a), Walsh (b), and Haar (c) transforms.

size in the "nonflat" areas may cause either image blur or insufficient noise suppression due to poor separation of signal and noise spectra in the spectrum of the noisy signal.

In what follows, we assume that images are represented by their samples over a rectangular sampling grid. In this geometry, recursive computing of the window transform coefficients is possible for rectangular windows aligned with the direction, in which filer window scans images: rowwise/columnwise, or diagonalwise. In the latter case, filter window is a rectangle rotated 45° with respect to image rows/columns. In what follows, we assume rowwise/columnwise image pixel-by-pixel filtering with a sliding rectangular window.

The least computationally expensive solution is filtering with a fixed window size. In this case, selection of the window size should be made as a compromise between noise suppression and image-preserving capabilities of the filtering. One can say that the size of the window is, roughly speaking, commensurable with the minimal size of signal/image details that have to be preserved and differentiated from noise in course of the filtering.

Obviously, image restoration capability of sliding window transform-domain filtering will be higher, if window size is selected adaptively in each window position. To this goal, filtering should be carried out in windows of multiple sizes, and in each window position, the best filtering result should be taken as the signal estimate in this position using methods of statistical tests such as, for instance, intersection of confidence intervals method described in Chapter 6. Another option in the multiple-window processing is combining, in a certain way, filtering results obtained for different windows. All this, however, increases correspondingly the computational complexity of the filtering, which may become prohibitive, especially in video processing.

5.3.4. DCT filtering in sliding window 3×3: thresholding the directional Laplacians

As it was stated above, in sliding window transform-domain filtering, filter window size should be commensurable with the size of signal (image) details. The smallest detail size is one sample (pixel), which corresponds to the smallest 1D

window size of three samples for 1D signal and 3×3 pixels for 2D images. Let us analyze the shape of the basis functions in local 2D DCT spectral analysis in the window 3×3. From nine basis functions

$$\begin{bmatrix} \mathbf{DCT00} & \mathbf{DCT10} & \mathbf{DCT20} \\ \mathbf{DCT01} & \mathbf{DCT11} & \mathbf{DCT21} \\ \mathbf{DCT02} & \mathbf{DCT12} & \mathbf{DCT22} \end{bmatrix}$$

$$= \begin{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -\sqrt{3} & 0 & \sqrt{3} \\ -\sqrt{3} & 1 & \sqrt{3} \\ -\sqrt{3} & 1 & \sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ 1 & -2 & 1 \\ 1 & -2 & 1 \end{bmatrix} \\ \begin{bmatrix} \sqrt{3} & \sqrt{3} & \sqrt{3} \\ 0 & 0 & 0 \\ -\sqrt{3} & -\sqrt{3} & -\sqrt{3} \end{bmatrix} \begin{bmatrix} 3 & 0 & -3 \\ 0 & 0 & 0 \\ -3 & 0 & 3 \end{bmatrix} \begin{bmatrix} \sqrt{3} & -2\sqrt{3} & \sqrt{3} \\ 0 & 0 & 0 \\ -\sqrt{3} & 2\sqrt{3} & -\sqrt{3} \end{bmatrix} \\ \begin{bmatrix} 1 & 1 & 1 \\ -2 & -2 & -2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 & -\sqrt{3} \\ -2\sqrt{3} & 0 & 2\sqrt{3} \\ \sqrt{3} & 0 & -2\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \end{bmatrix}, \tag{5.17}$$

only functions **DCT00**, **DCT20**, **DCT02**, and **DCT22** generate spectral coefficients with even indices that are involved in the computation of the window central pixel. As one can easily verify, these functions are

$$\begin{aligned} \mathbf{DCT00} &= \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \\ \mathbf{DCT20} &= \begin{bmatrix} 1 & -2 & 1 \\ 1 & -2 & 1 \\ 1 & -2 & 1 \end{bmatrix}, \\ \mathbf{DCT02} &= \begin{bmatrix} 1 & 1 & 1 \\ -2 & -2 & -2 \\ 1 & 1 & 1 \end{bmatrix}, \\ \mathbf{DCT22} &= \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}. \end{aligned} \tag{5.18}$$

Being applied in a sliding window, these functions generate image local mean (function **DCT00**) and image horizontal (function **DCT02**), vertical (function **DCT20**), and "isotropic" (function **DCT22**) Laplacians. As one can easily see, the latter can be decomposed into a sum of diagonal Laplacians taken at angles $\pm 45°$:

$$\mathbf{DCT22} = -(\mathbf{DCT220} + \mathbf{DCT221}), \tag{5.19}$$

where

$$\mathbf{DCT220} = \begin{bmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix}, \qquad \mathbf{DCT221} = \begin{bmatrix} 1 & 1 & -2 \\ 1 & -2 & 1 \\ -2 & 1 & 1 \end{bmatrix}. \tag{5.20}$$

This casts a new light at the filter denoising capability and leads to different possible modifications of the filter. One of the possible modifications is an implementation of the rejective filter of (5.14) through a direct image five-channel convolution with window functions **DCT00**, **DCT20**, **DCT02**, **DCT220**, and **DCT221**, thresholding the convolution results and recombining them into the output image. Second possible modification is a "*softened thresholding*" of the convolution results. By "softened thresholding" we mean the following operation:

$$\mathrm{OUTPUT} = \begin{cases} \mathrm{INPUT} & \text{if } |\mathrm{INPUT} > \mathrm{THR}|, \\ \mathrm{THR} \cdot \mathrm{sign}(\mathrm{INPUT}) \left| \dfrac{\mathrm{INPUT}}{\mathrm{THR}} \right|^{P} & \text{otherwise,} \end{cases} \tag{5.21}$$

where THR is the same threshold parameter as in that in (5.14) and P is a parameter that defines the degree of approximation of the "hard thresholding" of (5.14) (the "hard thresholding" is achieved when $P \to \infty$). The "softened thresholding" may help to reduce the loss of low-contrast image details in case of the hard thresholding.

5.3.5. Overlap-and-add filtering

In the described sliding window transform-domain filtering, only the window central pixel value was supposed to be estimated in each position of the window as it is shown in flow diagram of Figure 5.3. A modification of this method was suggested (see [19]) in which every pixel in a window is estimated in each window position and final estimate of the filter output is obtained, for every pixel, by averaging the corresponding outputs of all windows that contain the given sample (pixel).

Because of the need to perform inverse transform of the entire window in order to obtain estimates of all samples (pixels) within window, "overlap-and-add" filtering cannot benefit from the recursive computation of window spectra. Thus, per-pixel computational complexity of such a filtering, provided that it uses fast transforms, increases by additional $O(\mathit{WindowSize} \log \mathit{WindowSize})$ operations with respect to that of the filtering with the evaluation of only the central pixel. However, experimental experience shows that overlap-and-add filtering produces less artifacts and demonstrates a bit better denoising capability. Some comparative quantitative figures are presented below.

5.3.6. Performance and potentials of local adaptive filtering: experimental verification

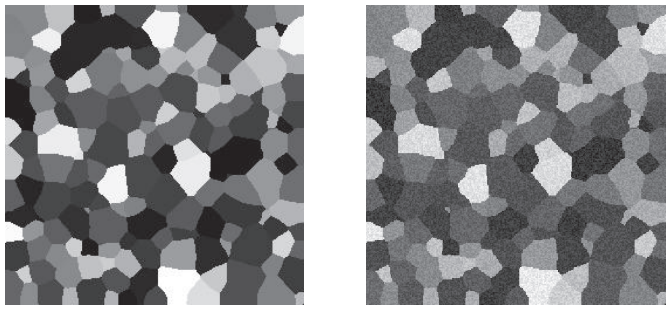
Image restoration and enhancement capability of the described sliding window DCT domain filtering (SWDCT) were expensively tested both on simulated test and real-life images and video. Some illustrative examples of image and video denoising, deblurring, and enhancement are given in Figures 5.5–5.12.

Figure 5.5(c) shows a result of denoising, by means of rejecting filtering in DCT domain in sliding window of 7×7 pixels of a 256×256 pixel test piecewise constant image (Figure 5.5(b)), corrupted by additive white noise with uniform distribution in the range $\pm$ 25 gray levels (image dynamic range is 256 gray levels). Original noiseless image is shown, as a reference, in Figure 5.5(a). Image in Figure 5.5(d) shows a rejective filter local "transparentness map" computed, for each processed pixel, as a fraction, from zero to one, of transform coefficients in the window that pass the thresholding. White in this figure corresponds to filter full "transparentness" (all transform coefficients pass the threshold), black corresponds to filter full "opaqueness" (none of coefficients, except dc-component, pass). One can clearly see on this image that within image flat areas, where no substantial changes of gray levels occur, the rejecting filter tends to replace pixels by local mean (local dc-component) over the window, while in vicinities of edges it passes practically all image local transform coefficients without modification, and therefore, preserves the input image content and does not blur edges, though does not filter out noise in these areas either. This "edge preserving" property of the described local adaptive filters can be even more explicitly seen on graphs of noisy and denoised image rows as the ones shown in Figure 5.5(e), for a row highlighted in Figure 5.5(c). Curiously enough, human visual system is also substantially less sensitive to noise in the vicinity of image edges than in "flat" areas.

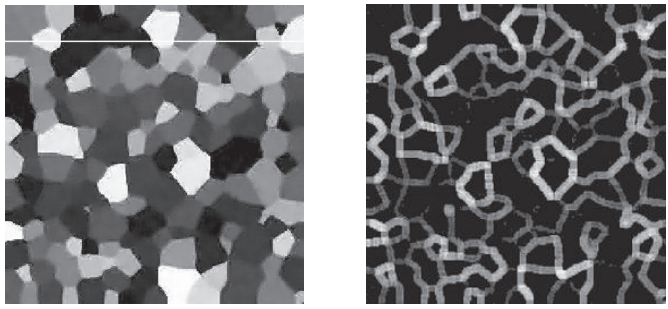
Figure 5.6 represents an example of a blind restoration of a high-resolution satellite image by sliding window DCT 3×3 filtering described in Section 5.3.4. For image deblurring, it was first subjected to "aperture correction" by applying a filer inverse to the image sensor's aperture function. As inverse filtering tends to amplify sensor's noise, deblurred image was denoised by means of direct 5-channel convolution with window functions **DCT00**, **DCT20**, **DCT02**, **DCT220**, and **DCT221**, soft thresholding the convolution results according to (5.21) and recombining them into the output image.

Figures 5.7(a)–5.7(c) illustrate denoising and Figures 5.7(d)–5.7(f) illustrate "blind" deblurring of color images using spatial sliding window processing in 3D transform domain (two spatial dimensions and color component dimension). For "blind" deblurring, "fractional spectrum filter" of (5.15) was used with window size $7 \times 7 \times 3$, $\{\lambda_{r,\mu} = 1\}$, $P = 0.75$, and filtering threshold found experimentally on the base of visual evaluation of the restored image quality.

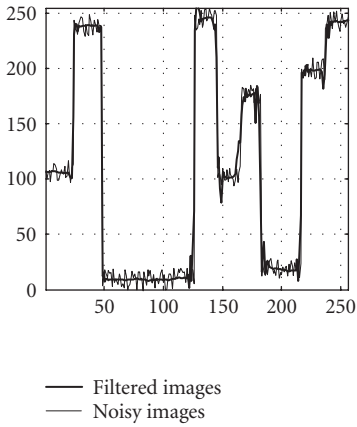
Figures 5.8 and 5.9 illustrate denoising and "blind" deblurring of test (Figure 5.8) and real life (Figure 5.9) video sequences using $5 \times 5 \times 5$ sliding window DCT domain processing in two spatial dimensions and in time dimension of video



(a) (b)



(c) (d)



(e)

FIGURE 5.5. Local adaptive filtering for denoising a piecewise constant test image. (a) Original 256 × 256 pixel test image; (b) its noisy copy; (c) filtered image (rejective filtering); (d) a map of local filter "transparence" (white for "transparent," dark for "opaque"); (e) plots of 32th row (indicated by white line in image (c)) of noisy (normal line) and filtered (bold line) images.

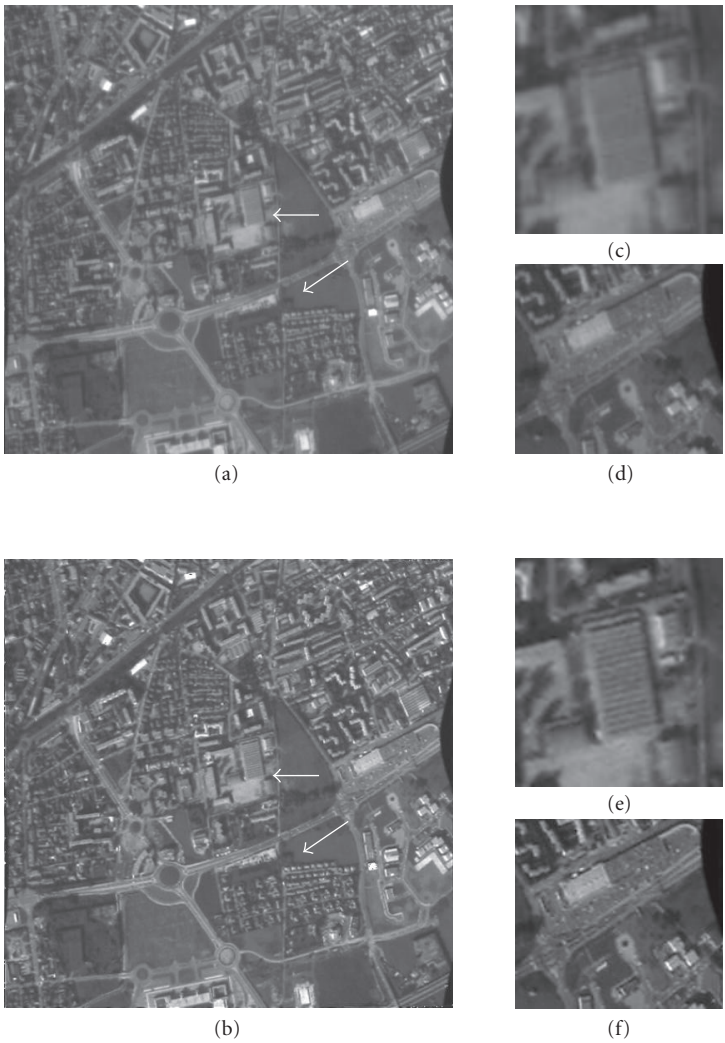
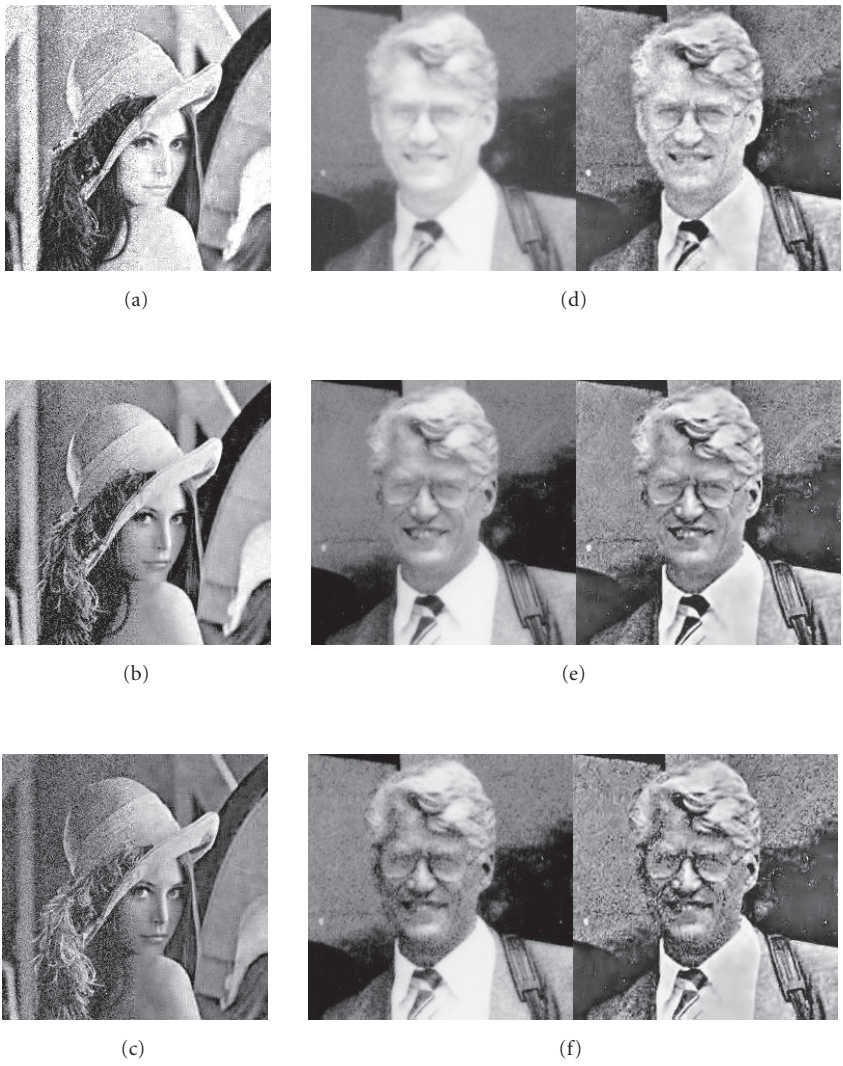


FIGURE 5.6. Denoising and deblurring a high-resolution satellite image by means of sliding window DCT domain filtering in window of 3×3 pixels (implemented as the five-channel filtering direct convolution): (a) initial image; (b) enhanced image; (c), (d) and (e), (f) corresponding magnified fragments of the above two images marked by white arrows.

frames. For enhancing resolution in real-life video illustrated in Figure 5.9, an "inverse" rejective filter was used with $\{\lambda_{r,\mu}\}$ specified by parameters of the thermal video camera used.

Figure 5.10 illustrates application of sliding spatial window DCT transform-domain filtering for denoising speckles in optical interferograms. The speckled



(a) (d)

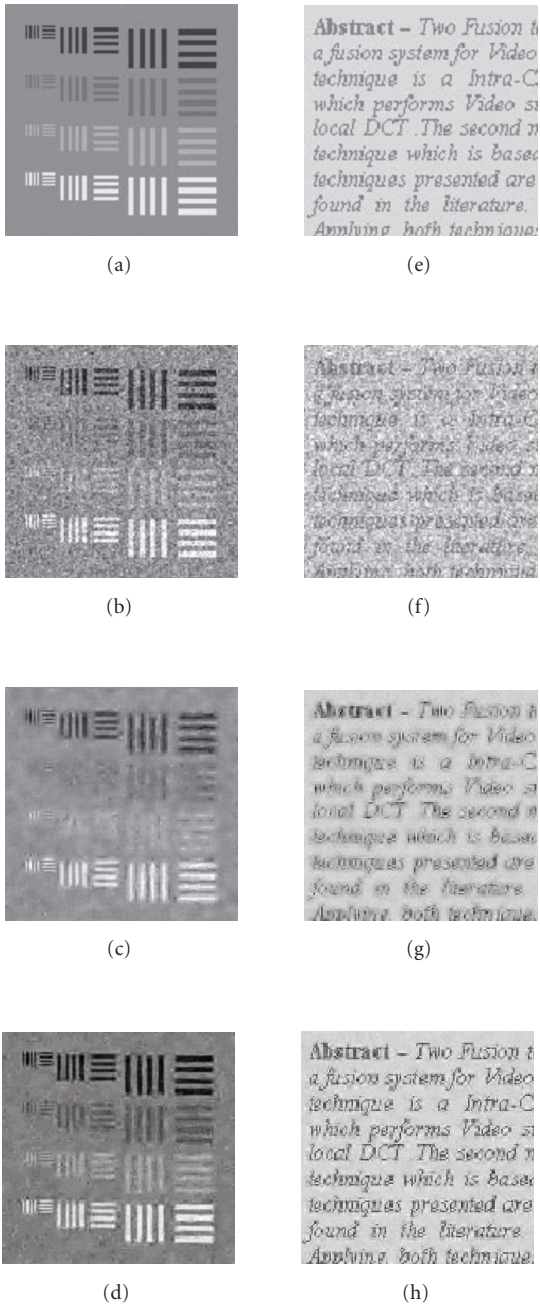
(b) (e)

(c) (f)

FIGURE 5.7. Denoising and blind deblurring of color images. Left column: left halves of the images represent noisy images and right halves are results of denoising. Right column: original blurred image (left) and deblurred image (right). Corresponding individual color image components are shown: R— in boxes (a) and (d); G— in boxes (b) and (e); B— in boxes (c) and (f).

interferogram was processed with filtering threshold proportional to the value of the window dc-component.

Figure 5.11 illustrates image enhancement using sliding window DCT domain "fractional spectrum" filter of (5.15). Window size in this filtering was 11×11 pixels, filter parameters P, g were, correspondingly, 0.75 and 2.5. Noise suppression



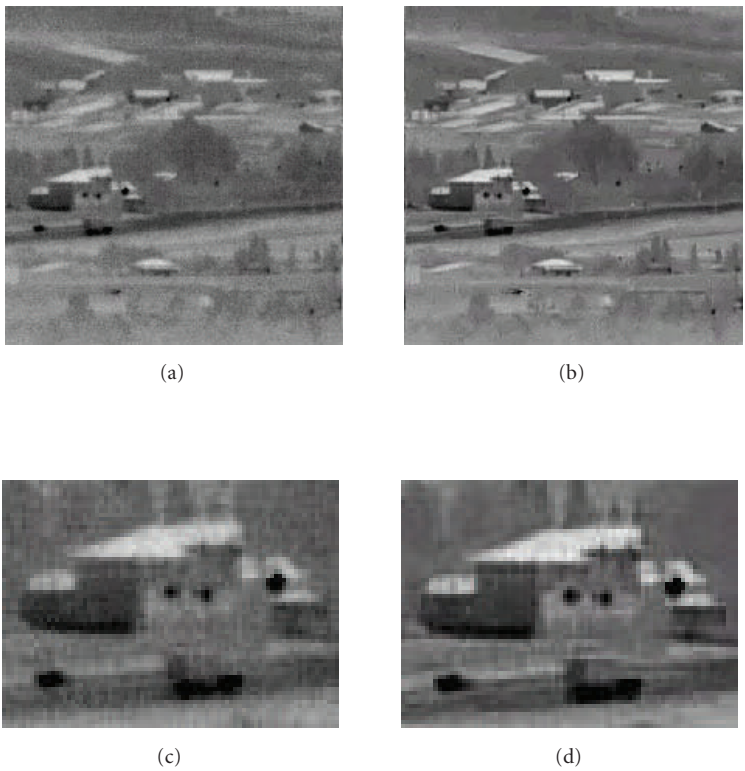
(a) (e)

(b) (f)

(c) (g)

(d) (h)

FIGURE 5.8. Local adaptive 2D and 3D sliding window DCT domain filtering for denoising video sequences. Top row—examples of frames of initial test video sequence. Middle row—result of 2D sliding 5×5 pixel spatial window filtering. Bottom row—examples of frames of restored video obtained using 3D sliding $5 \times 5 \times 5$ spatial/temporal window filtering. Full video can be found at http://www.eng.tau.ac.il/~yaro/Shtainman/shtainman.htm.



(a) (b)

(c) (d)

FIGURE 5.9. Examples of frames of initial (a) and restored (b) thermal real-life video sequence and corresponding magnified fragments of the images (c) and (d). Note enhanced sharpness of the restored image. Entire movie can be found at http://www.eng.tau.ac.il/~yaro/Shtainman/shtainman.htm.

threshold THR was 3 quantization intervals for all transform coefficients (for image dynamic range 0–255), which roughly corresponds to an estimate of standard deviation of the image noise component that was set to be cleaned out. Dimensions of images in Figure 5.11 are 512×512 pixels (a), (b); 256×256 (c), (d); and 330×440 (e), (f).

Figure 5.12 illustrates 1D filtering in Haar transform domain for denoising of a real-life ECG signal in sliding window of 16 pixels. Second and third (from top) images in the figure illustrate time-"frequency" (Haar transform spectral index) representation of the ECG signal before and after thresholding, correspondingly.

Quantitative data of experimental verification of additive noise suppression capability of three modifications of sliding window local adaptive filters are provided in Table 5.1 for rejective filter of (5.14), empirical Wiener filter of (5.13), and overlap-and-add filter described in Section 5.4.1.1 (see [19, 20]). As a filter denoising capability benchmark, an "ideal" Wiener filter was used. By the "ideal" Wiener filter we mean the filter of (5.11) built for the exactly known signal and

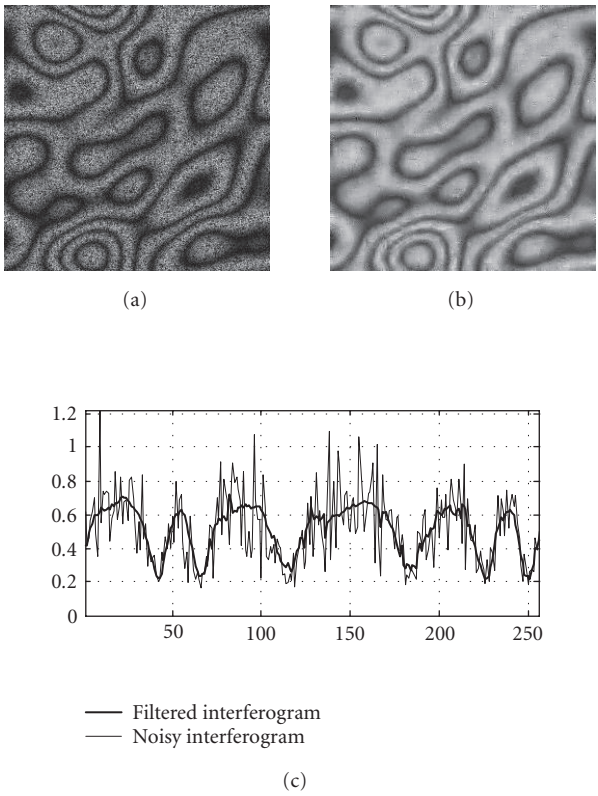


FIGURE 5.10. Application of local adaptive sliding window filtering in DCT domain for denoising speckled interferogram: (a) noisy interferogram, (b) SWDCT-filtered interferogram, (c) plots of a sample row of images (a) and (b).

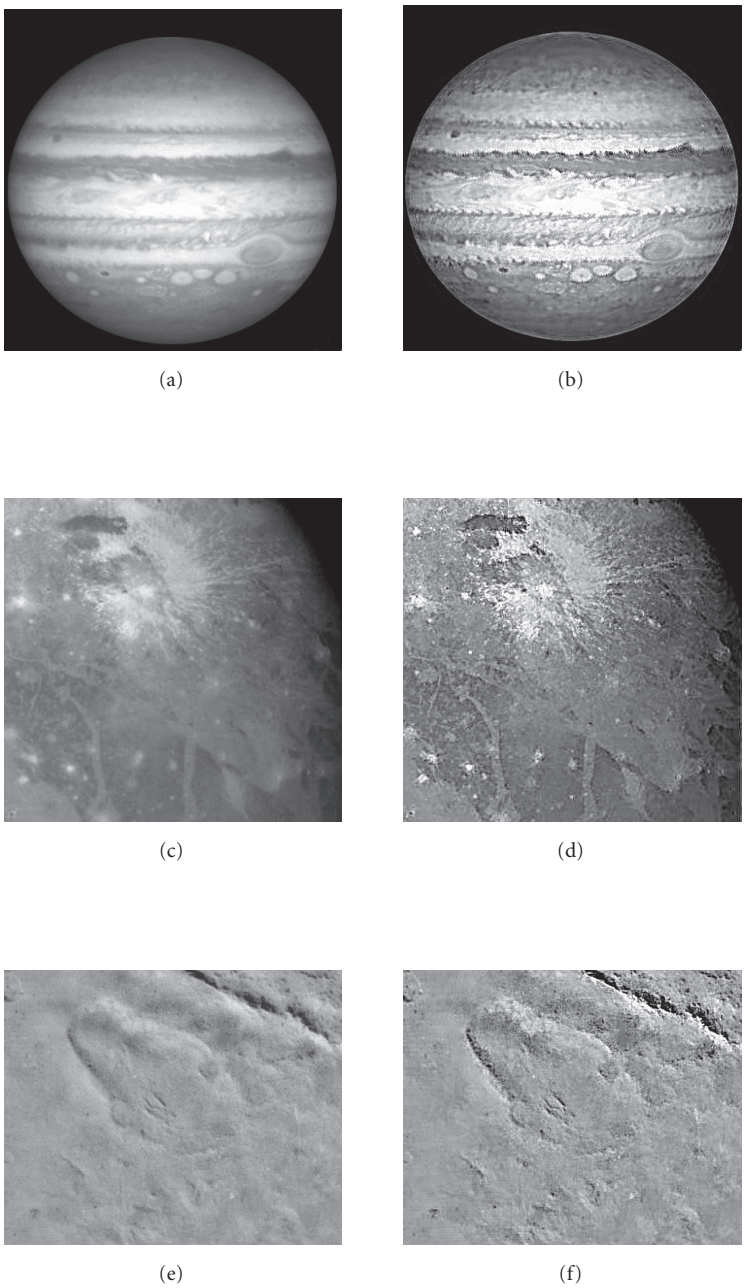
noise transform domain power spectra. Such a filter is MSE optimal for each specific individual realization of signal and noise. While this filter cannot be realized in practice, one can implement it in computer simulation when true signal spectrum and noise realizations are exactly known. The comparison results were obtained for three test images, optimized, for each image, window size, and threshold values of empirical Wiener and rejective filters.

One can summarize these experimental results as follows.

(i) The tested filters do provide a substantial noise reduction and exhibit quite good "edge" preserving capability.

(ii) Overlap-and-add filtering improves noise suppression capability of the filtering though not very substantially.

(iii) Filter noise suppression capability is not too far from that of the "ideal" Wiener filter. However, there is a room for further improvement that can be reached with better local spectra estimation.



(a) (b)

(c) (d)

(e) (f)

FIGURE 5.11. Image enhancement using sliding window DCT domain "fractional spectrum" filter: (a), (c), (e) initial images, (b), (d), (f) enhanced images. (Images (a) and (c) are adopted from http://www.solarsystem.nasa.gov/ planets/; image (e) is adopted from [32].)

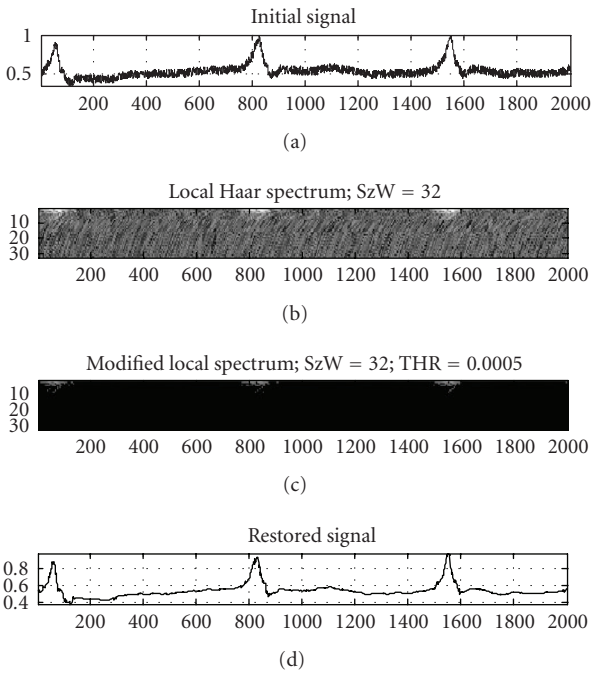


FIGURE 5.12. Denoising of a 1D electrocardiogram signal by local adaptive rejective filtering in the Haar Transform domain (window size is 16 samples). From top to bottom: input noisy signal, its time-Haar transform index representation, the same after thresholding and filtered signal.

5.4. Wavelet transforms and wavelet denoising methods

Wavelet transforms represent yet another family of transforms for transform-domain signal restoration. The notion of wavelet transforms refers to signal transform with transform kernels (basis functions) that are formed from a certain "mother" function by its both shift and scaling. For a continuous signal $a(x)$, its wavelet transform $\alpha(s, \xi)$ is defined as

$$\alpha(s, \xi) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} a(x) \varphi\left(\frac{x - \xi}{s}\right) dx, \tag{5.22}$$

where $\varphi(\cdot)$ is the transform "mother" function, and s is a scale parameter. If the mother function has finite support, wavelet transform allows multiresolution signal representation and analysis. It is this property that defines the attractiveness of wavelet transforms.

Equation (5.22) is, for every scale s, a convolution integral. Therefore, wavelet transforms can be treated as signal "subband" decompositions by signal filtering using filters with frequency responses equal to Fourier transform of the mother

Table 5.1. Noise suppression capability data for four sliding window DCT domain filtering methods: standard deviation of restoration error for different test images, different input noise levels, and different filtering methods.

Test image	Filter type	Noise level		
		15	30	60
Air photo1	Rejecting filter	6.1	8.1	10.8
	Empirical Wiener filter	6	8.1	10.8
	Overlap/add rejecting filter	5	7.1	10.4
	"Ideal" Wiener filter	3.6	5.6	9.2
Air photo2	Rejecting filter	5.6	6.4	10
	Empirical Wiener filter	5.6	6.4	10.4
	Overlap/add rejecting filter	5	6	8.2
	"Ideal" Wiener filter	3.3	5	8.2
Lena image	Rejecting filter	10.4	15	21
	Empirical Wiener filter	9.3	14.1	19
	Overlap/add rejecting filter	8	11.9	17.5
	"Ideal" Wiener filter	5.8	8.4	13

function on the corresponding scale:

$$H_s(f) = \int_{-\infty}^{\infty} \varphi\left(\frac{x}{s}\right) \exp(i2\pi f x)dx. \tag{5.23}$$

This interpretation suggests a very constructive and widely used way to implement wavelet transforms as iterative signal decomposition to lowpass and highpass components as it is illustrated in Figure 5.13. In such an implementation, direct and inverse wavelet transforms are defined by lowpass filters, interpolation filters, subsampling procedures, and by summation/subtraction units. Subsampling provides wavelet scaling while discrete lowpass filtering implements shifts. Subsampling is usually performed as a two-time decimation, which results in scales that are powers of two. By means of modifying lowpass filtering and interpolation procedures, one can implement different types of wavelet expansion for processing optimization.

Primarily, wavelet transforms were developed mostly as a tool for data compression (see [21, 22]). Donoho and Johnstone, see [9–11], pioneered their application to signal denoising. Wavelet denoising has obtained a nickname "wavelet shrinkage" with "hard" and "soft" thresholdings and operates basically in the same three steps introduced in Section 5.2.1, where T is now a wavelet transform. A direct analogy exists between wavelet shrinkage and sliding window transform-domain filtering as it is shown in Table 5.2 in denotations of Section 5.2.2. Wavelet shrinkage methods apply the "soft" and "hard" thresholdings to subbands-decomposed image components as it is shown in the block diagram in Figure 5.14.

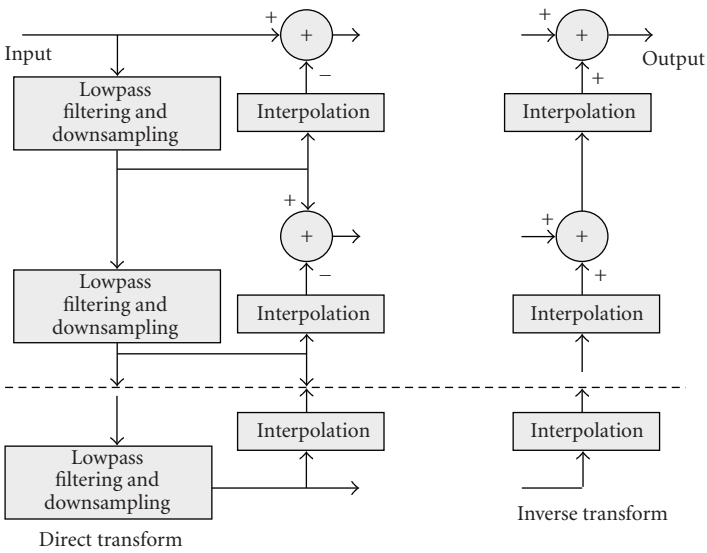


Figure 5.13. Direct and inverse wavelet transforms implemented as signal subband decompositions.

Table 5.2. Wavelet shrinkage and empirical Wiener filtering formulas for filter coefficients.

Wavelet shrinkage: "soft thresholding"	$\eta_r = \dfrac{\max(\lvert\beta_r\rvert - \mathrm{THR}, 0)}{\lvert\beta_r\rvert}$
Empirical Wiener scalar filtering	$\eta_r = \dfrac{\max(\lvert\beta_r\rvert^2 - \overline{\lvert\nu\rvert^2}, 0)}{\lvert\beta_r\rvert^2}$
Wavelet shrinkage: "hard thresholding"	$\eta_r = \begin{cases} 1 & \text{if } \lvert\beta_r\rvert > \mathrm{THR} \\ 0 & \text{otherwise} \end{cases}$
"Rejecting" filtering	$\eta_r = \begin{cases} 1 & \text{if } \lvert\beta_r\rvert > \mathrm{THR} \\ 0 & \text{otherwise} \end{cases}$

It was found in experiments that wavelet shrinkage may suffer from artifacts in the form of undershoots and overshoots to the denoised signal that are associated with boundary effects. In order to compensate this drawback, translation-invariant wavelet denoising was suggested (see [23]). In the translation-invariant denoising, multiple copies of the signal obtained by its cyclic shift are denoised by the above procedure and the output signal is found as an average over filtering

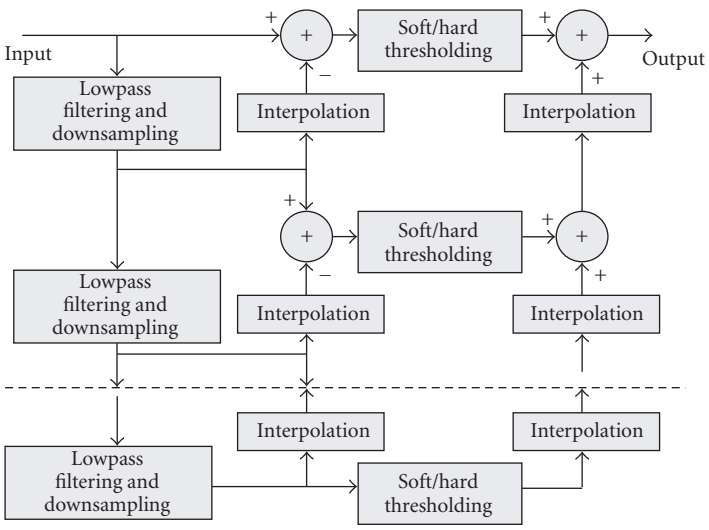


Figure 5.14. Flow diagram of wavelet denoising methods.

results for all the copies. Note also that wavelet transforms can be used in sliding window transform-domain processing as well.

5.4.1. Sliding window transform-domain filtering versus wavelet filtering: hybrid algorithms

5.4.1.1. Comparison of signal/image denoising capabilities of sliding window and wavelet filtering methods

The results of extensive experimental comparison of denoising capability of sliding window transform domain and wavelet filtering were reported in [19, 20, 24]. Compared were the following wavelet and sliding window filter modifications: Haar wavelet (WL-Haar) and Duabechis-4 wavelet (WL-Db4), their translation-invariant (TI) versions, sliding window DCT (SWDCT) and Haar (SWHaar) filters and sliding window Daubechis-4 (SWDb-4) filter, and their overlap-and-add versions ASWDCT, ASWHaar, ASWDb-4, correspondingly.

Two 1D test (ECG and piecewise constant) signals and 5 test images, Lena, piecewise (PW) constant image, two air photographs, and NMR image, were used in experiments. White Gaussian noise with standard deviation 0.1 was added to one-dimensional signals resulting in maximal absolute error 0.08. Noisy images were obtained for denoising experiments by adding white Gaussian noise with standard deviation 25 gray levels (of 255), which corresponds to PSNR = 10. For 1D signals, residual root mean square (RMSE) and maximal absolute (MAE) restoration errors were evaluated. Quality of image denoising was evaluated in

Table 5.3

Transforms	Noisy signals, MAE = 0.08				Noisy images, PSNR = 10				
	ECG signal		Piecewise constant signal		Lena	PW-constant	Air photo1	Air photo2	NMR
	RMSE	MAE	RMSE	MAE	PSNR	PSNR	PSNR	PSNR	PSNR
WL-Haar	0.06	0.042	0.042	0.024	26.7	**30**	26.9	32.3	30.9
WL-Db4	0.052	0.038	0.052	0.035	27.2	27.8	27.6	33.3	31.5
SWHaar	0.052	0.033	**0.037**	**0.022**	27.4	29.5	28.1	34.2	31.8
SWDCT	**0.04**	**0.028**	0.055	0.038	**27.8**	28	**29**	**34.6**	**32.7**

Table 5.4

Transforms	Noisy signals, MAE = 0.08				Noisy images, PSNR = 10				
	ECG signal		Piecewise constant signal		Lena	PW-constant	Air photo1	Air photo2	NMR
	RMSE	MAE	RMSE	MAE	PSNR	PSNR	PSNR	PSNR	PSNR
TI-WL-Haar	0.038	0.027	**0.022**	0.014	29.8	32.5	30	34.9	33.7
TI-WL-Db-4	0.035	0.026	0.035	0.023	30.1	30.8	30.4	35.2	34.2
ASWHaar	0.043	0.031	**0.022**	0.014	29.6	**33.5**	29.8	35.1	33.5
ASWDCT	**0.034**	**0.024**	0.044	0.03	**30.5**	31	**30.8**	**35.7**	**34.6**
ASWDb-4	0.035	0.025	0.039	0.028	29.9	31.8	30.2	35.3	33.8

terms of the ratio of image dynamic range to standard deviation of the residual noise (PSNR).

In each experiment, optimal parameters of filters such as threshold values, and for sliding window filters, window size, were optimized to obtain the lowest signal restoration error. The results are summarized in Table 5.3 for proper wavelet and sliding window filters and in Table 5.4 for their corresponding translation-invariant and overlap-and-add modifications. The best results are shown in boldface font. As one can see in both tables, SWDCT filtering outperforms other methods in most of the experiments with real-life signals and images. For test piecewise constant signals, SWHaar and WL-Haar showed better results.

5.4.1.2. Hybrid wavelet/SWTD filtering

While the SWDCT filtering was demonstrated to have certain superiority to the wavelet shrinkage in terms of noise suppression capability and local adaptivity provided appropriate selection of the window size for each particular image, its drawback is the need to manually select window size.

As it was already mentioned, one way to overcome this drawback of the moving window filtering is parallel filtering in multiple moving windows of different sizes selected, for obtaining the best signal estimation at every particular position,

of the window. A less computationally expensive alternative is combing multiresolution property of wavelets with better local adaptivity of SWDCT filtering in a hybrid wavelet-SWDCT processing method (see [25]). According to this method (Figure 5.15), image is first represented in multiple scales by its subband decomposition as in the wavelet shrinkage. Each subband component obtained is then subjected, instead of the wavelet shrinkage, to DCT moving window filtering with size of the window being kept the same for all scales. As one can see, in the hybrid method simple "soft" or "hard" thresholding of the signal in all scales is substituted by their DCT filtering in moving window. This substitution imitates parallel filtering of the initial image with a set of windows according to the scales selected.

Consider the following example. For high-resolution images, DCT filtering in moving window requires window size 3×3, otherwise the filtering may result in the loss of tiny image details. This requirement limits noise suppression capability of the filtering that is proportional to the window size. In hybrid filtering, when window 3×3 in the first scale is used, in the scale 2, effective window size is 6×6, in the scale 3 it is 12×12 and so on. This potentially may add to the filtering noise suppression capability. As it was already mentioned, second and third functions of DCT in 3×3 window represent vertical and horizontal Laplacian operators and the last function can be decomposed into a sum of diagonal Laplacians. Therefore, DCT filtering in 3×3-window can be replaced by filtering in the domain of four directional Laplacians. Experimental experience proves that for high-resolution images, such an implementation is advantageous to simple DCT since it produces less filtering artifacts. Its use in hybrid filtering promises additional advantages because it is equivalent to the corresponding increase of the number of effective basis functions. Examples of the effective set, for the input image plane, of these directional Laplacians in four scales are shown in Figure 5.16.

Described implementation was used in the experiments reported in (see [25]). For testing the method, modeled images and a number of real-life images were used. For all images, Gaussian white noise was added to produce noisy images for the filter input. Three types of filters were tested: moving window DCT filter, wavelet shrinkage filter, and the suggested hybrid filter.

In the wavelet shrinkage filter, a code of image pyramid from University of Pennsylvania package [26] was used. For all filters, optimal parameters that minimize RMS difference between initial noise-free image and the filtered one were experimentally found. For DCT moving window filtering, window size and filter threshold were optimized. For wavelet shrinkage and hybrid filtering, filter parameters were optimized and the best of the following types of wavelets was chosen: Haar wavelet ("haar"), binomial coefficient filters ("binom3," "binom9," and "binom13"), Daubechies wavelets ("daub2" and "daub4"), and symmetric quadrature mirror filters ("qmf5," "qmf9," and "qmf13").

Table 5.5 summarizes results (in terms of the standard deviation of residual noise) obtained for four of twelve test images of 256×256 pixels with 256 quantization levels and standard deviation of additive Gaussian noise equal to 13 (PSNR = 20): a test piecewise constant image, "Lena" image, an MRI image, and an air photograph. For DCT moving window filtering, optimal window size is

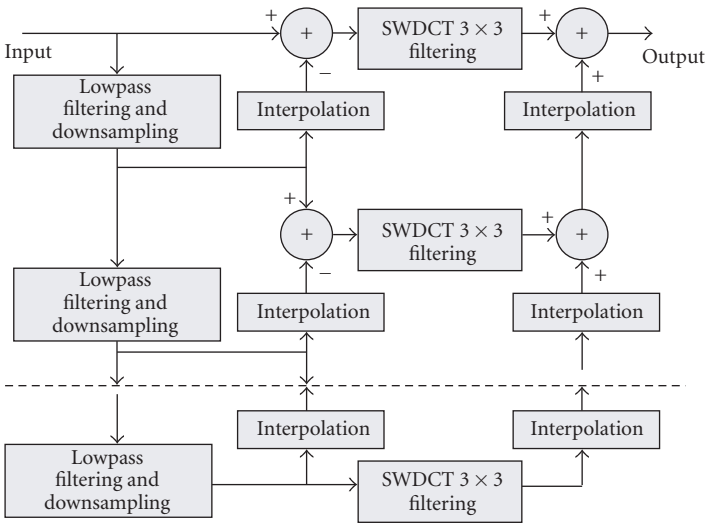


FIGURE 5.15. Flow diagram of hybrid wavelet/SWTD processing.

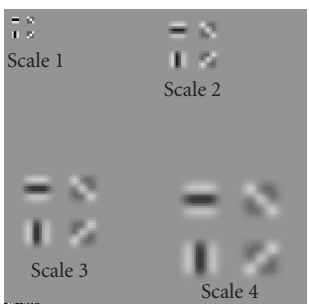


FIGURE 5.16. Basis functions of hybrid wavelet/sliding window processing with 4 directional Laplacians.

shown in brackets. For wavelet shrinkage, the best wavelet filter kernel is indicated in brackets. Figure 5.17 presents denoising results for visual comparison, which also confirms higher resulting image quality for the hybrid filtering.

5.5. Sliding window transform domain, wavelet and hybrid wavelet/SWTD filtering as versions of signal subband decomposition

In this section, we will show that a unified treatment of sliding window transform domain (SWTD), wavelet and hybrid wavelet/SWTD filtering is possible in terms of signal subband decomposition. For transform basis functions $\{\tau_n(r)\}$, direct and inverse transforms of a signal $\{a_l\}$ of N samples in sliding window of width

TABLE 5.5. Standard deviation of residual noise for different denoising methods and different test images. For wavelet shrinkage, type of the wavelet that provided the best result is indicated in brackets.

Filter		PW const. image	"Lena" image	MRI	Air photo
SWDCT	Hard thresholding	8.1 (3 × 3)	9.4 (3 × 3)	6.7 (5 × 5)	8.2 (3 × 3)
	Soft thresholding	7.5 (3 × 3)	**8.6** (5 × 5)	6.3 (7 × 7)	7.7 (3 × 3)
WL Shrinkage	Hard thresholding	8.6 (binom5)	10.1 (binom5)	8.5 (binom5)	9.3 (binom5)
	Soft thresholding	8.4 (binom5)	9.0 (qmf13)	7.8 (binom5)	8.1 (binom5)
Hybrid WL/SWDCT	Hard thresholding	8.7 (binom5)	9.4 (binom5)	6.6 (binom5)	8.2 (binom5)
	Soft thresholding	**7.9** (binom5)	**8.6** (binom5)	**6.2** (binom5)	**7.5** (binom5)

N_w centered at kth signal sample are

$$\begin{aligned} \alpha_r^{(k)} &= \sum_{n=0}^{N_w-1} a_{k+n-\mathrm{fix}(N_w/2)}\tau_n(r), \\ a_{k+n-\mathrm{fix}(N_w/2)} &= \sum_{r=0}^{N_w-1} \alpha_r^{(k)}\tilde{\tau}_r(n), \end{aligned} \tag{5.24}$$

where $\{\tilde{\tau}_r(n)\}$ are reciprocal basis functions orthogonal to $\{\tau_n(r)\}$ and operator $\mathrm{fix}(\cdot)$ denotes number rounding to its integer part.

For the window central sample

$$a_k = \sum_{r=0}^{N_w-1} \alpha_r^{(k)}\tilde{\tau}_r\left(\mathrm{fix}\frac{N_w}{2}\right), \tag{5.25}$$

$\{\alpha_r^{(k)}\}$ is "time" (signal domain)—"frequency" (transform domain) signal representation.

For fixed r, $\alpha_r^{(k)}$ is a vector of N samples. Compute its DFT over index k:

$$\mathbf{AT}_f^{(r)} = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}\alpha_r^{(k)}\exp\left(i2\pi\frac{kf}{N}\right) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N_w-1}\tau_n(r)\sum_{k=0}^{N-1}a_{k+n-\mathrm{fix}(N_w/2)}\exp\left(i2\pi\frac{kf}{N}\right). \tag{5.26}$$

Let

$$\mathbf{A}_f = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1}a_k\exp\left(i2\pi\frac{kf}{N}\right) \tag{5.27}$$

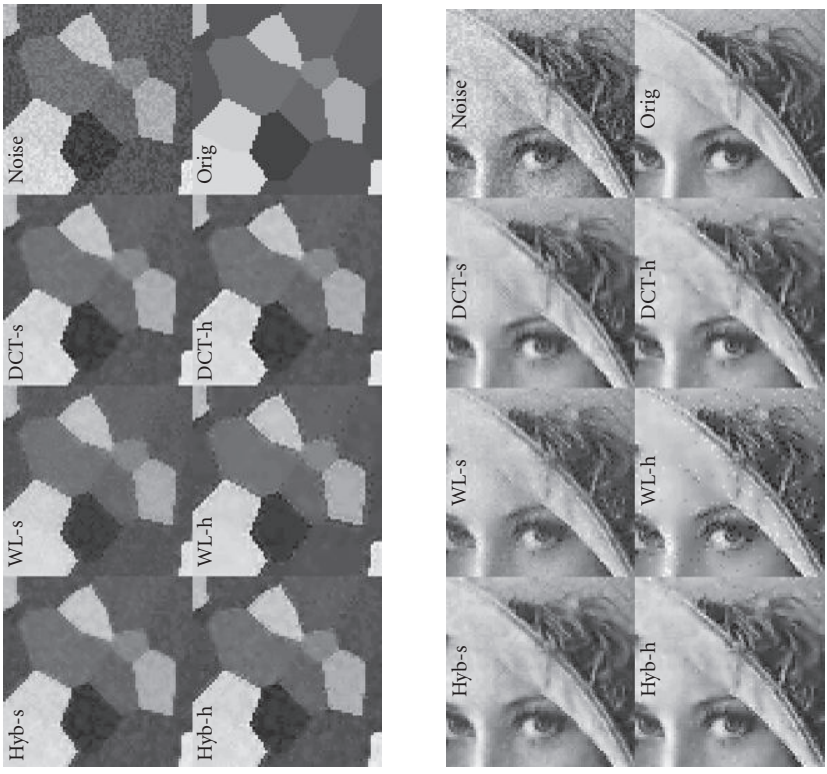


FIGURE 5.17. Comparison of wavelet shrinkage, SWDCT, and hybrid WL/SWDCT filtering for image denoising. Shown are 64×64 pixel fragments of the test piecewise constant (a) and "Lena" (b) images: originals (orig), corresponding fragments of the noisy images (noise), sliding window DCT filter outputs with hard and soft thresholds (DCT-h, DCT-s), wavelet shrinkage filtering outputs with hard and soft thresholds (WL-h, WL-s), and hybrid filtering outputs with hard and soft thresholds (Hyb-h, Hyb-s).

be DFT spectrum of signal $\{a_k\}$. By virtue of DFT shift theorem,

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_{k+n-\mathrm{fix}(N_w/2)} \exp\left(i2\pi \frac{kf}{N}\right) = \mathbf{A}_f \exp\left(-i2\pi \frac{n - \mathrm{fix}(N_w/2)}{N} f\right). \tag{5.28}$$

Then we obtain

$$\begin{aligned} \mathbf{A}\mathbf{T}_f^{(r)} &= \mathbf{A}_f \sum_{n=0}^{N_w-1} \tau_n(r) \exp\left(-i2\pi \frac{n - \mathrm{fix}(N_w/2)}{N} f\right) \\ &= \mathbf{A}_f \sum_{n=0}^{N-1} \mathrm{rect}\left(\frac{n}{N_w - 1}\right) \tau_n(r) \exp\left(-i2\pi \frac{n - \mathrm{fix}(N_w/2)}{N} f\right), \end{aligned} \tag{5.29}$$

where

$$\mathrm{rect}(x) = \begin{cases} 1 & 0 \le x \le 1, \\ 0 & \text{otherwise.} \end{cases} \tag{5.30}$$

From (5.29), it follows that

$$\mathbf{AT}_f^{(r)} \propto \mathbf{A}_f \cdot \mathbf{T}_f^{(r)}, \tag{5.31}$$

where

$$\mathbf{T}_f^{(f)} = \sum_{n=0}^{N-1} \mathrm{rect}\left(\frac{n}{N_w - 1}\right) \tau_n(r) \exp\left(-i2\pi \frac{nf}{N}\right) \tag{5.32}$$

is complex conjugate to DFT of rth windowed transform basis function.

Equation (5.31) has a clear interpretation: signal "time-frequency" representation $\{\alpha_r^{(k)}\}$ can, for a given index r, be regarded as signal bandpass filtering by the filter whose frequency response is DFT of the corresponding windowed basis function (5.32). In this sense, one can treat signal analysis with sliding window in transform domain as signal subband decomposition with the number of "bands" equal to the window size, the decomposition being performed by filters defined by (5.32). Figure 5.18 through 5.20 illustrate effective bandpass filter frequency responses for sliding window DCT, Walsh and Haar transforms. Figure 5.21 shows 14 subband components of a test image relevant in SWDCT processing for window size 7×7 pixels.

As it was already mentioned, wavelet signal transform can also be treated and implemented as signal subband decomposition with bandpass filters defined, on each wavelet decomposition scale, by the corresponding wavelet basis function. Figure 5.22 illustrates arrangement of subbands for binom5 wavelets for 8 scales of signals of 128 samples used in experiments with hybrid WL/SWDCT filtering described in Section 5.4.1.2.

One can easily see from comparison of Figures 5.18–5.20 and 5.22 that from the point of view of signal subband decomposition, the main difference between sliding window transform domain and wavelet signal analysis is arrangement of bands in the signal frequency range: while sliding window transform-domain signal analysis assumes uniform arrangement of subbands in the signal frequency range, in wavelet analysis subbands are arranged in a logarithmic scale. It is now obvious that hybrid wavelet/SWDCT signal analysis combines these two types of subband arrangements: wavelet decomposition provides "coarse" logarithmic scale subband decomposition which is complemented with "fine" uniform subsubbands within each wavelet subband provided by sliding window DCT analysis of the wavelet subbands. In particular, for hybrid WL/SWDCT filtering using DCT in the 3×3 pixel-window, each of subbands of wavelet decomposition is subdivided into subbands of DCT 3×3 shown in Figure 5.23. Resulting 2D subbands

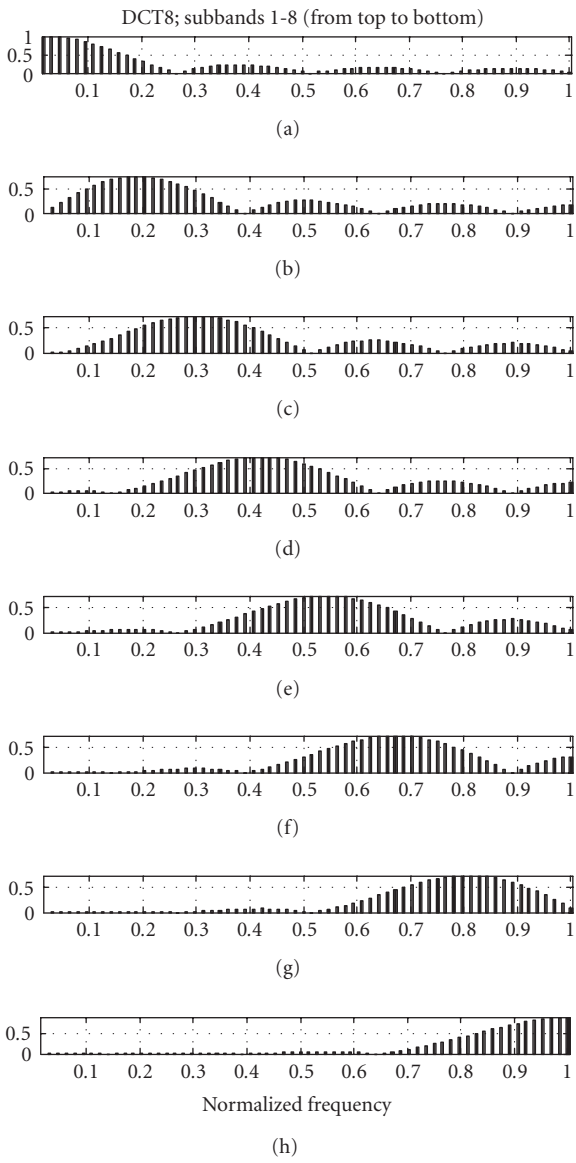


FIGURE 5.18. Frequency responses of bandpass filters that correspond to sliding window DCT (window size 8).

are illustrated in Figure 5.24. Shown are subbands for three last binom5 subbands (see Figure 5.22). One can see in the figure how these subbands are subdivided into 4 subbands that correspond to DCT 3×3 subbands.

Curiously enough, this "logarithmic coarse-uniform fine" band arrangement resembles very much the arrangements of tones and semitones in music. In Bach's

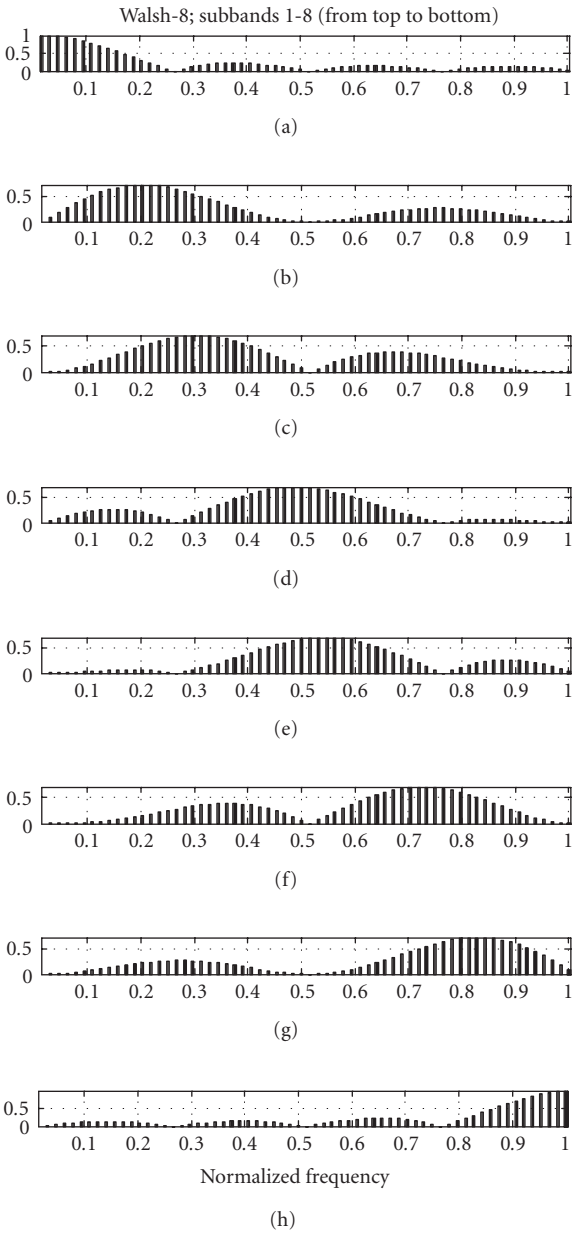


FIGURE 5.19. Frequency responses of bandpass filters that correspond to sliding window Walsh transform (window size 8).

equal-tempered scale, octaves are arranged in a logarithmic scale and 12 semitones are equally spaced within octaves (see [27]).

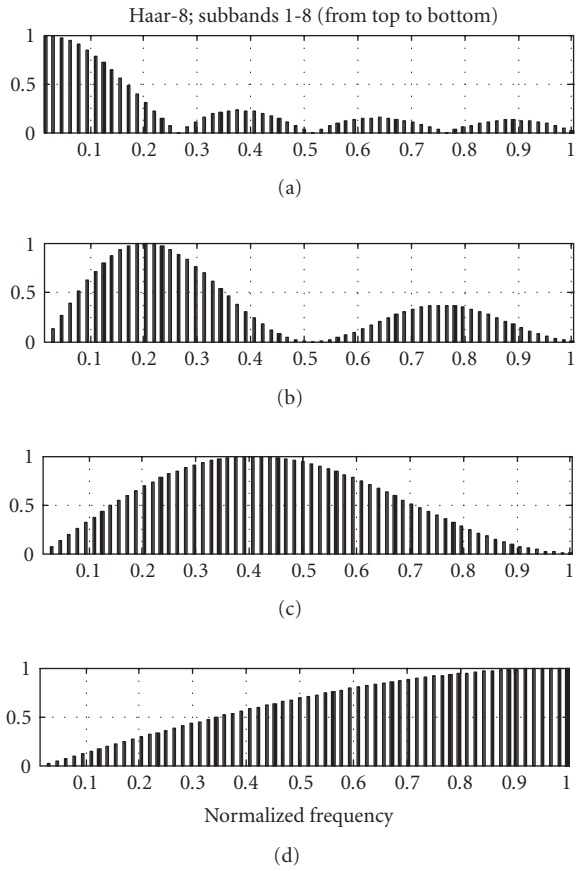


FIGURE 5.20. Frequency responses of bandpass filters that correspond to sliding window Haar transform (window size 8).

5.6. Conclusion

We have described a family of local adaptive image and video restoration filters that work in transform domain in sliding window, implement transform-domain scalar empirical Wiener filtering and are, in this way, optimized in terms of mean-square restoration error for each particular window position. We suggested discrete cosine transform (DCT) as the most appropriate transform for implementation of the filters, demonstrated application of the filters for image restoration (denoising and deblurring), and provided experimental data that confirm their high image denoising capability.

We also compared sliding window DCT domain filtering (SWDCT) with yet another existing transform-domain denoising method, wavelet shrinkage ones, and have shown that both methods can be treated in a unified way as hard/soft

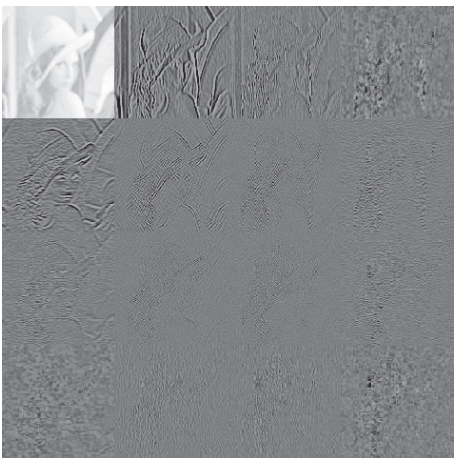


FIGURE 5.21. Subbands of Lena image for sliding window DCT domain image representation in 7×7 pixel-window. Only 16 bandpass filtered images that correspond to 16 (4×4) basis functions involved in the processing (functions with even indices) are shown arranged in lexicographic order from left to right and from top to bottom.

thresholdings of signal subband decompositions. In sliding window transform-domain methods, signal subbands are defined by Fourier transform of the transform basis functions involved in the processing and are uniformly arranged in the signal bandwidth. In wavelet methods, subbands are defined by wavelet basis functions and are arranged within the signal bandwidth in a logarithmic scale. We have shown experimental data that evidence in favor of SWDCT filtering in terms of the filter denoising capability, provided that the filter window size is selected appropriately. Its additional advantage is its capability of simultaneous image denoising deblurring and, as it is shown in Chapter 8, resampling in 1D, 2D, and 3D (multicomponent images and video) processing. As a way to avoid search of the filter optimal window size, we introduced a hybrid wavelet/sliding window processing and have shown that hybrid WL/SWDCT processing offers additional improvement of the filtering denoising capability.

What further advances in the development of the described family of filters are possible? Apparently, there are few options.

First of all, the gap in the denoising capability between empirical Wiener filtering implemented in SW filtering and "ideal" Wiener filtering can be narrowed by better spectra estimation than a simplistic "zero-order" approximation used in (5.13). For this, one can consider a recursive spectrum estimation, in which window spectral coefficients in several window positions adjacent to the current one are traced and used for spectrum estimation in the current window.

Second, described methods, being working in time(space)-frequency domain, apply only pointwise operations such as, for instance, shrinkage or zonal quantization, with (practically) no regards to the signal transform representation for

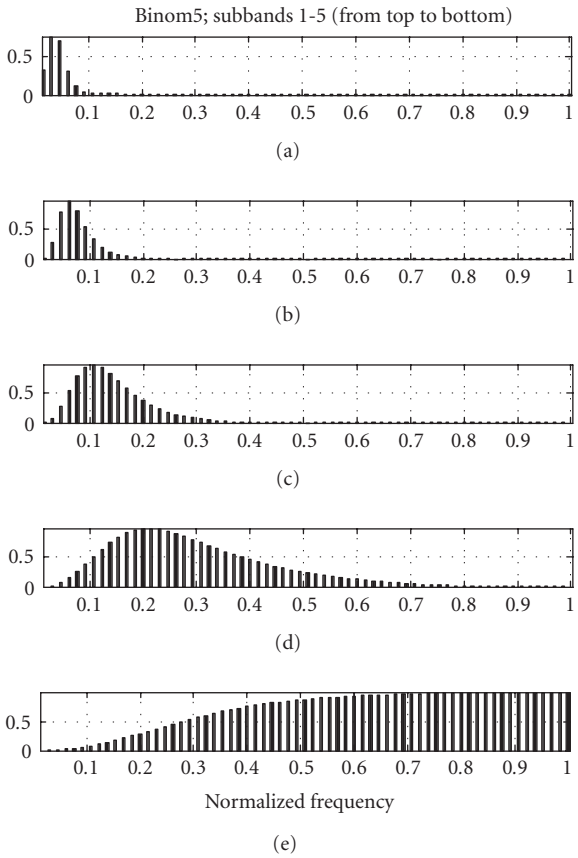


FIGURE 5.22. Frequency responses of five bandpass filters that correspond to binom5 wavelets ($N = 128$) and of two bandpass filters that correspond to DCT in the 3×3 window.

adjacent samples. Pointwise treatment of time(space)-frequency signal representation ignores enormous redundancy associated with the time-frequency signal representation in sliding window (note that in terms of the volume of data, this redundancy is window size-fold). This redundancy exhibits itself in a form of appearance in time-frequency domain of highly correlated and, to a certain degree, regular patterns that are characteristic for different signals and noises. These patterns can be much easier discriminated if they are treated, for 1D signals, as 2D (for image processing, 4D) ones rather then point-wise. This represents a substantial potential for improving time(space)-frequency domain signal processing efficiency (see [28]).

Yet another option is an extension of the overlap-and-add SWDCT filtering and translation-invariant wavelet filtering. One can apply, for restoration of the same signal, different transform-domain filters, such as SWDCT filters with different windows or sliding window or wavelet filters with different transforms or wavelets, and then combine the results using methods of data fusion. In the

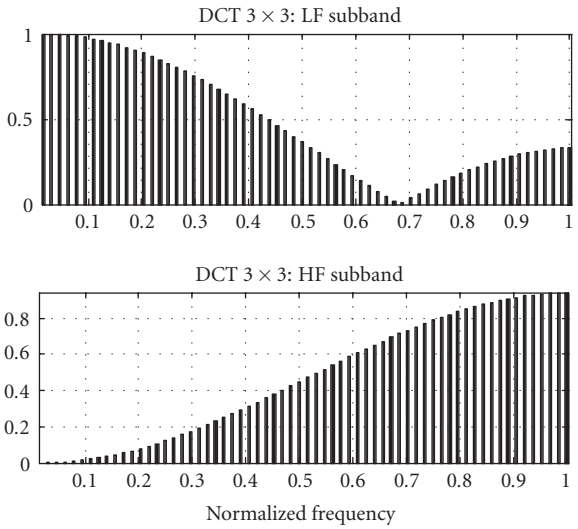


FIGURE 5.23. Frequency responses of two bandpass filters that correspond to DCT in the 3×3 window.

FIGURE 5.24. Last 3×4 2D subbands, shown in spatial frequency coordinates, of the hybrid WL/SWDCT filtering using binom5 wavelets and SWDCT in 3×3 window.

combining, simple averaging that is used in the overlap/add SWDCT filtering and translation-invariant wavelet filtering can be replaced by a pixelwise weighted averaging with optimization of weight coefficients to achieve better restoration result or by pixelwise interchannel rank filtering, such as median filtering, weighted order statistics filtering, or alike.

In 2D and 3D (video) processing, an immediate option is combining denoising/deblurring with simultaneous image resampling, as it is described in Chapter 8. A practical application of this idea to visible light range and thermal range video data fusion and stabilization of turbulent video was reported in [28, 29].

At last, one can mention recently proposed methods of sliding window transform-domain processing with adaptive selection of the window shape and size and shape-selective DCT (see [29]).

Appendix

A. Inverse DCT for computation of the central sample of the window of odd size

Let $\{\beta_r^{(k)}\}$ be signal spectral coefficients in the window of odd size N_w. Assume that pixels in the window in its kth position are indexed from k to $k + N_w - 1$. Compute inverse DCT transform of this spectrum $\{\beta_r^{(k)}\}$ for the window central pixel $\{a_{k+(N_w-1)/2}\}$:

$$\begin{aligned} a_{k+(N_w-1)/2} &= \beta_0^{(k)} + 2\sum_{r=1}^{N_w-1} \beta_r^{(k)} \cos\left(2\pi \frac{k + (N_w - 1)/2 + 1/2}{N_w} r\right) \\ &= \beta_0^{(k)} + 2\sum_{r=1}^{N_w-1} \beta_r^{(k)} \cos\left(\frac{\pi r}{2}\right) = \beta_0^{(k)} + 2\sum_{s=1}^{(N_w-1)/2} \beta_{2s}^{(k)} (-1)^s. \end{aligned} \tag{A.1}$$

It follows from (A.1) that computation of inverse DCT for the central sample of a window of odd size involves only signal spectrum coefficients with even indices. Therefore, only those spectral coefficients have to be computed in sliding window DCT domain filtering and the computational complexity of the filtering, where the use of recursive algorithms is $O[(N_w + 1)/2]$ per output pixel for 1D filtering, is $O[(N_{w1} + 1)(N_{w2} + 1)/4]$ operations for 2D filtering in a rectangular window of $N_{w1} \times N_{w2}$ samples and is $O[(N_{w1} + 1)(N_{w2} + 1)(N_{w2} + 1)/8]$ operations for 3D filtering in the window having the form of a rectangle parallelepiped of $N_{w1} \times N_{w2} \times N_{w3}$ samples.

Bibliography

[1] R. Y. Vitkus and L. Yaroslavsky, "Recursive algorithms for local adaptive linear filtration," in *Mathematical Research*, L. Yaroslavsky, A. Rosenfeld, and W. Wilhelmi, Eds., pp. 34–39, Academy, Berlin, Germany, 1987.

[2] D. Gabor, "Theory of communication," *Journal of the Institute of Electrical Engineers*, vol. 93, pp. 429–457, 1946.

[3] R. K. Potter, G. Koppand, and H. C. Green, *Visible Speach*, Van Nostrand, New York, NY, USA, 1947.

[4] L. Yaroslavsky, *Digital Signal Processing in Optics and Holography*, Radio i Svyaz', Moscow, Russia, 1987.

[5] R. Yu. Vitkus and L. Yaroslavsky, "Adaptive linear filters for image processing," in *Adaptive Methods for Image Processing*, V. I. Siforov and L. Yaroslavsky, Eds., pp. 6–35, Nauka, Moscow, Russia, 1988.

[6] L. Yaroslavsky, "Linear and rank adaptive filters for image processing," in *Digital Image Processing and Computer Graphics. Theory and Applications*, L. Dimitrov and E. Wenger, Eds., p. 374, R. Oldenburg, Vienna, Austria, 1991.

[7] L. Yaroslavsky, "Local adaptive filters for image restoration and enhancement," in *Proceedings of 12th International Conference on Analysis and Optimization of Systems: Images, Wavelets and PDE's*, M.-O. Berger, R. Deriche, and I. Herlin, Eds., Paris, France, June 1996.

[8] L. Yaroslavsky, "Local adaptive image restoration and enhancement with the use of DFT and DCT in a running window," in *Wavelet Applications in Signal and Image Processing IV*, vol. 2825 of *Proceedings of SPIE*, pp. 2–13, Denver, Colo, USA, August 1996.

[9] D. L. Donoho, "Nonlinear wavelet methods for recovery of signals, densities, and spectra from indirect and noisy data," in *Different Perspectives on Wavelets (San Antonio, TX, 1993)*, vol. 47 of *Proceedings of Symposia in Applied Mathematics*, pp. 173–205, American Mathematical Society, Providence, RI, USA, 1993.

[10] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation via wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[11] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, 1995.

[12] L. Yaroslavsky, *Digital Holography and Digital Image Processing*, Kluwer Academic, Boston, Mass, USA, 2004.

[13] L. Yaroslavsky and M. Eden, *Fundamentals of Digital Optics*, Birkhauser, Boston, Mass, USA, 1996.

[14] M. D. Levine, *Vision in Man and Machine*, McGraw-Hill, New York, NY, USA, 1985.

[15] H. S. Malvar and D. H. Staelin, "The LOT: transform coding without blocking effects," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 4, pp. 553–559, 1989.

[16] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, Boston, Mass, USA, 1990.

[17] K. G. Beauchamp, *Transforms for Engineers. A Guide to Signal Processing*, Clarendon Press, Oxford, UK, 1987.

[18] K. O. Egiazarian, "Running discrete orthogonal transforms and block adaptive LMS digital filters," *AMSE Review*, vol. 6, no. 3, pp. 19–29, 1988.

[19] R. Oktem, L. Yaroslavsky, and K. Egiazarian, "Signal and image denoising in transform domain and wavelet shrinkage: a comparative study," in *Proceedings of the 9th European Signal Processing Conference (EUSIPCO '98)*, S. Theodorisdis, I. Pitas, A. Stouraitis, and N. Kalouptsidis, Eds., pp. 2269–2272, Island of Rhodes, Greece, September 1998, Typorama editions.

[20] R. Oktem, L. Yaroslavsky, and K. Egiazaryan, "Evaluation of potentials for local transform domain filters with varying parameters," in *Proceedings of the 10th European Signal Processing Conference (EUSIPCO '00)*, M. Gabbouj and P. Kuosmanen, Eds., Tampere, Finland, September 2000.

[21] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall PTR, Englewood Cliffs, NJ, USA, 1995.

[22] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, New York, NY, USA, 1998.

[23] R. R. Coifman and D. L. Donoho, "Translation- invariant de-noising," in *Wavelets and Statistics*, A. Antoniadis, Ed., Lecture Notes, Springer, New York, NY, USA, 1995.

[24] R. Öktem, L. Yaroslavsky, K. Egiazarian, and J. Astola, "Transform domain approaches for image denoising," *Journal of Electronic Imaging*, vol. 11, no. 2, pp. 149–156, 2002.

[25] B. Z. Shaick, L. Ridel, and L. Yaroslavsky, "A hybrid transform method for image denoising," in *Proceedings of the 10th European Signal Processing Conference (EUSIPCO '00)*, M. Gabbouj and P. Kuosmanen, Eds., Tampere, Finland, September 2000.

[26] ftp://ftp.cis.upenn.edu/pub/eero/matlabPyrTools.tar.gz.

[27] J. Backus, *The Acoustical Foundations of Music*, Norton, New York, NY, USA, 1969.

[28] L. Yaroslavsky, K. Egiazaryan, and J. Astola, "Signal filtering in time-frequency domain: a next step forward," in *Proceedings of the IEEE-EURASIP Workshop on Nonlinear Signal Processing (NSIP '99)*, pp. 277–281, Antalia, Turkey, June 1999.

[29] L. Yaroslavsky, B. Fishbain, A. Shteinman, and Sh. Gepstein, "Processing and fusion of thermal and video sequences for terrestrial long range observation systems," in *Proceedings of the 7th International Conference on Information Fusion (FUSION '04)*, vol. 2, pp. 848–855, Stockholm, Sweden, June-July 2004.

[30] S. Gepshtein, A. Shtainman, B. Fishbain, and L. Yaroslavsky, "Restoration of atmospheric turbulent video containing real motion using rank filtering and elastic image registration," in *Proceedings of 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, September 2004.

[31] A. Foi, K. Dabov, V. Katkovnik, and K. Egiazarian, "Shape-adaptive DCT for denoising and image reconstruction," in *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, E. R. Dougherty, J. T. Astola, K. O. Egiazarian, N. M. Nasrabadi, and S. A. Rizvi, Eds., vol. 6064 of *Proceedings of SPIE*, San Jose, Calif, USA, January 2006.

[32] L. P. Yaroslavsky, et al., "Digital processing of Martian Surface Photographs from Mars-4 and Mars-5," *Cosmc Research*, vol. 13, no. 6, pp. 898–906, 1975.

L. Yaroslavsky: Department of Interdisciplinary studies, Faculty of Engineering, Tel Aviv University, Tel Aviv 69978, Israel

Email: yaro@eng.tau.ac.il

6 Adaptive varying window methods in signal and image processing

Vladimir Katkovnik, Karen Egiazarian, and Jaakko Astola

Two new approaches to signal reconstruction are introduced and demonstrated in this section: local polynomial approximation and adaptive window size selection based on the intersection of confidence interval rule. Local polynomial approximation is a technique of linear or nonlinear signal estimation using polynomial data fit in a sliding window. This technique is complemented with adaptive selection of the window size. The combined algorithm searches, in each window position, for a largest local vicinity of the current signal sample where the local polynomial approximation fit to the data remains to be satisfactory. The data estimates are found for a set of windows of different sizes and compared. The adaptive window size is defined as the largest one of the set for which the estimate does not differ significantly from the estimates for smaller windows, and the estimate obtained for this window is taken as the final current sample estimate. The considered local approximation can be regarded as nonparametric regression technique that places no a priori limits on the number of unknown parameters to model the signal. Basic concepts, methodology, and theory of spatially adaptive nonparametric regression-based signal and image processing are presented and their applications to various image processing problems are illustrated.

6.1. Introduction

This chapter is devoted to signal and image processing based on two independent ideas: *local approximation* for design of linear filters (estimators, transforms) and *adaptation* of these filters to a priori unknown smoothness of a signal of interest. Three key words associated with image analysis arise, to which we will give a more and more precise meaning:

(i) *locality*;
(ii) *anisotropy*;
(iii) *adaptivity*.

As flexible universal tools we use *local polynomial approximation* (LPA) for *approximation* and *intersection of confidence intervals* (ICI) for *adaptation*.

The LPA is applied for filter design using a polynomial fit in a sliding window. A size and shape of this *window* are considered as *basic adaptation parameters* of the filter.

The ICI is an *adaptation algorithm*. It searches for a largest local window size where LPA assumptions fit well to the observations. It is shown that the ICI adaptive LPA is efficient and allows to get a nearly optimal quality of estimation.

The LPA is applied in a sliding window and the adaptive size and shape of this window are defined for each estimation point. Thus, for each estimation point we have carefully selected adaptive neighborhoods used for processing.

One can say that the processing includes a pointwise segmentation of the data and this segmentation is produced with the main intention to optimize the accuracy of the result.

This adaptive polynomial fit is implemented as a MATLAB package LASIP (local approximation for signal and image processing) free available from the website http://www.cs.tut.fi/~lasip.

This package consists of a set of basic procedures implementing LPA and ICI methods and applied programs developed for particular applications. The main programs in LASIP are given as demo-procedures. However, these procedures and all routines are implemented as universal ones applicable for variety problems. It means that they can be used as routines of a universal MATLAB toolbox.

Pragmatically, we are focused on the ideas and the algorithm. Illustrations and references to LASIP procedures are aimed to help to realize the strength of the approach, possible fields of application and to use some of LASIP's programs.

To readers interesting in mathematical background and theory behind this approach to adaptive filtering we recommend our recent book [30].

6.2. Local approximation: ideas and algorithms

The idea of local smoothing and local approximation is very natural and appears in many branches of science. In sixties-seventies of the twentieth century the idea became a subject of an intensive theoretical study and applications: in statistics due to Nadaraya [36], Watson [50], Parzen [37], Stone [46], and in engineering sciences due to Brown [1], Savitzky and Golay [47], Petersen [38], Katkovnik [17–19], Cleveland [3].

The local polynomial approximation as a tool appears in different modifications and under different names: moving (sliding, windowed) least-square, Savitzky-Golay filter, reproducing kernels, moment filters, and so forth. We prefer the term LPA with a reference to publications on nonparametric estimation in mathematical statistics where the advanced development of this technique can be seen.

6.2.1. Windowing

The LPA is a universal method applicable for signals of any dimensionality. Nevertheless, in this chapter we prefer to talk about imaging assuming that all signals

and data are two-dimensional (2D). Imaging allows to give clear and transparent interpretation of algorithms as well as of results. Generalization to data of higher dimensionality is straightforward.

Suppose that we are given data (observations) of a signal (image intensity) y in the form $y_s = y(X_s)$, $s = 1,\dots,n$. Here X_s stays for a location of the sth observation. It is assumed in image processing that all signals are defined on a two-dimensional (2D) rectangular *regular grid*:

$$X = \{X_s : s = 1,\dots,n\} \tag{6.1}$$

with pixel's (grid node's) coordinates

$$X_s = \big(x_1(s), x_2(s)\big). \tag{6.2}$$

In this notation, image pixels are numbered by s taking values from 1 through n. The coordinates of these pixels can be given explicitly by the set

$$X = \{k_1, k_2 : k_1 = 1,\dots,n_1,\ k_2 = 1,\dots,n_2\}, \quad n = n_1 n_2, \tag{6.3}$$

where n is a total number of observations.

Noisy observations with an additive noise can be given in the following standard model commonly used for image and signal processing:

$$z_s = y(X_s) + \varepsilon_s, \quad s = 1,\dots,n, \tag{6.4}$$

where the additive noise ε_s is an error of the sth experiment usually assumed to be random, zero-mean independent for different s with $E\{\varepsilon_s\} = 0$, $E\{\varepsilon_s^2\} = \sigma^2$.

Let x be a "*center*" (desired or estimation point) and let a set U_x be neighborhood of x used for estimation.

A simplest estimate has a form of the sample mean

$$\hat{y}(x) = \frac{\sum_{s\in U_x} z_s}{\sum_{s\in U_x}} = \frac{\sum_{s\in U_x} z_s}{N_x}, \tag{6.5}$$

where $N_x = \sum_{s\in U_x}$ is a number of observations in the neighborhood U_x and the estimate is calculated as the sum of the observed signal values belonging to U_x divided by the number of observations.

The neighborhood U_x can be of different forms: square, rectangular, or circle. It is conventional to define a neighborhood by a special window function w. Then the neighborhood is a set of X_s from X where this window function is nonzero.

Mathematically it can be formulated as follows:

$$U_x = \{X_s : w(x - X_s) > 0\}. \tag{6.6}$$

Note that the window w is shifted by x and in this way define a neighborhood for any x.

Introduce a parameter h controlling a size of the neighborhood. It is convenient to do it through the window function using $w_h(x) = w(x/h)$.

Then the neighborhood of any size can be expressed through a fixed standard one. For instance, a disc of the radius h can be defined as

$$U_{x,h} = \{X_s : w_h(x - X_s) > 0\}, \tag{6.7}$$

where $w(x) = (1 - \|x\|^2)_+ = (1 - (x_1^2 + x_2^2))_+$. Here $(z)_+ = z$ if $z > 0$ $(z)_+ = 0$ if $z \le 0$.

In this case, the inequality $w(x - X_s) = (1 - \|x - X_s\|^2)_+ > 0$ defines a disc neighborhood of the radius 1 with the center x and the inequality $w_h(x - X_s) = (1 - \|x - X_s\|^2/h^2)_+ > 0$ defines a disc of the radius h.

The index h showing the size of the neighborhood is used also in notation for the neighborhood as $U_{x,h}$.

In windowing, the window function is used not only as an indicator of the neighborhood as it is in (6.7), but also as weights of observations used for estimation.

The window w is a function conventionally satisfying the following conditions:

$$w(x) \ge 0, \qquad w(0) = \max_x w(x), \qquad \int_{\mathbb{R}^2} w(x)dx = 1. \tag{6.8}$$

Thus the weights are nonnegative. The condition $w(0) = \max_x w(x)$ means that the maximum weight is given to the observation with $X_s = x$. This maximum could be achieved for a set of x close to $x = 0$. Then, the same weights are shared by a number of observations.

The values of w define comparative weights of the observations in the estimate. These weighted mean estimates have a form

$$\hat{y}_h(x) = \frac{\sum_s w_h(x - X_s)z_s}{\sum_s w_h(x - X_s)}. \tag{6.9}$$

Note that in (6.9) we do not need to indicate the neighborhood set $U_{x,h}$, and the summation can be shown as produced over all observations. Indeed, the window function $w_h(x - X_s)$ in (6.9) automatically separates the observations belonging to the corresponding neighborhood.

Note that the index h is introduced also for the estimate $\hat{y}_h(x)$.

If the window w is a rectangular function, all observations enter in the estimate with equal weights. Nonrectangular windows, such as triangular, quadratic, Gaussian, and so on, prescribe higher weights to observations z_s with X_s closer to the reference point x.

Let us mention windows conventional in signal processing: rectangular, triangular, Gaussian, Kaiser, Hamming, Bartlett, Blackman, Chebyshev, and so forth.

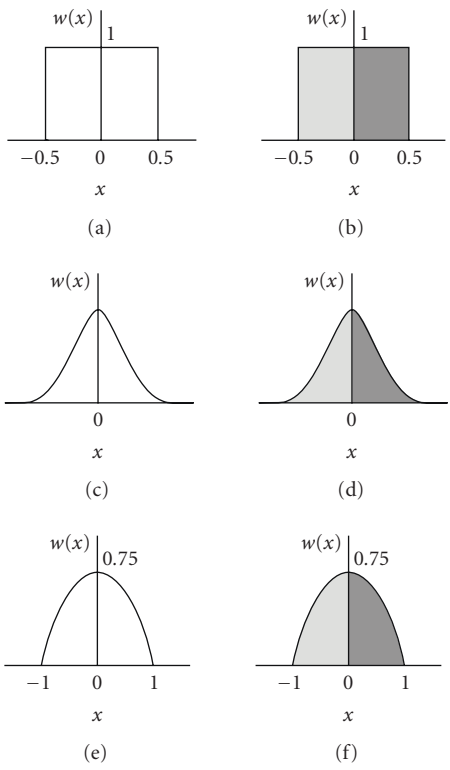


FIGURE 6.1. Left and right columns show, respectively, symmetric and nonsymmetric left/right windows: (a), (b) rectangular, (c), (d) Gaussian, (e), (f) quadratic.

These 1D windows can be used for derivation of 2D windows. A multiplicative window

$$w(x) = w_1(x_1)w_2(x_2), \tag{6.10}$$

where $w_1(x_1)$ and $w_2(x_2)$ are functions of scalar arguments, is commonly used for this purpose.

There is another way how to create nontrivial 2D windows different from the simple univariate ones (6.10).

Let us replace the argument in a 1D window by the norm $\|x\|$, where x is considered as a 2D vector and the norm is not exclusively Euclidian. Then, after normalization we obtain two-dimensional window functions satisfying conditions (6.8).

Examples of the 1D windows are shown in Figure 6.1. They are given in two versions: *symmetric* $w(x) = w(-x)$ and *nonsymmetric left/right windows*. The nonsymmetric windows are obtained from the symmetric ones. Let $w(x)$ be a symmetric window, then for the left window $w_L(x) = w(x)$ if $x \le 0$ and $w_L(x) = 0$ if $x > 0$, and for the right window $w_R(x) = w(x)$ if $x \ge 0$ and $w_R(x) = 0$ if $x < 0$.

Often the symmetry of the window is assumed by default in signal processing. However, the symmetry is not always a good choice and the nonsymmetry combined with the adaptive selection of the window size h can enable a serious improvement in results.

6.2.2. Local polynomial approximation

6.2.2.1. Zero-order approximation

Let us start from the zero-order approximation. It is assumed that the signal y is well approximated by a constant in some neighborhood of the point of interest x. We find this constant by the weighted least-square method (WLSM) as follows

$$\widehat{C} = \arg\min_{C} J_h(x, C), \tag{6.11}$$

$$J_h(x, C) = \sum_s w_h(x - X_s)(z_s - C)^2. \tag{6.12}$$

Here $J_h(x, C)$ is a quadratic criterion where the squared residuals $(z_s - C)^2$ are weighted according to values of the window function $w_h(x - X_s)$ and the estimate of C is found by minimizing this criterion.

Elementary calculations show that

$$\partial_C J_h(x, C) = \sum_s w_h(x - X_s)\partial_C(z_s - C)^2 = -2\sum_s w_h(x - X_s)(z_s - C) = 0. \tag{6.13}$$

This equation gives the estimate for C in the form

$$\widehat{C} = \frac{\sum_s w_h(x - X_s)z_s}{\sum_s w_h(x - X_s)}. \tag{6.14}$$

According to the idea of this approach, this $\widehat{C}$ is an estimate of the function y at the point x, that is, $\hat{y}_h(x) = \widehat{C}$.

This proves (6.9) for the weighted sample mean as the estimate minimizing the weighted squared error.

Let us assume that $x \in X$, then (6.9) is a convolution:

$$\hat{y}_h(x) = \sum_s g_h(x - X_s)z_s, \tag{6.15}$$

$$g_h(x) = \frac{w_h(x)}{\sum_x w_h(x)}, \tag{6.16}$$

with the invariant kernel (impulse response) function $g_h(x)$ depending in (6.15) only on the difference of the arguments.

We will use the standard notation for convolution

$$\hat{y}_h(x) = (g_h \circledast z)(x), \tag{6.17}$$

which says that this convolution is calculated on the regular grid X with x as an estimation point.

6.2.2.2. High-order approximation

The zero-order estimation is a special case of the general polynomial approximation.

It is assumed that the signal y is well approximated by a polynomial in some neighborhood of the point of interest x. We find the coefficients of the polynomial fit by WLSM and use this approximation in order to calculate the estimate for the point of interest x called *center* of the LPA.

Note that the term *center* does not assume a central position of x with respect to neighboring observations. This term aims to stress that the LPA is applied in order to obtain the estimate for this particular point x.

Formally, the LPA estimate is defined as follows. First, introduce the polynomial model of y in the form

$$\hat{y}(x) = \sum_{i=0}^{M} C_i \phi_i(x), \tag{6.18}$$

where $\phi_i(x)$ are some polynomial functions of x_1 and x_2, C_i are unknown parameters of the model, and $(M+1)$ is a total number of polynomials used in the fit.

Let us use a linear model $\hat{y}(x) = C_0 + C_1 x_1 + C_2 x_2$, then $\phi_0(x) = 1$, $\phi_1(x) = x_1$, $\phi_2(x) = x_2$ and overall we have three parameters C_0, C_1, C_2 defining the model.

Of course higher-order polynomials can be used in (6.18). It is convenient to represent this model in the vector form

$$\hat{y}(x) = C^T \phi(x), \tag{6.19}$$

where C is $(M+1)$-vector of the parameters and $\phi(x)$ is $(M+1)$-vector function composed from the polynomials ϕ_i.

Similar to (6.11) the criterion and the estimation are represented in the form

$$\widetilde{C}_h(x) = \arg\min_C \widetilde{J}_h(x, C), \tag{6.20}$$

$$\widetilde{J}_h(x, C) = \sum_s w_h(x - X_s)\left(z_s - \hat{y}(X_s)\right)^2. \tag{6.21}$$

Minimizing the criterion (6.20), we find the parameters $\widehat{C}_h(x)$ defining our model. The argument x in this estimate is used in order to emphasize that this estimate is calculated for the estimation point x. According to the idea of the local approximation, this estimate is used only for this point.

Finally, the estimate of the function has the following form:

$$\hat{y}_h(x) = \tilde{C}_h^T(x)\phi(x). \tag{6.22}$$

Usually a different local modeling is used. It is assumed that the estimates for z_s in (6.21) are represented in the form

$$\hat{y}(x - X_s) = C^T\phi(x - X_s) \tag{6.23}$$

and then the criterion is

$$J_h(x, C) = \sum_s w_h(x - X_s)\left(z_s - \hat{y}(x - X_s)\right)^2. \tag{6.24}$$

This shift in the argument of the polynomial function is used in order to emphasize the locality of the fit. As a result, the estimate of the function takes a form

$$\begin{gathered}\hat{y}_h(x) = \hat{C}_h^T(x)\phi(x - X_s)\,|_{x=X_s} = \hat{C}_h^T(x)\phi(0),\\ \hat{C}_h(x) = \arg\min_C J_h(x, C).\end{gathered} \tag{6.25}$$

This form of the estimate emphasizes a nonparametric nature of the estimate, where the dependence on the argument x exhibits itself through the coefficient $\hat{C}_h(x)$ of the fit, but not through the argument of the polynomials $\phi(x)$ as it is in the standard LSM.

The estimates (6.22) and (6.25) are identical provided that the set of the polynomials in $\phi(x)$ is complete. This completeness means the following. If m_1 and m_2 are the highest degrees of the polynomials $x_1^{k_1}$ and $x_2^{k_2}$ in the set, then all polynomials of the powers k_1 and k_2 with all $k_1 \le m_1$ and $k_2 \le m_2$ should be included in the set (in vector $\phi(x)$) as well as all their pairwise combinations.

For instance, if $m_1 = 1$ and $m_2 = 0$, the set of polynomials includes $\phi_0(x) = 1$, $\phi_1(x) = x_1$. If $m_1 = 1$ and $m_2 = 1$, the set includes $\phi_0(x) = 1$, $\phi_1(x) = x_1$, $\phi_2(x) = x_2$, and $\phi_3(x) = x_1x_2$.

The completeness of the polynomial set is always required as it will be discussed later.

In what follows, we use the LPA model in the difference form (6.23)-(6.24) with the estimate given by the formula (6.25).

The minimum conditions for $J_h(x, C)$ have a form $\partial_C J_h(x, C) = 0$. By calculating this derivative on the vector-parameter C, we arrive to the normal set of the equations

$$\begin{gathered}\Phi_h\hat{C} = \sum_s w_h(x - X_s)\phi_h(x - X_s)z_s,\\ \Phi_h = \sum_s w_h(x - X_s)\phi_h(x - X_s)\phi_h^T(x - X_s).\end{gathered} \tag{6.26}$$

Provided that the matrix Φ_h is nonsingular, $\det \Phi_h \neq 0$, the solution can be written as

$$\widehat{C} = \Phi_h^{-1} \sum_s w_h(x - X_s)\phi_h(x - X_s)z_s. \tag{6.27}$$

Inserting $\widehat{C}$ into (6.25), we find the signal estimate as an output of the linear shift-invariant filter:

$$\widehat{y}_h(x) = \sum_s g_h(x - X_s)z_s, \tag{6.28}$$

$$g_h(x - X_s) = w_h(x - X_s)\phi_h^T(x - X_s)\Phi_h^{-1}\phi(0), \tag{6.29}$$

where $g_h(x - X_s)$ is the impulse response of this filter for a fixed $x \in X$.

For $x \in X$ this estimate can be rewritten as the convolution (6.17) with a quite essential difference that the kernel g_h is defined by the high-order LPA:

$$\begin{aligned} g_h(x) &= w_h(x)\phi_h^T(x)\Phi_h^{-1}\phi(0), \\ \Phi_h &= \sum_s w_h(x)\phi_h(x)\phi_h^T(x). \end{aligned} \tag{6.30}$$

We will call these kernels smoothing or filtering kernels because they give the low-pass filters and are used for smoothing and denoising.

The terms *smoothing parameter* or *bandwidth* are used for h as equivalent to the initially introduced term *the window size*.

The polynomial approximation gives a natural way to obtain the estimates not only of the functions but also of their derivatives.

Let the polynomials of the powers $k_1 \leq m_1$ and $k_2 \leq m_2$ be used. Then the derivatives $\partial_{x_1}^{r_1}\partial_{x_2}^{r_2}$ of the orders $r_1 \leq m_1$ and $r_2 \leq m_2$ can be estimated.

In order to derive these differentiation operators let us differentiate both sides of (6.19) $\partial_{x_1}^{r_1}\partial_{x_2}^{r_2}\widehat{y}(x) = C^T\partial_{x_1}^{r_1}\partial_{x_2}^{r_2}\phi(x)$.

Using in this model the estimates of C is a natural way to derive the derivative estimate as follows:

$$\widehat{y}_h^{(r_1,r_2)}(x) = \widetilde{C}^T\partial_{x_1}^{r_1}\partial_{x_2}^{r_2}\phi(x), \tag{6.31}$$

where $\widetilde{C}$ is found from (6.20) and the indices of the estimate show the order of the derivative.

For the difference LPA model (6.23) we arrive to a different form of the derivative estimate:

$$\widehat{y}_h^{(r_1,r_2)}(x) = \widehat{C}^T\partial_{x_1}^{r_1}\partial_{x_2}^{r_2}\phi(0), \tag{6.32}$$

where $\widehat{C}$ is found from (6.25).

Let $x \in X$, then it can be checked that the estimate (6.32) can be represented in the convolution form with the shift-invariant kernel

$$\hat{y}_h^{(r_1,r_2)}(x) = \sum_s g_h^{(r_1,r_2)}(x - X_s) z_s = \left(g_h^{(r_1,r_2)} \circledast z\right)(x), \tag{6.33}$$

$$g_h^{(r_1,r_2)}(x - X_s) = w_h(x - X_s)\phi_h^T(x - X_s)\Phi_h^{-1}\partial_{x_1}^{r_1}\partial_{x_2}^{r_2}\phi(0), \tag{6.34}$$

where $\partial_{x_1}^{r_1}\partial_{x_2}^{r_2}\phi(0) = \partial_{x_1}^{r_1}\partial_{x_2}^{r_2}\phi(x)|_{x=0}$.

We will call these kernels *differentiating kernels* because they are used for estimation of derivatives.

6.2.3. Accuracy

The LPA accuracy is characterized by an error defined as a difference between the true signal value and the estimate:

$$e_y(x,h) = y(x) - \hat{y}_h(x). \tag{6.35}$$

For the observations (6.4), these errors are composed of the systematic (bias) and random components corresponding to the deterministic y and the random noise ε, respectively.

The bias of the estimate is a difference between the true signal and the expectation of the estimate:

$$m_{\hat{y}_h}(x,h) = y(x) - E\{\hat{y}_h(x)\}, \tag{6.36}$$

where $E\{\cdot\}$ means the mathematical expectation with respect to the random error ε in (6.4).

Elementary calculations give

$$E\{\hat{y}_h(x)\} = E\{(g_h \circledast z)(x)\} = (g_h \circledast E\{z\})(x) = (g_h \circledast y)(x). \tag{6.37}$$

Then, the bias of the estimates is

$$m_{\hat{y}_h}(x,h) = y(x) - (g_h \circledast y)(x). \tag{6.38}$$

The random estimation errors are

$$e_y^0(x,h) = -(g_h \circledast \varepsilon)(x). \tag{6.39}$$

Assuming that the random ε are independent and identically distributed (i.i.d.) with variance σ^2, we obtain for the variance of the estimate $\hat{y}_h(x)$

$$\sigma_{\hat{y}_h}^2(x,h) = E\{(e_y^0(x,h))^2\} = \sigma^2 \sum_{k\in\mathbb{Z}^d} g_h^2(k) = \sigma^2 ||g_h||^2. \tag{6.40}$$

The total accuracy defined as the expectation of the squared error is

$$E\{e_y^2(x,h)\} = m_{\hat{y}_h}^2(x,h) + \sigma_{\hat{y}_h}^2(x,h). \tag{6.41}$$

A similar formula is valid for the derivative estimate $\hat{y}_h^{(r_1,r_2)}(x)$ with the only difference that the smoothing kernel g_h is replaced by the differentiating kernel $g_h^{(r_1,r_2)}$.

6.2.4. Implementation of kernel design in LASIP

Calculation of the kernels g_h (6.30) and $g_h^{(r_1,r_2)}$ (6.34) is implemented in *LASIP* in the program *demo_CreateLPAKernels.m*.

Input parameters of this program are as follows:

(1) the order of LPA $m = [m_1, m_2]$, where m_1 and m_2 are the LPA orders on the variables x_1 and x_2, respectively;
(2) the window size is defined by two parameters: h_1 for length (or the size in the variable x_1) and h_2 for width (or the size in the variable x_2);
(3) the window function w is defined by the "*window_type*" parameter allowing to select w from a list of the window functions;
(4) the parameter TYPE specifies an indicator function used as a factor of the window function. TYPE = 00 means that the whole window function w is used in the kernels. TYPE = 10 means that the window w is used for $x_1 \geq 0$ and any x_2 and TYPE = 11 means that the window w is used for $x_1 \geq 0$ and any $x_2 \geq 0$.

Let $w(x)$ be a symmetric function, say Gaussian: $w(x) \sim \exp(-\|x\|^2)$. With TYPE = 10 a half of this window function is used with $x_1 \geq 0$, with TYPE = 11 a quarter of this window function is used with $x_1, x_2 \geq 0$. Only with TYPE = 00 the whole window function is used for all x_1 and x_2. The parameter TYPE is used for design of symmetric and nonsymmetric kernels and defines a type of the nonsymmetry of the kernel.

The program returns the smoothing kernel and the set of the differentiating kernels corresponding to the given power $m = [m_1, m_2]$. The differentiating kernels $g_h^{(k_1,k_2)}$ are calculated for all partial derivatives, that is, $0 \leq k_1 \leq m_1$, $k_2 = 0$ and for $0 \leq k_2 \leq m_2$, $k_1 = 0$, and for the mixed derivatives such that $k_1 + k_2 \leq \max(m_1, m_2)$.

The zero-order $r_1 = r_2 = 0$ of the differentiating kernel $g_h^{(0,0)}$ corresponds to the smoothing kernel g_h.

Thus, the program is universal for design of kernels (filters) for smoothing and differentiation which could be obtained using the approximation of the power $m = [m_1, m_2]$.

The program *utility_DrawLPAKernels.m* produces automatic visualization of these results. It draws images of all kernels $g_h^{(k_1,k_2)}$ and their 3D surface curves. It draws also the amplitude frequency characteristics of the kernels.

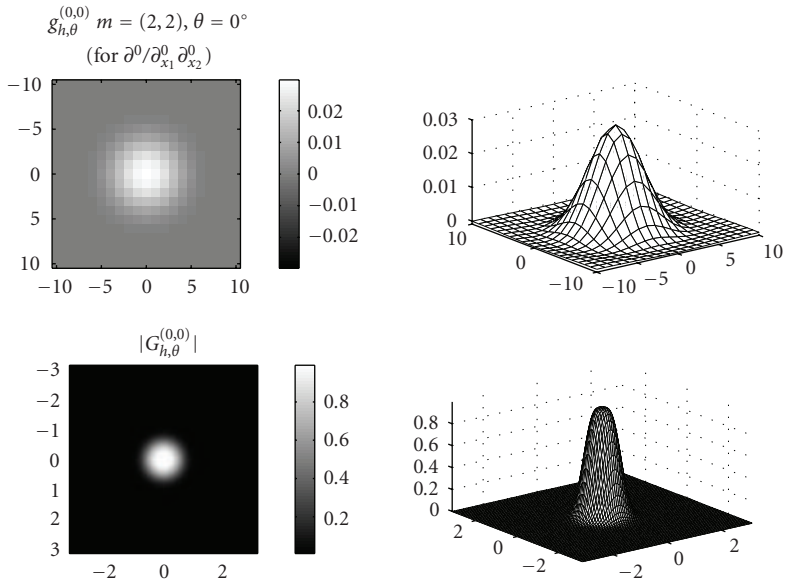


FIGURE 6.2. The smoothing kernel g_h and its amplitude frequency characteristic $|G_h|$: Gaussian symmetric window, $m = [2, 2]$, support size 21×21. The lowpass filter is shown with a peak of the frequency characteristic at $\omega = 0$.

Let us illustrate the work of these programs. It is convenient to use for demonstration the Gaussian window w as it gives smooth good looking surfaces and the frequency characteristics enabling a clear interpretation.

Assume that $m = [2, 2]$, the size parameters are $h_1 = h_2 = 11$ and TYPE = 00. It gives the symmetric window function of the square support 21×21. The size of the symmetric window support is defined as $2h_k - 1$, $k = 1, 2$, while for the nonsymmetric one it is equal to h_k, $k = 1, 2$. The program calculates a set of the six kernels $g_h^{(k_1,k_2)}$ corresponding to $k_1 = k_2 = 0$; $k_1 = 1$, $k_2 = 0$; $k_1 = 2$, $k_2 = 0$; $k_1 = 0$, $k_2 = 1$; $k_1 = 0$, $k_2 = 2$; $k_1 = 1$, $k_2 = 1$. Some of these results can be seen in Figures 6.2–6.5.

The smoothing kernel g_h in Figure 6.2 is a symmetric function with a peak at $x_1 = 0$, $x_2 = 0$. Its frequency characteristic is typical for the lowpass filter with a peak at $\omega_1 = 0$, $\omega_2 = 0$.

The differentiation kernels $g_h^{(1,0)}$, $g_h^{(0,1)}$ in Figures 6.3 and 6.4 are directional forming weighted finite-differences in the directions of the estimated derivatives $\partial^{(1,0)}$, $\partial^{(0,1)}$, respectively, for the arguments x_1 and x_2. The frequency characteristics have zero values at $\omega_1 = 0$, $\omega_2 = 0$. These differentiation filters are bandpass filters. The frequency characteristics are directional with two distinctive peaks.

The differentiation kernel $g_h^{(1,1)}$ for estimation of the cross-derivative $\partial^{(1,1)}$ is shown in Figure 6.5. Its frequency characteristic has zero value at $\omega_1 = 0$, $\omega_2 = 0$ and four distinctive peaks.

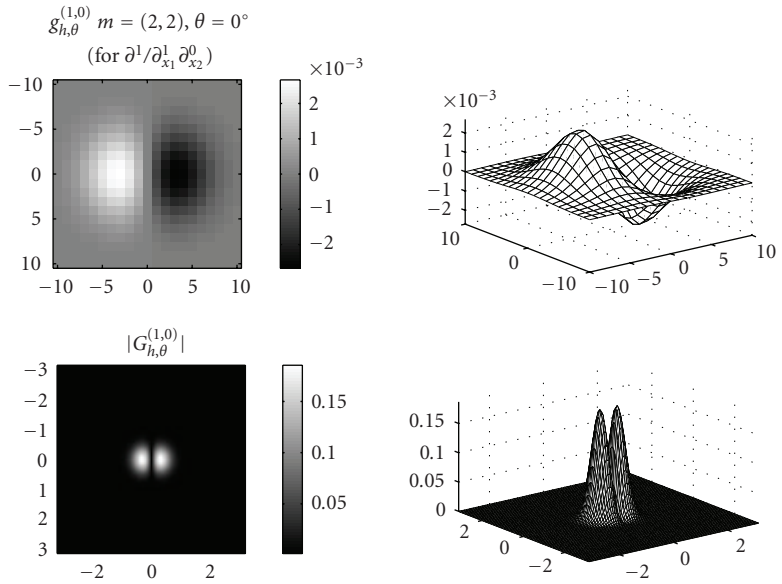


FIGURE 6.3. The differentiating kernel $g_h^{(1,0)}$ and its amplitude frequency characteristic $|G_h^{(1,0)}|$: Gaussian symmetric window, $m = [2,2]$, support size 21×21. The frequency characteristic is equal to zero at $\omega = 0$.

Kernels with nonsymmetric quadrant Gaussian windows are shown in Figures 6.6 and 6.7. The supports of the kernels are restricted to the quadrant of the standard Cartesian coordinate system. The frequency characteristics of these kernels are quite similar to the frequency characteristics of the corresponding symmetric window kernels shown in Figures 6.2 and 6.5, respectively.

6.3. Adaptive window size

6.3.1. Motivation

Let us use the smoothing LPA operators with kernels g_h for filtering noise from noisy data. Compare estimators with different window sizes. If $h_1 = h_2 = 1$, the kernel $g_h(x) \simeq \delta(x)$, and the estimate $\hat{y}_h(x) \simeq z(x)$, then there is no filtering at all and the output signal $\hat{y}_h(x)$ is identical to observations. The estimator $h_1, h_2 > 1$ smooths the input signal z and it means smoothing for both the noise and the signal itself.

It is obvious that larger h_1, h_2 mean stronger smoothing. Thus, we are able to filter out the noise but at the price of some smoothing of the signal. Always there is a reasonable compromise between noise attenuation and signal degradation caused by this filtering and this compromise indicates a reasonable choice for an adequate value of the smoothing parameters h_1, h_2.

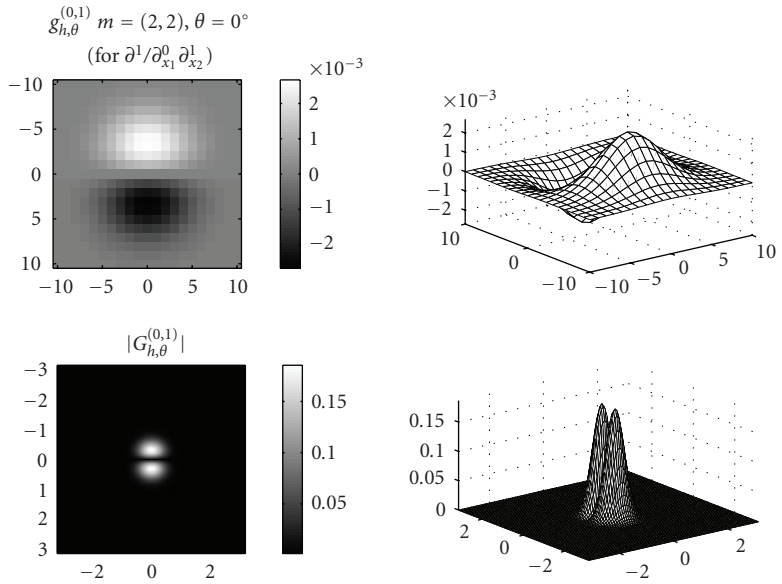


FIGURE 6.4. The differentiating kernel $g_h^{(0,1)}$ and its amplitude frequency characteristic $|G_h^{(0,1)}|$: Gaussian symmetric window, $m = [2,2]$, support size 21×21. The frequency characteristic is equal to zero at $\omega = 0$.

Images in Figure 6.8 give illustration of this role of the smoothing parameter. In Figures 6.8(a), 6.8(b) we can see the true and noisy images. This noisy image is identical to the output signal of the filter with $h_1 = h_2 = 1$.

Larger values $3, 5, 7, 11$ of $h_1 = h_2$ result in obvious noise filtering and image oversmoothing for larger values of h (Figures 6.8(c)–6.8(g)). The values of RMSE given for these smoothed images show that the optimal smoothing is achieved with $h = 2$, see Figure 6.8(c). Comparing the value of RMSE for different h, we may note that indeed this optimization results in essential decreasing of the RMSE value. However, visually improvement with respect to the observed noisy data is not impressive and the achieved image quality is quite poor.

Much better results can be obtained by selecting a varying window size optimizing the accuracy for each pixel. It results in the optimal window size for each pixel and in the smoothing parameter depending on the argument x as $h = h(x)$. These results obtained by the ICI algorithm discussed later in this section are shown in Figure 6.8(h). The obtained RMSE $= 10.14$ is much smaller the best results achieved by the selection of the global invariant window size $h = 2$ giving RMSE $= 15.77$. Visual improvement of filtering is also quite remarkable.

Note that this adaptive filtering is produced using the symmetric kernel shown in Figure 6.2 with selection of the adaptive window sizes from a given set of the window sizes $H = [1, 2, 3, 5, 7, 11]$.

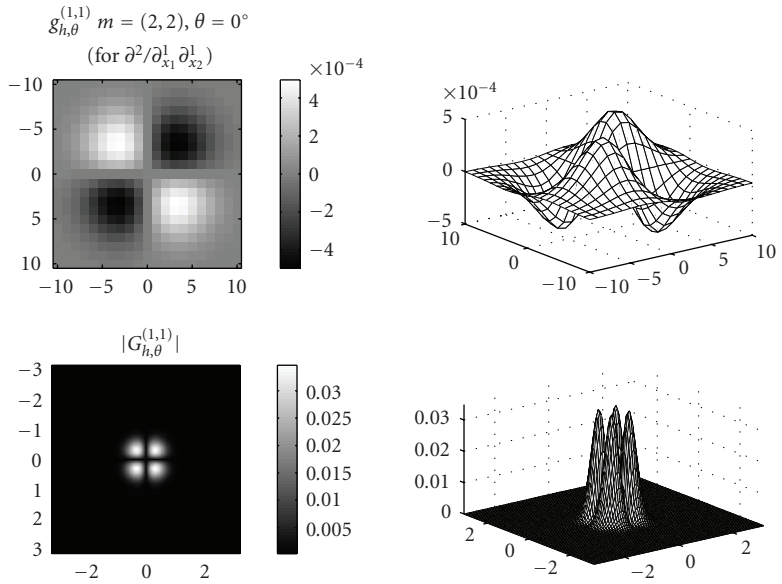


FIGURE 6.5. The differentiating kernel $g_h^{(1,1)}$ and its amplitude frequency characteristic $|G_h^{(1,1)}|$: Gaussian symmetric window, $m = [2,2]$, support size 21×21. The frequency characteristic is equal to zero at $\omega = 0$.

Figure 6.9 gives a further illustration of the adaptive varying window sizes obtained for filtering and demonstrated in Figure 6.8(h). Black and white in this image correspond to smaller and larger window sizes. The bar in this image shows the numerical values of the adaptive window sizes. We may note that near the edges of the image where the image intensity is discontinuous the adaptive window sizes automatically take the smallest values. Far from this edge, the window sizes are mainly large with isolated black points showing noise effects resulting in small window sizes taken by the adaptive algorithm erroneously for the areas where the large window size would be more appropriate.

The last image in Figure 6.8(i) demonstrates a further remarkable improvement if filtering is achieved using a special nonsymmetric directional filtering presented later in Section 6.4. This last filtering allows to achieve much better results with smaller RMSE = 3.71 and much better visual quality of the denoised imaging.

The presented results illustrate our main idea and intention to introduce the varying window size filtering with a special choice of the smoothing parameter giving the results close to the best possible.

A selection of a proper window size (smoothing, scale) parameter is a hot topic both in signal processing and statistics.

Two main groups of techniques are exploited. The first one is based on estimation of the bias and the variance with scale calculation using the theoretical formulas for the optimal window size. This sort of methods is known as "*plug-in*"

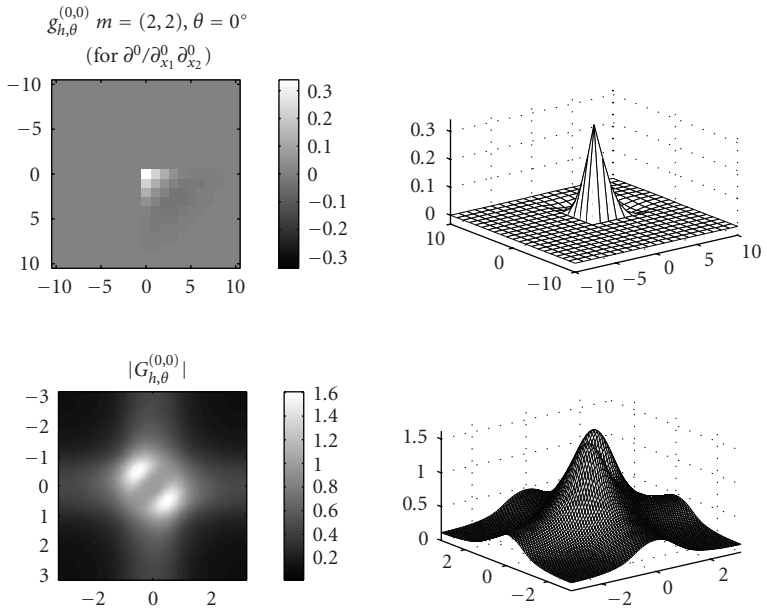


FIGURE 6.6. The smoothing kernel g_h and its amplitude frequency characteristic $|G_h|$: Gaussian nonsymmetric quadrant window, $m = [2, 2]$, support size 21×21. The frequency characteristic is equal to one at $\omega = 0$.

methods. Overall, these methods give smooth curves with good filtering of random errors. However, the estimate bias depends on unknown high-order derivatives of the signal. As a result, the algorithms are quite complex and have a number of parameters to be tuned, in particular, for estimation of these derivatives.

The second alternative idea has no deal with the bias and the formulas for the optimal scale selection. This is a group of methods based on *quality-of-fit* statistics such as *cross-validation*, *generalized cross-validation*, C_p, *Akaike criteria*, and so forth, which are applied for a model selection or direct optimization of estimation accuracy.

Most of publications concerning the quality-of-fit approach are related to a data-based global (constant) scale selection (e.g., [15, 16, 21, 49]).

The problem of varying window size pointwise adaptation has received a powerful impetus in connection with a number of novel ideas that have appeared in mathematical statistics. Various developments of these ideas and various statistical rules for adaptation can be seen in [34, 35, 39–42, 44], and so forth.

These methods are from a class of quality-of-fit statistics applied locally in a pointwise manner.

Methods known under a generic name *Lepski's approach* [31–34] are of special interest for us, as the ICI rule used in this chapter as a main adaptation tool belongs to this group of methods.

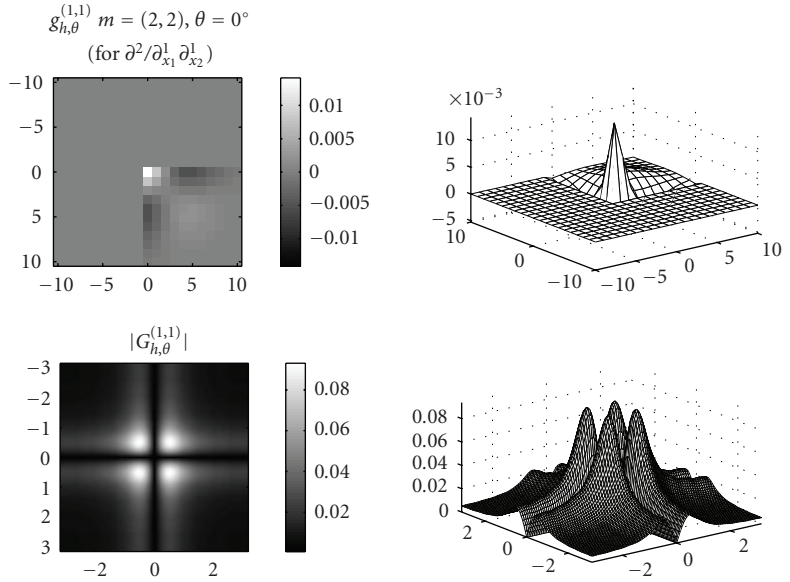


FIGURE 6.7. The differentiating kernel $g_h^{(1,1)}$ and its amplitude frequency characteristic $|G_h^{(1,1)}|$: Gaussian nonsymmetric quadrant window TYPE = 11, $m = [2,2]$, support size 21×21. The frequency characteristic is equal to zero at $\omega = 0$.

Let us present the idea of *Lepski's approach*. In what follows, the kernel of the estimator depends on a single scalar parameter h. In particular, it can be $h_1 = h_2 = h$.

Introduce a set of possible values of this scalar h:

$$H = \{h_{(1)} < h_{(2)} < \cdots < h_{(J)}\} \tag{6.42}$$

starting from a small $h_{(1)}$ and increasing to a maximum $h_{(J)}$, and let $\hat{y}_{h_i}$ be the LPA estimate of y calculated for $h_i \in H$.

For small h the estimate $\hat{y}_h$ has a small bias and a large standard deviation. The adaptive scale algorithm compares the estimates of increasing h with the main intention to find the maximum scale when the estimate's deviation is not large, can be explained by random components of the estimates, and provides a nearly ideal balance between the biasedness and randomness.

The Lepski's approach defines the pointwise adaptive scale $h^+(x)$ according to the rule

$$h^+(x) = \max_i \{h_i : |\hat{y}_{h_j}(x) - \hat{y}_{h_i}(x)| \le \Gamma(h_j, h_i, x), \forall h_j < h_i,\ i = 1, \ldots, J\}, \tag{6.43}$$

where $\Gamma(h_j, h_i, x)$ is a given threshold.

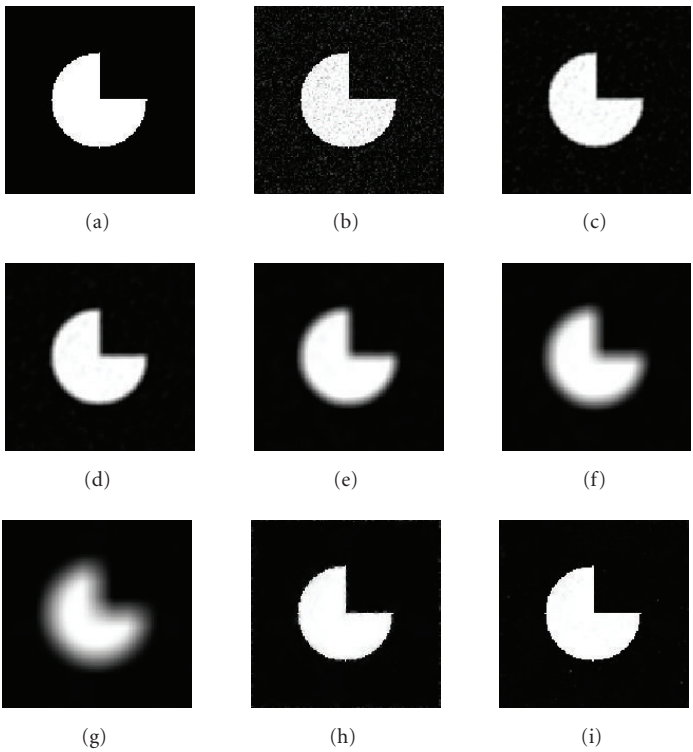


FIGURE 6.8. Gaussian filtering "cheese" noisy image: (a) true image; (b) $h_1 = h_2 = 1$ (noisy image), RMSE = 25.71; (c) $h_1 = h_2 = 2$, RMSE = 15.77; (d) $h_1 = h_2 = 3$, RMSE = 18.31; (e) $h_1 = h_2 = 5$, RMSE = 23.71; (f) $h_1 = h_2 = 7$, RMSE = 28.42; (g) $h_1 = h_2 = 11$, RMSE = 36.01; (h) adaptive varying window, RMSE = 10.14; (i) directional adaptive varying window, RMSE = 3.71.

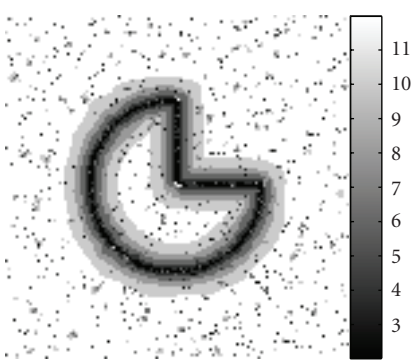


FIGURE 6.9. ICI adaptive window sizes: symmetric Gaussian window kernel, $m = [0, 0]$; "cheese" image. Small window sizes (black) are concentrated near the discontinuity curve. It allows a good sharpness of edges in the filtered image. Far from this curve, the window size is mainly large (white), what enables a strong noise suppression in this areas. Black isolated points are a result of noise effects. They appear in pixels where the ICI erroneously takes small window sizes.

The procedure (6.43) is looking for a largest scale h_i in order to obtain the maximum smoothing effect for the random errors. However, a large h_i can result in a significant bias error.

The estimates $\hat{y}_{h_j}(x)$ with the scale $h_j < h_i$ are compared with the estimate $\hat{y}_{h_i}(x)$ of the scale h_i. If all the differences $|\hat{y}_{h_j}(x) - \hat{y}_{h_i}(x)|$, $h_j < h_i$, can be explained by the random errors, then the bias is not large and larger h_i can be selected.

The adaptive scale h^+ is defined as a maximum h_i such that all estimates $\hat{y}_{h_j}(x)$ with $h_j < h_i$ are not too different from $\hat{y}_{h_i}(x)$.

The threshold $\Gamma(h_j, h_i, x)$ is a key element of the algorithm as it says when the difference between the estimates is large or small.

The rule (6.43) enables a multiple pairwise statistical test on the significance of the systematic error in the differences $\hat{y}_{h_j}(x) - \hat{y}_{h_i}(x)$.

There are a few versions of this concept which are different mainly by estimates $\hat{y}_h(x)$, sets of scales H, and hypothesis testing rules. The threshold $\Gamma(h_j, h_i, x)$ in the right-hand side of the inequality in (6.43) is an important parameter of this hypothesis testing.

Theoretical results given in the papers cited above show that this sort of algorithms has quite exciting properties in terms of the adaptivity over wide classes of signals, minimax estimation, convergence rates.

These adaptive algorithms work without special estimation of change points and singularities. The main intention is to estimate the signal with the varying adaptive scale selection. Near the change points, small scale values are selected automatically and in this way the accuracy of the local estimation is ensured.

We are concentrated on one particular form of this class of the methods developed in [13] independently from the main stream of works in this direction. The term "*intersection of confidence intervals*" (ICI) for this method appeared in [22].

It is shown in a number of publications that the ICI-based algorithms are quite practical and efficient for adaptive signal and image processing with application to different problems: median filtering [27], probability density estimation [25], time-frequency analysis [6, 23], beamforming and direction of arrival estimation [24]. A successful application of the ICI rule to image denoising has been reported in [26, 28, 30]. A number of interesting modifications of the ICI rule for regression problems with Gaussian and non-Gaussian noises is developed in [48].

6.3.2. Intersection of confidence interval rule

Let $y(x)$ and $\hat{y}_h(x)$ be the signal and its estimate, with the estimate and the standard deviation $\sigma_{\hat{y}_h}(x, h)$ defined according to the formulas (6.28) and (6.40), respectively. Then the confidence interval of this estimate is defined as follows:

$$\mathcal{D}_j = \{\hat{y}_{h_j}(x) - \Gamma \cdot \sigma_{\hat{y}_h}(x, h_j),\ \hat{y}_{h_j}(x) + \Gamma \cdot \sigma_{\hat{y}_h}(x, h_j)\}, \tag{6.44}$$

where $\Gamma > 0$ is a parameter defining the width of the confidence interval.

If the estimate $\hat{y}_{h_j}(x)$ is unbiased, the parameter Γ can be interpreted and calculated as follows. Assume that the noise ε_s is subject to the Gaussian probability density $\mathcal{N}(0, \sigma^2)$. Let $\chi_{1-\beta/2}$ be $(1-\beta/2)$th quantile of the standard Gaussian distribution $\mathcal{N}(0,1)$. Then $\Gamma = \chi_{1-\beta/2}$ and the true $y(x)$ belongs to the confidence interval $\mathcal{D}_j$ with the probability $p = 1-\beta$.

Thus, for the unbiased estimate the parameter Γ is the quantile and its value guarantees some probability that the true $y(x)$ is indeed inside this interval.

In the ICI rule, the confidence intervals are used for both biased and unbiased estimates and in this development Γ is a parameter of the algorithm with its meaning beyond the standard quantile of the Gaussian distribution.

Now we describe the ICI rule. Let $\mathcal{D}_j$ be a sequence of the confidence intervals corresponding to h_j defined in H. Consider sequential intersections of these confidence intervals starting from h_1. Thus we consider the intersection of two intervals $\mathcal{D}_1$ and $\mathcal{D}_2$, then go to the intersection of three intervals $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ and continue up to the moment when the intersection is empty. If it happens for the intervals $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_{i+1}$, the window size $h^+ = h_i$ is selected as the adaptive one.

This rule can be formulated in the following way [30]. *Consider the intersection of the intervals $\mathcal{D}_j$, $1 \le j \le i$, with increasing i, and let i_+ be the largest of those i for which the intervals $\mathcal{D}_j$, $1 \le j \le i$, have a point in common. This i_+ defines the adaptive scale and the adaptive* LPA *estimate as follows*:

$$\hat{y}^+(x) = \hat{y}_{h^+(x)}(x), \quad h^+(x) = h_{i_+}. \tag{6.45}$$

The ICI rule is illustrated in Figure 6.10. Let us use this figure in order to formulate the idea of the ICI rule once more. We start from the smallest window size with $h_1 = 1$ assuming that there is no smoothing at all in this estimate. Then the estimate coincides with the observation and the true $y(x)$ definitely belongs to the confidence interval $\mathcal{D}_1$ with some probability depending on Γ. Considering the next interval $\mathcal{D}_2$, we do not know if the true $y(x)$ belongs to this interval as the estimate with $h_2 > h_1$ may be already biased. However, the intersection of $\mathcal{D}_2$ with $\mathcal{D}_1$ gives a chance that the true $y(x)$ belongs also to $\mathcal{D}_2$ and probably is located in the intersection of these intervals. We continue this analysis while there is an intersection because if there is no intersection there is no chance that the true $y(x)$ belongs also to the last considered confidence interval. When we arrive at the stage that there is no intersection, we take the window size corresponding to the last accepted confidence interval as the adaptive window size of the estimate.

These calculations are performed for each x and give the varying adaptive $h^+(x)$ for each pixel of the image.

The ICI rule is presented here as a sort of reasonable heuristic procedures. In reality, it is derived from the accuracy analysis of the estimates with the smoothing parameter h. The adaptive window size given by the ICI rule is close to the optimal values minimizing the pointwise mean squared error [30].

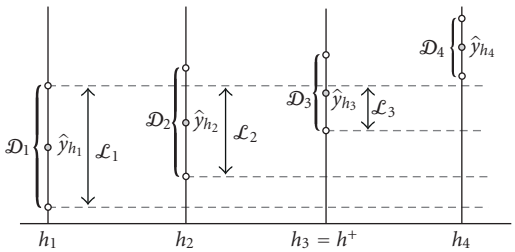


FIGURE 6.10. The idea of the ICI rule. We start from the smallest window size $h_1 = 1$ which guarantees that the true $y(x)$ with some probability belongs to the confidence interval $\mathcal{D}_1$. There is an intersection (nonempty common part) of the intervals $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ and there is no an intersection of the interval $\mathcal{D}_4$ with this common part of the intervals $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$. Therefore, the window size h_3 is selected as the adaptive window size.

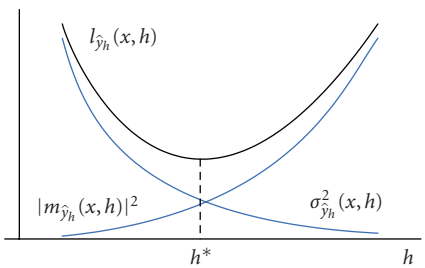


FIGURE 6.11. The squared bias $|m_{\hat{y}h}(x,h)|^2$ and the variance $\sigma^2_{\hat{y}h}(x,h)$ are illustrating the bias-variance trade-off. The "ideal" balance is achieved at $h = h^*$ minimizing the mean squared loss function $l_{\hat{y}h}(x,h)$.

Figure 6.11 illustrates a typical scenario when the ICI rule is applicable as well as the meaning of the adaptive values obtained by using it. In this figure we can see the estimation bias and the variance as functions of the window size h. Typically the bias and the variance are growing and decreasing functions of h, respectively. Then, the mean squared error has a minimum with respect to h. This optimal value provides a special bias-variance balance minimizing the mean squared criterion.

The remarkable property of the ICI is that this rule is able to give the adaptive window size close to this optimal value.

The structure of the adaptive algorithm using the ICI rule is shown in Figure 6.12. The kernels g_h are used in order to calculate a number of estimates and the ICI rule selects the best estimate for each pixel. It follows from Figure 6.12 that the algorithm can be interpreted as a multichannel filter bank with switching between channels.

More details concerning the derivation, justification of the ICI rule, and the role of the parameter Γ can be found in [30, Chapter 6] where the accuracy analysis of this rule is given.

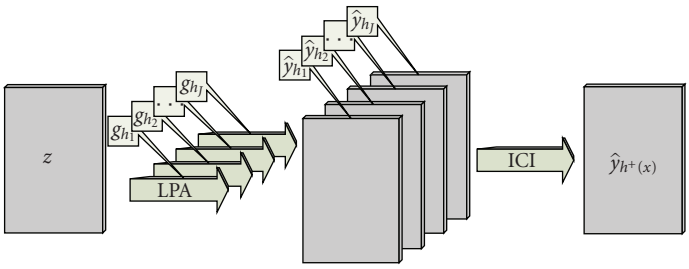


FIGURE 6.12. A layout of the adaptive scale LPA-ICI algorithm.

6.3.2.1. Intersection of confidence interval rule for differentiation

The ICI algorithm is universally applicable for estimation of both the signal and the signal derivative. The only difference is in the used estimates and in formulas for variance.

The inputs of the ICI rule are the signal estimates and their standard deviations given for all $h \in H$. The output is the corresponding adaptive window size and the adaptive estimate.

In order to use the ICI rule for differentiation, we need to use as inputs the estimates of the derivatives and their standard deviations calculated for all $h \in H$. Then the output of the ICI rule is the adaptive window size and the adaptive estimate of the corresponding derivative.

When actually this idea works? Figure 6.11 gives the explicit answer to this question. If the bias of the derivative estimate is an increasing function of h and the variance is a decreasing function of h, then the ICI gives the window size close to the optimal ones minimizing the mean squared error of this estimate. The assumption that the variance is decreasing function of h holds always for the LPA estimates, and thus the main point is whether the bias is really a growing function of h.

6.3.3. Implementation of ICI rule in LASIP

The ICI rule is implemented in *LASIP* in the program *function_ICI.p*. This program is used in the following format: [YICI, h_opt, std_opt] = function_ICI(yh, stdh, gammaICI).

Input variables: "*yh*" is an array $(n_1 \times n_2 \times J)$ of the image estimates $y_h(x)$ calculated for all $h \in H$, "*stdh*" is a vector $(J \times 1)$ of the standard deviations of the estimates calculated according to (6.40) for all $h \in H$, and "*gammaICI*" is a value of the parameter Γ used in the ICI rule.

The program returns the following outputs: "*YICI*" is an adaptive estimate $(n_1 \times n_2)$ of the image, "*h_opt*" is an array of the image size $(n_1 \times n_2)$ with the values of the window sizes used for each pixel of the image, "*std_opt*" is an array of the size $(n_1 \times n_2)$ with the variances of the adaptive estimates for each pixel.

This implementation is fast because all calculations are produced as a loop over the J scales. The complexity and time of calculations depend mainly on the number of the window sizes J and values of the window sizes h_k. Larger h_k results in larger calculation time.

6.4. Anisotropic directional filters

6.4.1. Directional windows and kernels

Intuitively, an anisotropy means that an image intensity $y(x)$ has a different behavior in different directions in a neighborhood of the pixel x. Points, lines, edges, textures are typical elements in images. They are locally defined by position, orientation, size or scale. Often being of small size, these specific features encode a great portion of information contained in images. All these image elements define anisotropic areas in the image intensity y.

In order to deal with this sort of anisotropy, the kernel estimates based on the symmetric window functions are not efficient. Steerable filters [11] and directional wavelets (rigelets, curvelets) [45] are only examples of tools developed in signal and image processing to deal with data anisotropy.

Our approach to anisotropy is based on the use of starshaped size/shape adaptive neighborhoods built for each image pixel.

Figure 6.13 illustrates this concept and shows sequentially: a local best ideal estimation neighborhood U_x^* (Figure 6.13(a)), a sectorial segmentation of the unit disc (Figure 6.13(b)), and the sectorial approximation of U_x^* using the parameter $h^+ = h^+(\theta_t)$ defining the length of the corresponding sectors in the directions θ_t, $t = 1,\ldots,K$, (Figure 6.13(c)). Varying size sectors enable one to get a good approximation (Figure 6.13(d)) of any neighborhood of the point x provided that it is a star-shaped body.

This approximation is defined by K sector-size parameters h_t^*. Adaptation using all of these parameters simultaneously is difficult encountering some technical and principal points. The technical one is obvious as it assumes that vector (of dimension K) optimization of the estimates should be done and that it always requires intensive computing. This vector optimization can be a difficult multimode problem.

The adaptation implemented by the ICI rule assumes that the estimates can be ordered according to their variances. The ICI rule selects the estimate with the smallest variance (largest area of neighborhood) provided that the bias is of the order of the variance. In the star-shaped neighborhood there are a lot of sectors of the equal area and with equal variance of the estimates. Then, ordering the estimates according to their variance and selection of the estimates according to these variances is not able to indicate a unique neighborhood optimizing the estimation accuracy. It illustrates the principal difficulties of multivariable star-shaped neighborhood optimization.

To be practical, we use a sequential procedure with independent selection of the sectorial length for each of K directions.

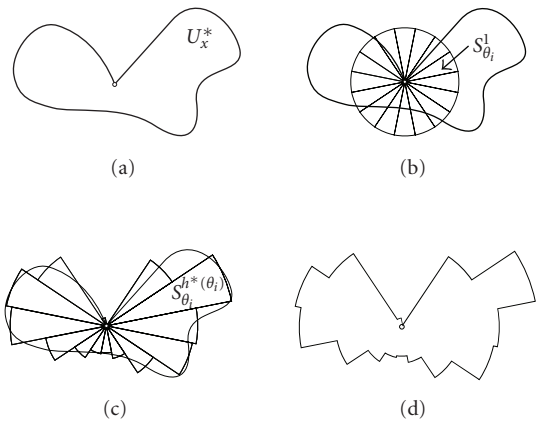


FIGURE 6.13. An optimal neighborhood of the estimation point x: (a) the best estimation set U^*, (b) the unit ball segmentation, (c) the sectorial approximation of U^*, (d) a set approximating U^*.

For each of the directions, the sector size is characterized by a single parameter h_i and we use the ICI rule in order to find the adaptive h_t^+.

In this way for each pixel x we obtain K directional adaptive window sizes and K adaptive directional estimates.

The final estimates are produced from the directional ones using a special aggregation (fusing) procedure.

The directional kernels used for estimation in the sectors are obtained by the window function w which is designed in such a way that its support completely belongs to a particular sector.

The program *demo_CreateLPAKernels.m* discussed in Section 6.2.4 is used for calculation of the directional kernels $g_{h,\theta}(x)$. Here the angle θ denotes the direction of the estimate. The parameter *directional_resolution* of this program refers to the number of sectors K. With $K = 4$ and the parameter TYPE $= 11$, we obtain four directional kernels with supports equal to four quadrants of the standard Cartesian coordinate system. These kernels for one of the quadrants are shown in Figures 6.6 and 6.7 for smoothing and differentiation, respectively.

For high-resolution imaging, we use eight-sector estimates with $\theta = \theta_i$, $\theta_i = \pi(i-1)/4$, $i = 1, \ldots, 8$. In order to obtain kernels for these estimates, the sectorial support window is used in this program. The length and the width of the window are defined by the parameters h_1 and h_2. Assuming that $h_2 = 1$ for all h_1, linewise kernels are returned by the program *demo_CreateLPAKernels.m*.

The LPA order of the kernel is defined by the parameter $m = [m_1, m_2]$ giving the kernel orders along and across of the radial direction. If the size of the kernel is not sufficient to fit the desirable order of the LPA approximation, the zero order is selected automatically.

In particular, for $h_2 = 1$, the order m_2 is always reduced to $m_2 = 0$.

6.4.2. ICI rule for directional estimates

There are some extra features of the ICI rule for the directional estimates. The program *function_ICI.p* accepts the fourth argument in the format: [YICI,h_opt, std_opt] = function_ICI(yh,stdh,gammaICI, $(2\pi\times i)$/*directional_resolution*), where $i = 0,\ldots,K-1$.

In this form the program includes weighted median filtering of the ICI adaptive scales. This filtering is directional and is used in order to reduce a level of randomness in the corresponding adaptive scales.

This filtering results in the essential improvement of the algorithm performance.

For instance the results shown in Figure 6.8(i) are obtained by using the ICI rule with this median directional filtering which gives RMSE = 3.71. Applying the program without this filtering, we obtain larger RMSE = 4.87. It is necessary to note that with this filtering we use a smaller value of $\Gamma = 1.05$ versus $\Gamma = 1.5$ used in the algorithm with no filtering.

6.4.3. Estimate aggregation

The final estimate is calculated as the weighted mean of the directional adaptive estimates obtained for K directions:

$$\hat{y}(x) = \frac{\sum_{t=1}^{K} \lambda_t \hat{y}_{h^+(x),\theta_t}(x)}{\sum_{t=1}^{K} \lambda_t},$$
$$\lambda_t = \frac{\sigma_{\hat{\theta}_t}^{-2}(x)}{\sum_{t=1}^{K} \sigma_{\hat{\theta}_t}^{-2}(x)(x)}, \tag{6.46}$$

where $\sigma^2_{\hat{\theta}_t}(x)$ is a variance of the estimate $\hat{y}_{h^+(x),\theta_t}$.

This inverse variance weighting of the estimates is optimal assuming that the directional estimates $\hat{y}_{h^+(x),\theta_t}$ are unbiased and statistically independent. The estimate aggregation defined by (6.46) is one of the efficient ways to deal with the K estimates obtained for x.

6.5. Applications

6.5.1. Image denoising

In this section, we present simulation results demonstrating performance of the developed directional algorithms in image denoising.

We use the standard criteria for evaluation of the algorithm:

(1) root mean squared error (RMSE), RMSE $= \sqrt{(1/n)\sum_x (y(x) - \hat{y}(x))^2}$;

(2) signal-to-noise ratio (SNR) in dB, SNR $= 10\log_{10}(\sum_x |y(x)|^2 / \sum_x |y(x) - \hat{y}(x)|^2)$;

(3) improvement in SNR (ISNR) in dB, $\mathrm{ISNR} = 20\log_{10}(\hat{\sigma}_z/\mathrm{RMSE})$, where $\hat{\sigma}_z$ is an estimate of the observation standard deviation;
(4) peak signal-to-noise ratio (PSNR) in dB, $\mathrm{PSNR} = 20\log_{10}(\max_x |y(x)|/\mathrm{RMSE})$;
(5) mean absolute error (MAE), $\mathrm{MAE} = (1/n)\sum_x |y(x) - \hat{y}(x)|$;
(6) maximum absolute error (MAXDIF), $\mathrm{MAXDIF} = \max_x |y(x) - \hat{y}(x)|$.

It is well known that there is no one-to-one correspondence between the visual image quality and the above criteria. Different criteria show quite different optimal values for the design parameters of the algorithm. A visual inspection, which of course is quite subjective, continues to be the most important final performance criterion.

6.5.1.1. "Cheese" test image

A test signal is the 128×128 "cheese" binary image corrupted by an additive zero-mean Gaussian noise. The image intensity takes integer values $y = 0, 1$ with the noise standard deviation $\sigma = 0.1$.

For estimation, we use narrow nonoverlapping sectorial kernels of Type 10 with a uniform window function w, the order $m = [0, 0]$.

The kernel supports are defined by the two-dimensional scale $h = (h_1, h_2)$ with $h \in H$

$$H = \left\{ \binom{1}{1}, \binom{2}{1}, \binom{3}{1}, \binom{5}{1}, \binom{7}{2}, \binom{11}{3} \right\}. \tag{6.47}$$

The scales h_1 and h_2 form pairs and used for estimation in these pairs. The supports of the kernels are sectorial of the width equal $h_2 = 1$ for $h_1 = 1, 2, 3, 5$, and the width h_2 is increased to 2 and 3 pixels for $h_1 = 7$ and $h_1 = 11$, respectively.

The estimates are calculated as the sample means of observations included in the kernel supports. The ICI rule is used with the threshold $\Gamma = 1.05$.

The estimates and the adaptive scales $h_j^+(x)$ are found for eight directions:

$$\theta_j = (j-1)\frac{\pi}{4}, \quad j = 1, \ldots, 8. \tag{6.48}$$

These ICI adaptive directional estimates are fused in the final one using the multiwindow method (6.46).

The central panel of Figure 6.14 shows the true image. Eight surrounding panels show the ICI adaptive scales $h_j^+(x)$ for the corresponding eight directions θ_j. Thus, we can see the adaptive scales for directional estimates looking at the horizontal and vertical directions, that is, to *East*, *North*, *West*, and *South*, as well as to four diagonal directions *Nord-East*, *Nord-West*, *South-West*, *South-East*. White and black correspond to large and small window size values, respectively. The adaptive window sizes delineate the intensity function of the image very well. This delineation is obviously directional as the contours of the image are shadowed from the corresponding directions.

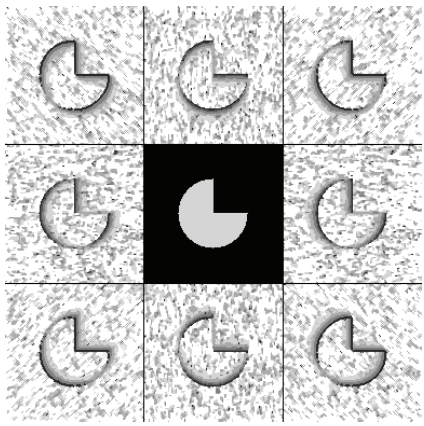


FIGURE 6.14. ICI adaptive directional scales $\widehat{h}_j^+(x)$, $\theta_j = (j-1)\pi/4$, $j = 1,\dots,8$, for "cheese" test image. The true image is shown in the central panel.

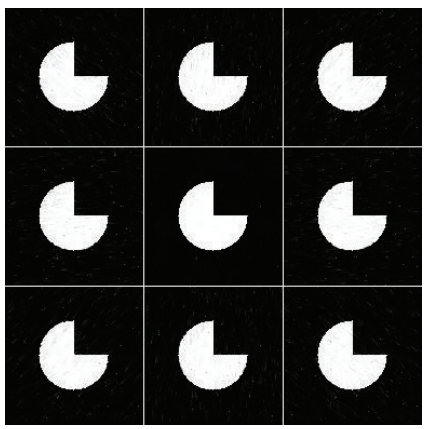


FIGURE 6.15. ICI adaptive directional estimates y_{h+,θ_j}, $\theta_j = (j-1)\pi/4$, $j = 1,\dots,8$, for "cheese" test image. The final fused estimate is shown in the central panel.

Figure 6.15 demonstrates the obtained estimates. The central panel shows the final fused estimate calculated from the directional ones according to the multi window estimate formula (6.46). The surrounding panels show the sectorial directional adaptive scale estimates $\widehat{y}_{\widehat{h}_j^+(x),\theta_j}(x)$, $j = 1,\dots,8$, corresponding to the adaptive scales given in Figure 6.14 for the corresponding directions.

The noise effects are clearly seen in the adaptive scales h_j^+ as spread solated black points. A directional nature of the adaptive estimates $\widehat{y}_{\widehat{h}_j^+(x),\theta_j}(x)$ is obvious since the corresponding directions are seen as a line-wise background of this imaging. The multiwindow fusing allows to delete and smooth these directional line effects and obtain a good quality final estimate.

Table 6.1. "Cheese" image: criteria values for the eight directional and final multiwindow estimates.

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	*Final Est*
ISNR, dB	7.22	6.91	7.11	7.50	7.20	6.92	7.17	7.22	16.80
SNR, dB	18.87	18.56	18.77	19.16	18.85	18.57	18.82	18.88	28.45
PSNR, dB	27.14	26.83	27.04	27.43	27.13	23.41	27.09	27.15	36.73
RMSE	11.19	11.60	11.33	10.83	11.22	11.58	11.26	11.18	3.71
MAE	6.81	6.77	6.95	6.65	6.72	6.82	6.86	6.73	2.40
MAXDIF	178.8	172.1	187.9	143.7	195.8	154.1	180.5	175.9	65.31

Overall, the multiwindow estimation allows to reveal and preserve sharp edges of this binary image and at the same time efficiently suppress the noise.

The numerical results are given for the considered directional estimates in Table 6.1. They show the criteria values for the all eight directional and final estimates.

The criterion values for the final estimate compared with the directional ones show a strong improvement in the fused estimate. In particular, we have for ISNR the values about 7 dB for the sectorial estimates, while for the fused estimate ISNR $\simeq$ 16 dB. The used adaptive weight fusing is an efficient tool for the accuracy improvement. Visually, the improvement effects are clear from the comparison of the directional and final estimates in Figure 6.15. As it is seen from Table 6.1, the fusing works very well for all criteria. The criteria RMSE, MAE, and MAXDIF are recalculated in this table for the image intensity values $0, 255$.

6.5.1.2. Real-life images

For texture images, we use the developed algorithm with narrower line-wise kernels from the set defined by the parameters

$$H = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 5 \\ 1 \end{pmatrix}, \begin{pmatrix} 7 \\ 1 \end{pmatrix}, \begin{pmatrix} 11 \\ 1 \end{pmatrix} \right\}. \tag{6.49}$$

The algorithm demonstrates a quite good performance with criterion values presented in Table 6.2. In this table, for the binary images "cheese" and "testpat1" the wider window sizes (6.47) are used. It is assumed that the texture image intensity is scaled to the interval $[0, 1]$. In the table numerical results, the criteria RMSE, MAE, and MAXDIF are recalculated for the image intensity belonging to the interval $[0, 255]$.

In successive first three columns in Figures 6.16 and 6.17, true image, noisy image, and the adaptive estimates are shown. The last column shows the adaptive window sizes obtained for the horizontal direction.

Table 6.2. Accuracy criterion for LPA-ICI adaptive imaging.

Test image	ISNR dB	SNR dB	PSNR dB	RMSE	MAE
Cheese	15.86	27.49	35.12	4.47/4.14	2.79
Lena	9.41	24.08	29.42	8.62	6.02
Cameraman	8.04	22.53	28.01	10.13	6.47
Peppers	9.46	24.72	29.47	8.57	6.12
Boats	7.81	22.47	27.82	10.36	7.42
Testpat1	7.53	25.85	27.51	10.74	6.07

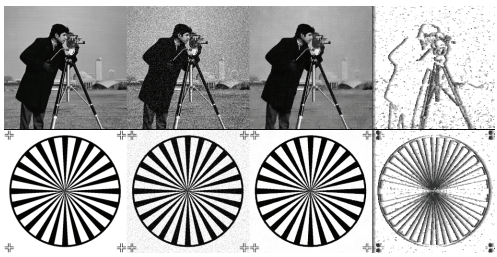


Figure 6.16. 256×256 test images "cameraman" and "testpat1." The successive columns show images: true, noisy, estimate, and distribution of the adaptive window sizes for the horizontal estimates.

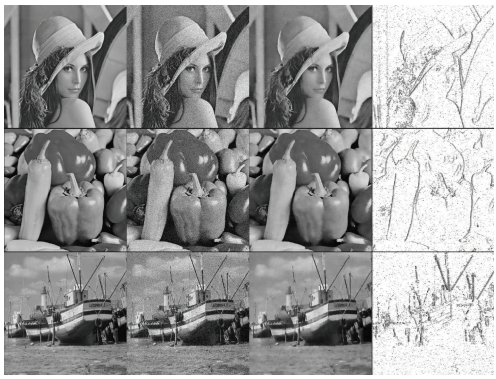


Figure 6.17. 512×512 test images "lena," "peppers," and "boats." The successive columns show images: true, noisy, estimate, and distribution of the adaptive window sizes for the horizontal estimates.

The program *function_shape_explorer.m* returns the adaptive neighborhoods obtained by the ICI rule. Figure 6.18 illustrates the work of this program for "cameraman" test image and shows the adaptive neighborhoods used for different pixels and their eight-directional sizes.

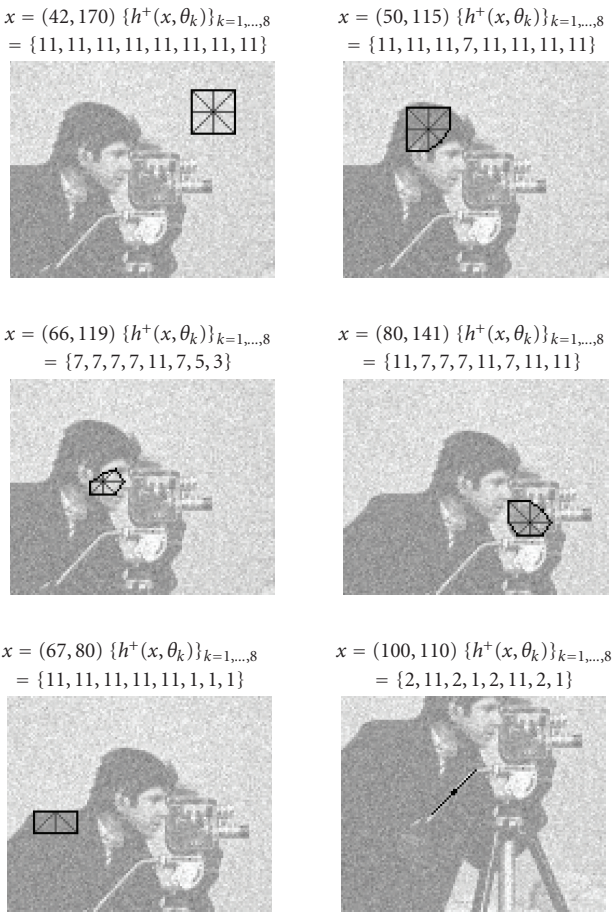


FIGURE 6.18. Adaptive neighborhoods for different pixels of "cameraman" test image.

6.5.2. Differentiation

In a number of applications, one is interested not in a signal itself but rather in a signal's derivative.

The LPA can be used for design of the linear differentiating directional kernels (6.33)-(6.34) with the estimates

$$\hat{y}_{h,\theta}^{(r_1,r_2)}(x) = \left(g_{h,\theta}^{(r_1,r_2)} \circledast z\right)(x), \tag{6.50}$$

where $g_{h,\theta}^{(r_1,r_2)}$ are the directional kernels for the derivative calculated in the direction θ.

This sort of derivative is a generalization of the directional finite differences. The generalization concerns a possible use of higher-order polynomials in the LPA and an arbitrary shape of the window-support w.

Directional estimates of the signal $\hat{y}_{h,\theta_i}(x)$ can be different for different directions. However, they always estimate the same variable $y(x)$. In this way, all of these estimates have the same meaning and can be naturally fused in a single estimate, say as the weighted mean in (6.46).

A situation is different for differentiation, as for each sector θ, we estimate a directional derivative and these derivatives have different meaning for each direction. Thus, a fusing similar to the one used for the signal estimation has no sense for the derivatives. Something in common should be found in the directional derivatives in order to use them jointly.

In what follows, we show how the directional derivatives can be used for precise estimation of the vector gradient. The main idea is that all of these directional derivatives can be treated as directional projections of the unique vector gradient.

Let us start from the coordinate system used for the directional estimation. Figure 6.19 shows a coordinate system rotated by an angle θ. A link between the basic and "*rotated*" Cartesian coordinate systems is defined by the geometrical transform

$$\begin{aligned} u_1 &= x_1 \cos\theta + x_2 \sin\theta, \\ u_2 &= -x_1 \sin\theta + x_2 \cos\theta, \end{aligned} \tag{6.51}$$

where (u_1, u_2) are new rotated coordinates depending on x_1, x_2 and θ is a fixed angle.

The inverse of the transform (6.51) is

$$\begin{aligned} x_1 &= u_1 \cos\theta - u_2 \sin\theta, \\ x_2 &= u_1 \sin\theta + u_2 \cos\theta. \end{aligned} \tag{6.52}$$

In the matrix notation we have for (6.51)-(6.52)

$$\begin{aligned} &u = U(\theta)x, \qquad x = U^T(\theta)u, \\ &U(\theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}. \end{aligned} \tag{6.53}$$

The kernels $g_{h,\theta}^{(r_1,r_2)}$ for these derivatives are calculated by the program *demo_CreateLPAKernels.m* already discussed in Sections 6.2.4 and 6.4.

Assuming that the signal y is differentiable at the point x with a vector gradient $\nabla \mathbf{y} = (\partial_{x_1} y, \partial_{x_2} y)^T$, the link between the vector gradient and the directional derivative has the following well-known form:

$$\partial_\theta y = \partial_{x_1} y \cdot \partial_{u_1} x_1 + \partial_{x_2} y \cdot \partial_{u_1} x_2 = \partial_{x_1} y \cdot \cos\theta + \partial_{x_2} y \cdot \sin\theta. \tag{6.54}$$

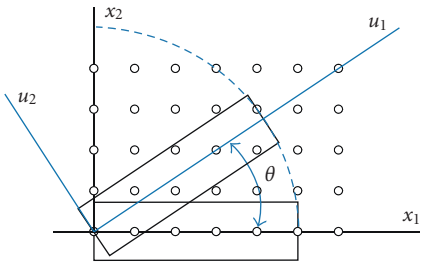


FIGURE 6.19. A rotation of the rectangular support shows that the number of pixels as well as their location inside the support depend on the rotation angle θ. For the LPA design of the kernels, new variables u_1 and u_2 are used as arguments for calculation of the window function and the polynomials.

The estimates of the first-order derivatives $\partial_{x_1} y$ and $\partial_{x_2} y$ can be calculated as the directional estimates $\hat{y}_{h,0}^{(1,0)}(x)$ and $\hat{y}_{h,\pi/2}^{(1,0)}(x)$, respectively, directed to $\theta = 0$ and $\theta = \pi/2$.

The corresponding first-order directional derivative $\partial_\theta y$ can be calculated according to (6.54) as

$$\partial_\theta \hat{y} = \hat{y}_{h,0}^{(1,0)} \cdot \cos\theta + \hat{y}_{h,\pi/2}^{(1,0)} \cdot \sin\theta. \tag{6.55}$$

However, this first-order directional derivative can be calculated also as $\hat{y}_{h,\theta}^{(1,0)}$ using the kernel $g_{h,\theta}^{(1,0)}$.

What is the difference between the two estimates $\partial_\theta \hat{y}$ and $\hat{y}_{h,\theta}^{(1,0)}(x)$? If the function y is continuous differentiable and a neighborhood used for estimation is small, then obviously these two estimates are equivalent.

However, when a signal can be nondifferentiable and defined only on a discrete grid, these estimates present different information about the local behavior of the function. The directional estimate $\hat{y}_{h,\theta}^{(1,0)}$ is a sort of generalized finite-differences calculated for the direction θ while the estimate (6.55) is calculated from the similar finite-differences but only for horizontal and vertical directions.

It is obvious that for an arbitrary function given on the discrete grid this horizontal and vertical information can be quite misleading, say for the diagonal directions $\theta = \pi/4$ or $\theta = 3\pi/4$. This is why the estimates $\hat{y}_{h,\theta}^{(1,0)}(x)$ and $\partial_\theta \hat{y}$ calculated from (6.55) can be very different.

One of the applications of the multiple directional derivatives is to use them for calculation of the vector gradient. If the directional derivatives $\hat{y}_{h,\theta_i}^{(1,0)}$ are given, the following equation links them with the vector gradient:

$$\hat{y}_{h,\theta_i}^{(1,0)} = \partial_{x_1} y \cdot \cos\theta_i + \partial_{x_2} y \cdot \sin\theta_i. \tag{6.56}$$

The least-square method can be used to find $\partial_{x_1} y$ and $\partial_{x_2} y$ from the multidirectional derivative estimates. Minimizing

$$J = \sum_{i=1}^{K} \left[\hat{y}^{(1,0)}_{h,\theta_i} - (\partial_{x_1} y \cdot \cos\theta_i + \partial_{x_2} y \cdot \sin\theta_i) \right]^2, \tag{6.57}$$

we find

$$\hat{\boldsymbol{\nabla}}\mathbf{y} = (B^T B)^{-1} B^T \hat{\mathbf{y}}^{(1,0)}_h, \tag{6.58}$$

where $\hat{\mathbf{y}}^{(1,0)}_h = (\hat{y}^{(1,0)}_{h,\theta_1}, \ldots, \hat{y}^{(1,0)}_{h,\theta_K})^T$ is a vector of the directional derivative and $B = (B_i)_{K\times 2}$, $B_i = (\cos\theta_i, \sin\theta_i)$.

It is obvious that this estimate of the gradient is different from what can be obtained just using the corresponding horizontal and vertical derivative estimates $\hat{y}^{(1,0)}_{h,0}$ and $\hat{y}^{(1,0)}_{h,\pi/2}$.

Similarly to the smoothing kernels, the differentiating estimates $\hat{y}^{(1,0)}_{h,\theta_i}$ have the window size parameter h defining the support of the estimate and the weights of the observations used for estimation. The ICI rule is applicable for the adaptive selection of h in these directional estimates.

Denote the adaptive scale directional derivative estimates as $\hat{y}^{(1,0)}_{h^+(x),\theta_i}$.

It is natural for noisy data to replace the criterion (6.57) by the weighted least-square

$$J = \sum_{i=1}^{K} \frac{1}{\sigma^2_{\theta_i}} \left[\hat{y}^{(1,0)}_{h^+(x),\theta_i} - (\partial_{x_1} y \cdot \cos\theta_i + \partial_{x_2} y \cdot \sin\theta_i) \right]^2, \tag{6.59}$$

where the weights $1/\sigma^2_{\theta_i}$ are the inverse variances of the adaptive estimates $\hat{y}^{(1,0)}_{h^+(x),\theta_i}$.

In the vector-matrix notation J can be rewritten as

$$J = \left(\hat{\mathbf{y}}^{(1,0)}_h - B\boldsymbol{\nabla}\mathbf{y}\right)^T \Lambda \left(\hat{\mathbf{y}}^{(1,0)}_h - B\boldsymbol{\nabla}\mathbf{y}\right), \tag{6.60}$$

where $\hat{\mathbf{y}}^{(1,0)}_h = (\hat{y}^{(1,0)}_{h^+(x),\theta_1}, \ldots, \hat{y}^{(1,0)}_{h^+(x),\theta_K})^T$ is a vector of the estimates, $\Lambda = \mathrm{diag}\{1/\sigma^2_{\theta_1}, \ldots, 1/\sigma^2_{\theta_K}\}$ is a diagonal matrix, and $B = (B_i)_{K\times 2}$, $B_i = (\cos\theta_i, \sin\theta_i)$.

Minimization of (6.60) gives the estimate of the gradient as

$$\hat{\boldsymbol{\nabla}}\mathbf{y} = (B^T \Lambda B)^{-1} B^T \Lambda \hat{\mathbf{y}}^{(1,0)}_h. \tag{6.61}$$

Thus, starting from the adaptive scale scalar directional derivatives we obtain the estimate of the vector-gradient $\triangledown\mathbf{y}$.

Figures 6.20–6.23 illustrate the performance of the adaptive window size differentiator for the test image "peppers." In these experiments, the additive Gaussian noise has the standard deviation $\sigma = 0.05$. For differentiation it is a quite large level of the noise.

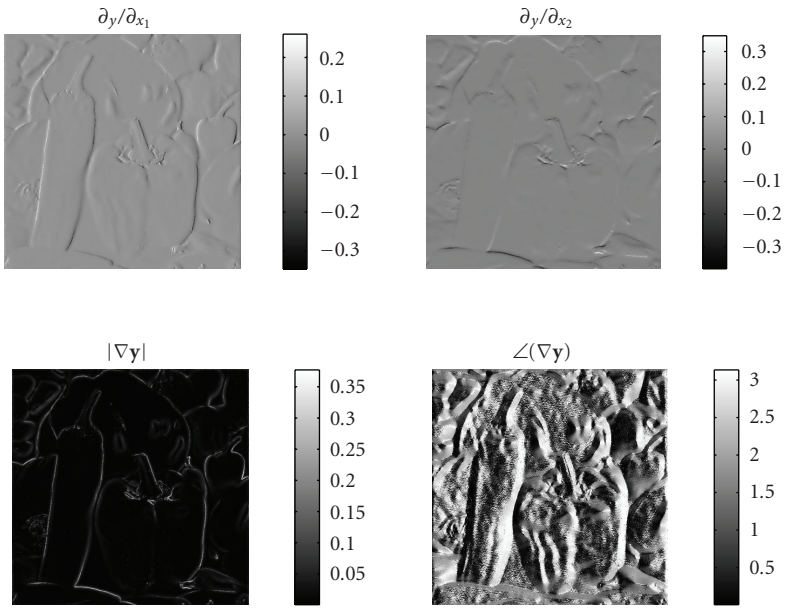


FIGURE 6.20. "Peppers" test image. The adaptive multidirectional estimates of the horizontal and vertical derivatives, the absolute value of the gradient and the gradient angle.

For differentiation we use the kernels $g_{h,\theta}^{(1,0)}$ of the linewise supports, the power $m = [1, 0]$, with the uniform window symmetric on variable x_2, the window TYPE = 10.

In Figure 6.20 we can see two (horizontal and vertical) of the eight adaptive multidirectional derivative estimates, the absolute value of the reconstructed gradient and gradient angle. For comparison, in Figures 6.21-6.22 we show these horizontal and vertical derivative estimates assuming that the differentiator window has a fixed size equal to $h = 3$ and $h = 11$, respectively. The absolute value and the angle of the vector gradient are also shown. It is clear that the estimates with the smallest possible window $h = 3$ give the sharp edges, but the derivative estimates are very noisy. The estimates with the larger window $h = 7$ are less noisy but the derivative edges become wider and definitely oversmoothed.

The adaptive window gradient estimates in Figure 6.20 demonstrate that a delicate job is done preserving the edges which are very sharp and smoothing the noisy from the estimates. Figure 6.23 shows the distribution of the adaptive window sizes for the directions $\theta = 0$ and $\theta = 45°$. In this figure, we can see also the noisy data ($\sigma = 0.05$) and the sum of the absolute values of the directional derivatives estimates calculated for all 8 directions with the corresponding adaptive window sizes. This last image gives very accurate edges of the image and can be used as very effective and noise-resistant edge detector.

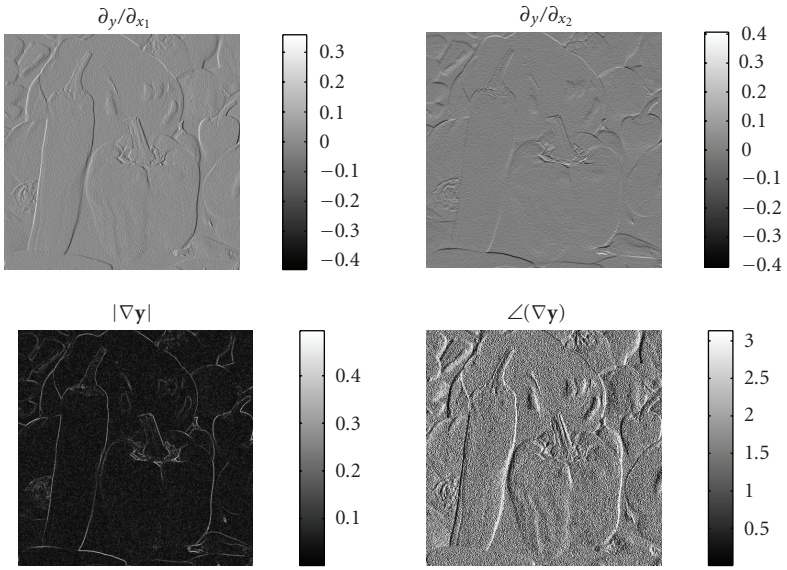


FIGURE 6.21. "Peppers" test image. The estimates of the horizontal and vertical derivatives calculated with the fixed small window size $h = 3$. The absolute value of the gradient and the gradient angle are also shown. The estimates are quite noisy.

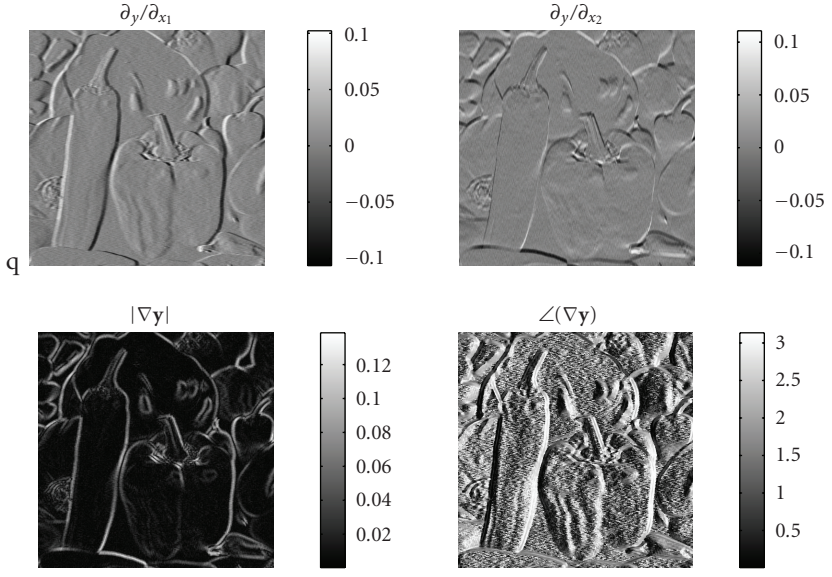


FIGURE 6.22. "Peppers" test image. The estimates of the horizontal and vertical derivatives calculated with the fixed large window size $h = 11$. The absolute value of the gradient and the gradient angle are also shown. The estimates are denoised and edges are oversmoothed.

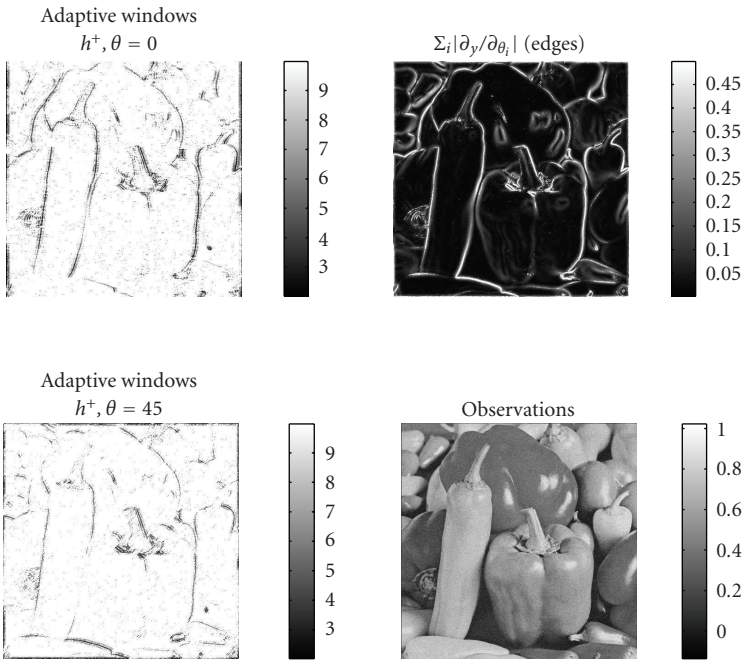


FIGURE 6.23. "Peppers" test image. The adaptive window sizes for estimates of the derivatives for $\theta = 0$ and $\theta = 45°$. The sum of the absolute values of the directional derivatives estimates gives good indications of the image edges. These derivative estimates are calculated for all 8 directions with the corresponding adaptive window sizes. Noisy observations.

6.5.3. Anisotropic gradient

Here we introduce a concept of the *directional anisotropic gradient* based on directional nonsymmetric LPA differentiation and the ICI adaptive scale selection.

Figure 6.24 illustrates the concept of the anisotropic gradient for two cases when the signal y is discontinuous and continuous. Figure 6.24(a) shows a piecewise linear discontinuous y composed from two linear fragments. It is assumed that x^0 is a discontinuity point. There are left and right neighborhoods of x^0 belonging to the corresponding argument areas $V(x^0_-)$ and $V(x^0_+)$, where the signal is regular with a unique derivative and a unique normal vector $\mathbf{n}_y$. Thus, at the point x^0 we have two neighborhoods $V(x^0_-)$ and $V(x^0_+)$ giving two different derivatives $\partial_x y$ and two different normal vectors.

Compare this situation with another one depicted in Figure 6.24(b), where again y is composed from two linear functions and the point x^0 belongs to intersection of these two lines. In this case, the function is continuous at x^0 and nevertheless there are two different normal vectors $\mathbf{n}_y(x^0)$ and two different left and right derivatives of y.

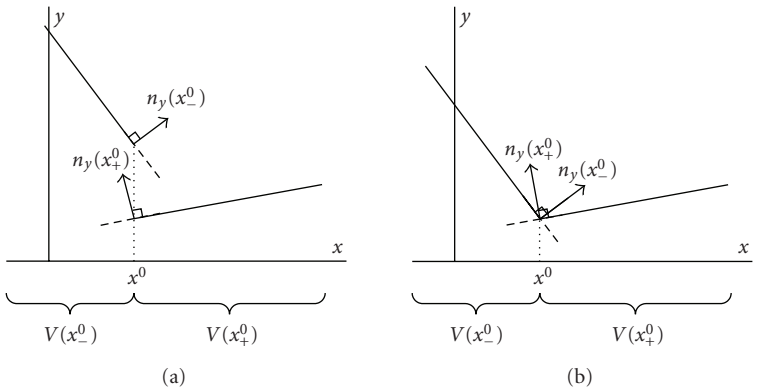


FIGURE 6.24. The concept of the anisotropic gradient for discontinuous (a) and continuous (b) functions. The estimator of the normal vector n_y should use the data for the argument value $x < x^0$ from the vicinity $V(x^0_-)$ and from the vicinity $V(x^0_+)$ if $x > x^0$.

In what follows, we use the ICI rule in order to identify and to build a largest neighborhood where the function allows a first-order polynomial model and where as a result the function is differentiable and the vector gradient can be found.

Similarly to what it is done in Section 6.4 for signal estimation, we introduce a sectorial partition of the unit ball with K nonoverlapping sectors having x as a common vertex. Let $g^{(1,0)}_{h,\theta}$ be nonsymmetric differentiating kernels defined on these sectors with the corresponding first-order derivative estimates $\widehat{y}^{(1,0)}_{h,\theta}$.

Assume that the support of the differentiating kernel $g^{(1,0)}_{h,\theta}$ is optimized by selection of the ideal scale $h^*(\theta)$ minimizing the corresponding least-square error.

The union of these optimal size supports defines the ideal neighborhood U^*_x for estimation of the gradient $\nabla \mathbf{y}(x)$. It is the largest neighborhood of x which can be used for gradient estimation. It is obvious that this differentiation ideal neighborhood is similar to the ideal neighborhood for the signal estimation in Figure 6.13.

Using the ICI for the adaptive window size derivative estimation allows to get some approximation of this ideal differentiation neighborhood. Assume that the support of the differentiating kernel $g^{(1,0)}_{h,\theta}$ is optimized by selection of the adaptive scale h^+ minimizing the corresponding least-square error. The union of these optimal size supports defines the adaptive neighborhood U^*_x for estimation of the gradient $\nabla \mathbf{y}(x)$. This star-shaped area is a largest adaptive neighborhood of x which can be used for gradient estimation because it is the largest area where the linear polynomial model fits the data. This idea is quite similar to the one used for the adaptive signal estimation as it is illustrated in Figure 6.13.

Let the directional adaptive derivative estimates $\widehat{y}^{(1,0)}_{h^+(x),\theta_i}$ be calculated. We fuse these estimates together for the gradient calculation using the procedure defined in (6.59)–(6.61).

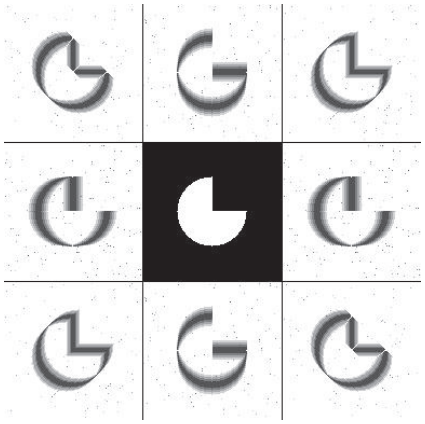


FIGURE 6.25. ICI adaptive directional scales $\widehat{h}_j^+(x)$, calculated for the directional derivative estimation, $\theta = (j-1)\pi/4,\ j = 1,\dots,8$, "cheese" test image. The true image is shown in the central panel. Standard mode of differentiation.

Two points make a principal difference with the adaptive gradient estimates considered in Section 6.5.2:

(i) *the derivative estimates are nonsymmetric*;

(ii) *the directions with the smallest adaptive* $h^+ = h_1$ *are eliminated from the fused estimate* (6.61).

These two conditions are used in order to obtain for estimation neighbourhoods which are not allowed to cross image edges.

When the ICI algorithm performs perfectly, the union of the sectorial supports of the adaptive estimates $\widehat{y}^{(1,0)}_{h^+(x),\theta_i}$ is a star-shaped set which allows the following interesting interpretation.

This star-shaped set is the largest neighborhood of x where the signal y is differentiable. In the more accurate terms, we can say that this star-shaped set is the largest area where the estimate of the vector gradient $\widehat{\nabla} y$ *fits signal observations provided that y is differentiable at the point x*.

Thus, the ICI adaptive estimate of the directional derivatives used in (6.61) gives two important complementary results. The first is the estimate of the gradient, and the second is the largest star-shaped neighborhood of x, where y is differentiable.

All these derivative calculations are implemented in the *LASIP* program *demo_AnisotropicGradient.m* allowing to calculate both the standard and anisotropic gradients.

6.5.3.1. Differentiation examples

Example 6.5.1 (gradient for "cheese"). Here we wish to illustrate the performance of differentiating operators in standard and anisotropic modes. As a test image, we

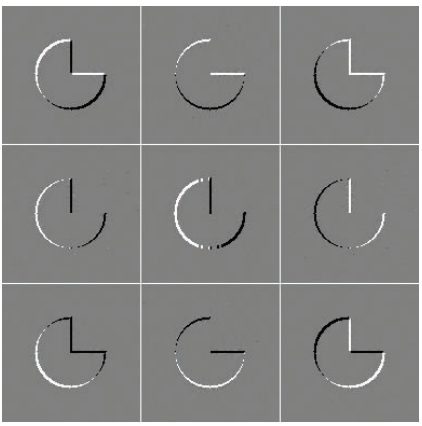


FIGURE 6.26. ICI adaptive directional estimate of the first-order derivatives, $\theta_j = (j-1)\pi/4$, $j = 1,\dots,8$, "cheese" test image. The adaptive scales $\widehat{h}_j^+(x)$ are shown in Figure 6.25. The obtained estimate of the horizontal derivative is shown in the central panel. Standard mode of differentiation.

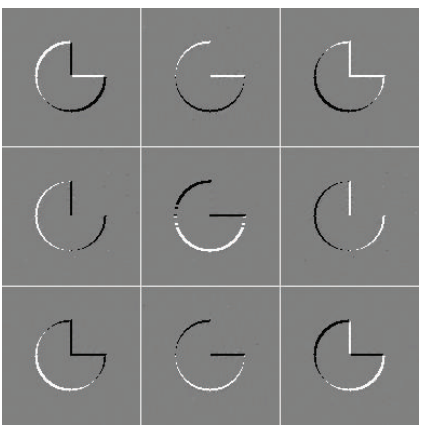


FIGURE 6.27. ICI adaptive directional estimate of the first-order derivatives, $\theta_j = (j-1)\pi/4$, $j = 1,\dots,8$, "cheese" test image. The adaptive scales $\widehat{h}_j^+(x)$ are shown in Figure 6.25. The obtained estimate of the vertical derivative is shown in the central panel. Standard mode of differentiation.

use "cheese" given with the zero-mean Gaussian noise of $\sigma = 0.05$. It is a quite high level of the noise for differentiation.

The kernels $g_{h,\theta}^{(1,0)}$ have the linewise support of the width equal to one and the length taking values from the set $h = [2, 4, 6, 8, 10]$. The parameter $\Gamma = 2.5$, the window function is Gaussian.

First, we apply the differentiating operators in the so-called *standard mode of differentiation*. This means that the kernel $g_{h,\theta}^{(1,0)}$ has a symmetric support on the variable x_1 and the gradient estimates are calculated according to (6.61).

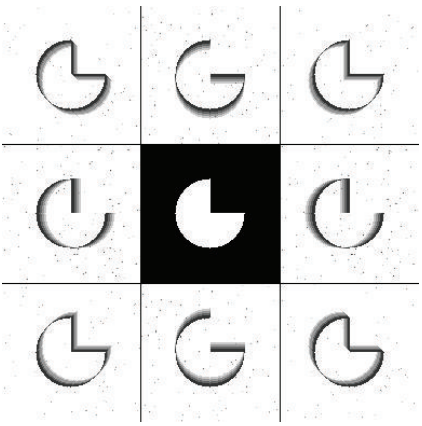


FIGURE 6.28. ICI adaptive anisotropic directional scales $\widehat{h}_j^+(x)$, calculated for the directional derivative estimation, $\theta_j = (j-1)\pi/4,\ j = 1,\dots,8$, "cheese" test image. The true image is shown in the central panel. Comparing shows that these adaptive scales obtained for the anisotropic estimates are sharper than those for the standard differentiation in Figure 6.25. Anisotropic mode of differentiation.

The adaptive window sizes selected for directional estimation of the derivatives $\partial_{x_1} y$ and $\partial_{x_2} y$ are shown in Figure 6.25 with the true "cheese" image in the central panel of this figure. Images of the corresponding directional derivative estimates are shown in Figures 6.26 and 6.27. The central panel of these two images show the estimates of the horizontal $\partial_{x_1} y$ and vertical derivatives $\partial_{x_2} y$. The directional adaptive estimates shown in eight panels around the central one are the same in both Figures 6.26 and 6.27. Note that the level of the noise of the final (horizontal and vertical) estimates is much lower than it is for the directional estimates. The line-wise background well seen in the directional estimates disappears in these final estimates.

Further, we go to the anisotropic derivative estimates (*anisotropic mode of differentiation*). In order to change the mode of our estimates, we calculate them with the nonsymmetric support. This means that $g_{h,\theta}^{(1,0)} = 0$ for $x_1 < 0$. The fusing of the directional derivatives into the horizontal and vertical estimates is carried out using (6.61), where the directional estimates $g_{h,\theta}^{(1,0)}$ with the minimum window size $h_1 = 2$ are dropped in this formula.

Figure 6.28 demonstrates the obtained adaptive window sizes. In comparison with the adaptive window sizes shown in Figure 6.25 for the standard differentiation, these adaptive window sizes for the anisotropic estimates are much sharper with narrower areas of small values of the window sizes near the edges of "cheese." The adaptive directional estimates with the estimate of the horizontal derivative in the central panel are shown in Figure 6.29.

While the directional estimates in the eight panels surrounding the central one look more or less similarly to the directional estimates in Figure 6.26, the final

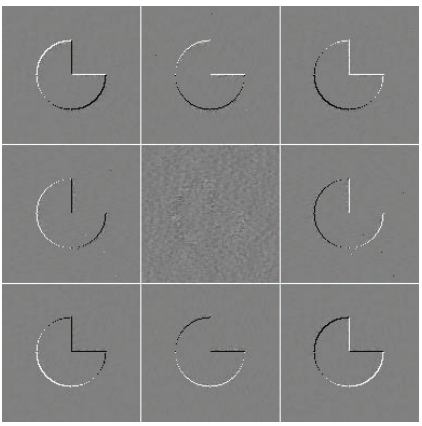


FIGURE 6.29. ICI adaptive anisotropic directional estimate of the first-order derivatives, $\theta_j = (j - 1)\pi/4$, $j = 1, \ldots, 8$, "cheese" test image. The adaptive scales $\widehat{h}_j^+(x)$ are shown in Figure 6.26. The obtained anisotropic estimate of the horizontal derivative is shown in the central panel. Even the values of these derivatives are shown multiplied by 10; they are not visible as their values are very close to 0. Note that the edges of the image are not visible in this derivative because of the nature of the anisotropic differentiation. Compare these images versus the central image in Figure 6.26 where these edges are clearly seen. Anisotropic mode of differentiation.

horizontal estimate shown in the central panel is completely different. Here we observe the effect which makes a great deal of difference between the standard and the anisotropic differentiation. The anisotropic differentiation means that the adaptive neighborhood selected for estimation is smooth. For "cheese" these neighborhood is always the area where the function takes a constant value with the derivatives equal to zero. The central panel of Figure 6.29, where the horizontal derivative estimate is about equal to zero for all pixels including the discontinuity edge, demonstrates a nearly perfect anisotropic performance of the algorithm.

The main feature of the anisotropic differentiation demonstrated in this example is that the algorithm does not cross the discontinuity curves and the edges of "cheese" test image are completely not seen in this derivative estimation.

6.6. Conclusion

The developed algorithms define a class of nonlinear spatially adaptive filters demonstrating a state-of-art performance and on many occasions visually and quantitatively outperforming the best existing methods (see, e.g., [7, 8, 28, 30]). In this chapter, we are restricted to the Gaussian denoising and differentiation only.

However, the LPA-ICI technique is quite universal and generalized for a wide scope of imaging problems including: Gaussian and non-Gaussian denoising, nonblind and blind multichannel deblurring, super-resolution imaging, poissonian signal processing, multiresolution imaging, and edge detection, color imaging, and

so forth. The algorithms implemented in Matlab for these applications as well as further references can be found on *LASIP* website http://www.cs.tut.fi/~lasip.

Acknowledgments

This work was supported by the Academy of Finland, Project no. 213462 (Finnish Centre of Excellence program (2006–2011)). In part, the work of the first author is supported by *Visiting Fellow* Grant from Nokia Foundation.

Bibliography

[1] R. Brown, *Smoothing, Forecasting and Prediction of Discrete Time Series*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1963.

[2] E. J. Candès and D L. Donoho, "Recovering edges in ill-posed inverse problems: optimality of curvelet frames," *The Annals of Statistics*, vol. 30, no. 3, pp. 784–842, 2002.

[3] W. S. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *Journal of the American Statistical Association*, vol. 74, no. 368, pp. 829–836, 1979.

[4] W. S. Cleveland and C. Loader, "Smoothing by local regression: principles and methods," in *Statistical Theory and Computational Aspects of Smoothing*, W. Hardel and M. Schimek, Eds., pp. 10–49, Physica, Heidelberg, Germany, 1996.

[5] D. L. Donoho, "Wedgelets: nearly minimax estimation of edges," *The Annals of Statistics*, vol. 27, no. 3, pp. 859–897, 1999.

[6] I. Djurović and L. Stanković, "Nonparametric algorithm for local frequency estimation of multidimensional signals," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 467–474, 2004.

[7] A. Foi, "Anisotropic Nonparametric Image Processing: Theory, Algorithms and Applications," Tesi di Dottorato, Dipartimento di Matematica, Politecnico di Milano, Italy, 2005.

[8] A. Foi, V. Katkovnik, K. Egiazarian, and J. Astola, "A novel anisotropic local polynomial estimator based on directional multiscale optimizations," in *Proceedings of 6th International Conference on Mathematics in Signal Processing*, pp. 79–82, Cirencester, UK, 2004.

[9] A. Foi, V. Katkovnik, K. Egiazarian, and J. Astola, "Inverse halftoning based on the anisotropic LPA - ICI deconvolution," in *Proceedings of International TICSP Workshop on Spectral Methods and Multirate Signal Processing (SMMSP '04)*, J. Astola, K. Egiazarian, and T. Saramäki, Eds., pp. 49–56, Vienna, Austria, September 2004.

[10] A. Foi, R. Bilcu, V. Katkovnik, and K. Egiazarian, "Anisotropic local approximations for pointwise adaptive signal-dependent noise removal," in *Proceedings of 13th European Signal Processing Conference (EUSIPCO '05)*, p. 4, Antalya, Turkey, September 2005.

[11] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 9, pp. 891–906, 1991.

[12] L. Ganesan and P. Bhattacharyya, "Edge detection in untextured and textured images—a common computational framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 27, no. 5, pp. 823–834, 1997.

[13] A. Goldenshluger and A. Nemirovski, "On spatially adaptive estimation of nonparametric regression," *Mathematical Methods of Statistics*, vol. 6, no. 2, pp. 135–170, 1997.

[14] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice-Hall, Upper Saddle River, NY, USA, 2nd edition, 2002.

[15] T. J. Hastie and C. Loader, "Local regression: automatic kernel carpentry," *Statistical Science*, vol. 8, no. 2, pp. 120–143, 1993, (with discussion).

[16] C. M. Hurvich, J. S. Simonoff, and C.-L. Tsai, "Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion," *Journal of the Royal Statistical Society. Series B*, vol. 60, no. 2, pp. 271–293, 1998.

[17] V. Katkovnik, "Problem of approximating functions of many variables," *Automation and Remote Control*, vol. 32, no. 2, part 2, pp. 336–341, 1971.

[18] V. Katkovnik, "Homogeneous integral averaging operators obtained by the method of least squares," *Automation and Remote Control*, vol. 32, no. 11, part 1, pp. 1767–1775, 1971.

[19] V. Katkovnik, *Linear Estimation and Stochastic Optimization Problems*, Nauka, Moscow, Russia, 1976.

[20] V. Katkovnik, "Linear and nonlinear methods of nonparametric regression analysis," *Soviet Journal of Automation and Information Sciences*, vol. 5, pp. 25–34, 1979.

[21] V. Katkovnik, *Nonparametric Identification and Smoothing of Data (Local Approximation Methods)*, Nauka, Moscow, Russia, 1985.

[22] V. Katkovnik, "A new method for varying adaptive bandwidth selection," *IEEE Transactions on Signal Processing*, vol. 47, no. 9, pp. 2567–2571, 1999.

[23] V. Katkovnik and L. Stanković, "Periodogram with varying and data-driven window length," *Signal Processing*, vol. 67, no. 3, pp. 345–358, 1998.

[24] A. B. Gershman, L. Stanković, and V. Katkovnik, "Sensor array signal tracking using a data-driven window approach," *Signal Processing*, vol. 80, no. 12, pp. 2507–2515, 2000.

[25] V. Katkovnik and I. Shmulevich, "Kernel density estimation with adaptive varying window size," *Pattern Recognition Letters*, vol. 23, no. 14, pp. 1641–1648, 2002.

[26] V. Katkovnik, K. Egiazarian, and J. Astola, "Adaptive window size image de-noising based on intersection of confidence intervals (ICI) rule," *Journal of Mathematical Imaging and Vision*, vol. 16, no. 3, pp. 223–235, 2002.

[27] V. Katkovnik, K. Egiazarian, and J. Astola, "Application of the ICI principle to window size adaptive median filtering," *Signal Processing*, vol. 83, no. 2, pp. 251–257, 2003.

[28] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola, "Directional varying scale approximations for anisotropic signal processing," in *Proceedings of 12th European Signal Processing Conference (EUSIPCO '04)*, pp. 101–104, Vienna, Austria, September 2004.

[29] V. Katkovnik, K. Egiazarian, and J. Astola, "A spatially adaptive nonparametric regression image deblurring," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1469–1478, 2005.

[30] V. Katkovnik, K. Egiazarian, and J. Astola, *Local Approximation Techniques in Signal and Image Processing*, SPIE Press, Bellingham, Wash, USA, 2006.

[31] O. V. Lepskii, "On one problem of adaptive estimation in Gaussian white noise," *Theory of Probability and Its Applications*, vol. 35, no. 3, pp. 454–466, 1990.

[32] O. V. Lepskii, "Asymptotically minimax adaptive estimation I: upper bounds. Optimally adaptive estimates," *Theory of Probability and Its Applications*, vol. 36, no. 4, pp. 682–697, 1991.

[33] O. V. Lepski and V. G. Spokoiny, "Optimal pointwise adaptive methods in nonparametric estimation," *The Annals of Statistics*, vol. 25, no. 6, pp. 2512–2546, 1997.

[34] O. V. Lepski, E. Mammen, and V. G. Spokoiny, "Optimal spatial adaptation to inhomogeneous smoothness: an approach based on kernel estimates with variable bandwidth selectors," *The Annals of Statistics*, vol. 25, no. 3, pp. 929–947, 1997.

[35] D. Mercurio and V. G. Spokoiny, "Estimation of time-dependent volatility via local change point analysis," WIAS, preprint 904, 2004.

[36] E. A. Nadaraya, "On estimating regression," *Theory of Probability and Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[37] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Statistics*, vol. 81, pp. 215–222, 1962.

[38] I. F. Petersen, "Statistical optimization by smoothing," *Engineering Cybernetics*, no. 2, pp. 36–44, 1969 (Russian).

[39] J. Polzehl and V. G. Spokoiny, "Adaptive weights smoothing with applications to image restoration," *Journal of the Royal Statistical Society. Series B*, vol. 62, no. 2, pp. 335–354, 2000.

[40] J. Polzehl and V. G. Spokoiny, "Propagation-separation approach for local likelihood estimation," *Probability Theory Related Fields*, vol. 135, no. 3, pp. 335–362, 2006.

[41] J. Polzehl and V. G. Spokoiny, "Image denoising: pointwise adaptive approach," *The Annals of Statistics*, vol. 31, no. 1, pp. 30–57, 2003.
[42] J. Polzehl and V. G. Spokoiny, "Varying coefficient regression modeling," WIAS, preprint 818, 2003.
[43] E. P. Simoncelli and H. Farid, "Steerable wedge filters for local orientation analysis," *IEEE Transactions on Image Processing*, vol. 5, no. 9, pp. 1377–1382, 1996.
[44] V. G. Spokoiny, "Estimation of a function with discontinuities via local polynomial fit with an adaptive window choice," *The Annals of Statistics*, vol. 26, no. 4, pp. 1356–1378, 1998.
[45] J.-L. Starck, E. J. Candès, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Transactions on Image Processing*, vol. 11, no. 6, pp. 670–684, 2002.
[46] C. J. Stone, "Consistent nonparametric regression (with discussion)," *The Annals of Statistics*, vol. 5, no. 4, pp. 595–645, 1977.
[47] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry*, vol. 36, pp. 1627–1639, 1964.
[48] L. Stanković, "Performance analysis of the adaptive algorithm for bias-to-variance trade-off," *IEEE Transactions on Signal Processing*, vol. 52, no. 5, pp. 1228–1234, 2004.
[49] J. S. Simonoff, *Smoothing Methods in Statistics*, Springer, New York, NY, USA, 1998.
[50] G. S. Watson, "Smooth regression analysis," *Sankhya. Series A*, vol. 26, pp. 359–372, 1964.

Vladimir Katkovnik: Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, Tampere 33101, Finland
Email: katkov@cs.tut.fi

Karen Egiazarian: Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, Tampere 33101, Finland
Email: karen@cs.tut.fi

Jaakko Astola: Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, Tampere 33101, Finland
Email: jta@cs.tut.fi

7 Image interpolation by optimized spline-based kernels

Atanas Gotchev, Karen Egiazarian, and Tapio Saramäki

In this chapter, we discuss the arbitrary scale image interpolation problem considered as a convolution-based operation. Use is made of the class of piecewise (spline-based) basis functions of minimal support constructed as combinations of uniform *B*-splines of different degrees that are very susceptible for optimization while being also very efficient in realization. Adjustable parameters in the combinations can be tuned by various optimization techniques. We favor a minimax optimization technique specified in Fourier domain and demonstrate its performance by comparing with other designs in terms of error kernel behavior and by interpolation experiments with real images.

7.1. Introduction

Reconstruction of a certain continuous function from given uniformly sampled data is a common problem in many image processing tasks such as image enlargement, rotation, and rescaling. The problem is broadly known as image interpolation.

CCD or CMOS arrays, the most used devices for recording digital images, create patterns of discrete pixels on rectangular grids. Many image processing applications, such as zooming, affine transforms, unwarping, require generating new pixels at coordinates different from the given grid. These new pixels can be generated by a two-stage procedure. First, a certain continuous function is reconstructed to fit the given uniformly sampled data and then it is resampled at the desired coordinates. The process is widely known as interpolation and, in order to be successful, the reconstruction (interpolation) functions have to be chosen properly. Some of the most commonly adopted interpolation models represent the continuous function under reconstruction as a discrete sum of weighted and shifted, adequately chosen, basis functions. This is the so-called linear or *convolution-based* interpolation. A more general framework considers the reconstruction functions as generators of shift-invariant spaces. This formalism allows making an elegant relation between the continuous function to be sampled or reconstructed and its

approximation (projection) on a shift-invariant space in terms of its discrete representation (samples). Moreover, for this formalism, the approximation theory provides tools for quatifying the error between the function and its approximation. Among reconstruction functions generating shift-invariant spaces, B-splines have attracted special attention due to their efficiency and excellent approximation properties. Furthermore, piecewise polynomial functions expressed as linear combinations of B-splines of different degrees have been optimized to achieve even better approximation properties for shortest possible function's support.

In this chapter, we address the problem of image interpolation by optimized spline-based kernels. In Section 7.2, we state the interpolation problem in signal and frequency domains. We then describe the most general distortions caused by nonideal interpolation and quantitative measures for evaluating the performance of a particular interpolator. Further, in Section 7.3, we demonstrate how to design and analyze piecewise polynomial basis functions of minimal support which can be used in interpolation and projection tasks. The design is based on linear combinations of uniform B-splines of different degrees. We compare the basis kernels designed following different optimization criteria in terms of error kernel behavior and by interpolation experiments with real images. All our notations are for the one-dimensional (1D) case. While it helps to clarify the ideas in a simpler way with higher dimensions easy to generalize, it is also practical as in many applications separable interpolators as tensor product of 1D functions are preferred for their computational efficiency.

7.2. Basics of sampling and interpolation

7.2.1. Sampling

Consider the most popular imaging system consisting of an optical system and digital image capturing device. The optical system (lens) focuses the scene on the imaging plane. Its performance depends, among other factors such as distance and focal length, on the finite aperture size. It determines the optical resolution limits. The optical transfer function (OTF), being the autocorrelation function of the aperture size [54], is also finite and hence cuts all spatial frequencies outside the region of its support.

The next is the sampling device, transforming the captured image into pixels. In most modern digital imaging systems it is a kind of imaging sensor based on the principle of charge-coupled devices (CCDs) [54]. Then, the spatial resolution is limited by the spatial sampling rate, that is, the number of photo-detectors per unit length along a particular direction.

The process of taking real scenes and converting them into pixels can be mathematically formalized by the processes of image acquisition and sampling. The optical system can be modeled as a spatial linear lowpass filter acting on the continuous two-dimensional signal. The output band-limited signal is then sampled at the sampling rate determined by the spatial distribution of the photo-detectors. While

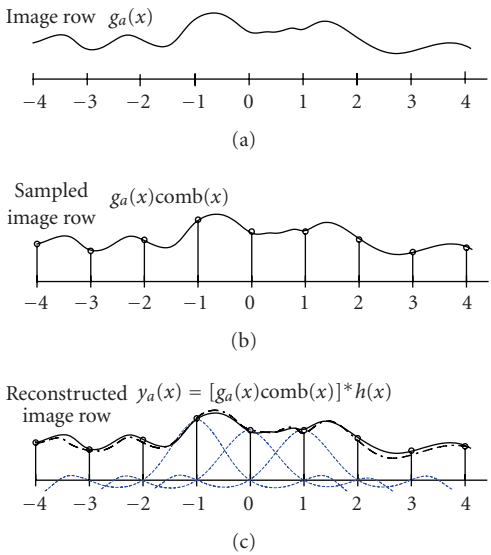


FIGURE 7.1. Sampling and reconstruction operations. (a) Initial continuous signal; (b) sampled at integers; (c) reconstructed by linear convolution (dashed-dotted line).

the optical system somehow blurs the signal, an insufficient sampling rate, caused by detectors spaced few and far between, can introduce aliasing. These two effects are inevitable. However, they have been well studied and are taken into account when designing modern imaging systems [14, 54].

Consider the one-dimensional function $r(x)$ representing a true scene along the x-axis. As a result of optical blurring, modeled as linear convolution with a kernel $s(x)$, we obtain another continuous function $g_a(x)$, smoother and band-limited (Figure 7.1(a)):

$$g_a(x) = (r * s)(x). \tag{7.1}$$

For the sake of simplicity we assume that the initial sampling grid is placed on the integer coordinates and generate the discrete sequence $g[k] = g_a(x)$. In continuous time it can be modeled by the product $g_a(x)\,\text{comb}(x)$, where

$$\text{comb}(x) = \sum_{k=-\infty}^{\infty} \delta(x-k). \tag{7.2}$$

The resulting impulse train is a sum of integer-shifted ideal impulses weighted by the signal values at the same coordinates

$$g_p(x) = g_a(x) \sum_{k=-\infty}^{\infty} \delta(x-k) = \sum_{k=-\infty}^{\infty} g[k]\delta(x-k). \tag{7.3}$$

Figure 7.1(b) illustrates the sampling process.

The blurring effect can be quantified as the difference between the functions $r(x)$ and $g_a(x)$ measured by the L_2 error norm

$$\varepsilon^2_{\text{blur}} = \|r - g_a\|_{L_2} = \int_{-\infty}^{\infty} \big(r(x) - g_a(x)\big)^2 dx. \tag{7.4}$$

It is expressible in the Fourier domain as

$$\varepsilon^2_{\text{blur}} = \int |1 - S(2\pi f)|^2 \, |R(2\pi f)|^3 df, \tag{7.5}$$

where $R(2\pi f)$ is the Fourier transform of the initial signal $r(x)$ and $S(2\pi f)$ is the frequency response of the (optical) blurring system.

The effect of sampling is well studied in the Fourier domain. The Fourier transform $G_p(2\pi f)$ of the product $g_a(x)\,\text{comb}(x)$ contains the original spectrum of $g_a(x)$ and its replications around multiples of 2π [43],

$$G_p(2\pi f) = \sum_{n=-\infty}^{\infty} G_a\big(2\pi(f - n)\big), \tag{7.6}$$

where $G_a(2\pi f)$ is Fourier transform of the blurred continuous signal $g_a(x)$.

Any overlap of the replicated spectra can cause a degradation of the original spectrum. The effect is known as *aliasing*.

7.2.2. Interpolation

Generally speaking, the role of the interpolation is to generate some missing intermediate points between the given discrete pixels. First, based on the existing discrete data, a continuous signal is generated and then, the desired interpolated samples are obtained by resampling it at the desired coordinates. In this general setting, the interpolation factor is *arbitrary*, not necessarily an integer, not even a rational number. In most cases, the continuous (analog) model fitting is performed by convolving the samples with some appropriate continuous interpolating kernel [62], as illustrated in Figure 7.1(c).

The interpolation does not recover the original scene continuous signal itself. In most cases one starts directly with discrete data and does not know the characteristics of the optical system nor the sampling rate and the effects introduced by the sampling device. What one can do is to try matching a continuous model that is consistent with the discrete data in order to be able to perform some further continuous processing, such as differentiation, or to resample the fitted continuous function into a finer grid. Within this consideration, interpolation does not deal with inverse problems such as deconvolution, although some knowledge about the acquisition and sampling devices could help in the choice of the interpolating function [66].

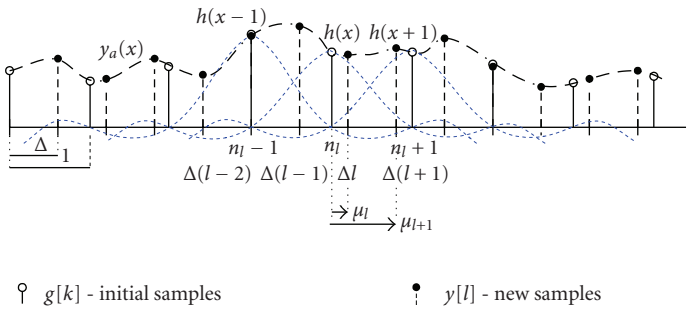


FIGURE 7.2. Interpolation problem in signal domain. White circles represent initial discrete sequence $g[k]$ with sampling ratio 1. Black circles represent new sequence $y[l]$ taken with smaller sampling interval Δ from continuous function $y_a(x)$.

7.2.2.1. Interpolation problem in signal domain

Assume that there exist given discrete data $g[k]$ defined on a uniform grid $\boldsymbol{\kappa} \equiv \{0, 1, 2, \ldots, L_{\text{in}} - 1\}$, $k \in \boldsymbol{\kappa}$, with sampling step equal to unity, as shown in Figure 7.2. The $g[k]$'s can represent, for example, the pixel intensities in an image row or column. Furthermore, it is supposed that the data have been taken from a certain continuous function $g_a(x)$.

The general interpolation problem is then to generate a new sequence $y[l]$ with sampling points located at new points between the existing samples $g[k]$ with the goal being to yield an image (row or column) with an increased number of pixels (a magnified, a zoomed in image). The new sequence $y[l]$ can be generated by first fitting an approximating continuous function $y_a(x)$ to the given sequence $g[k]$ and then resampling it at the desired new sampling points, as shown in Figure 7.2. For the uniform resampling, $y[l] = y_a(x_l) = y_a(l\Delta)$ with $\Delta < 1$.

The function $y_a(x)$ approximating the original continuous signal $g_a(x)$ can be generated by a linear, convolution-type model:

$$y_a(x) = \sum_k g[k]h(x - k). \tag{7.7}$$

Here, $h(x)$ is a continuous convolution kernel (interpolation filter) mapping the discrete data onto the continuous model function $y_a(x)$. The only information that we have about the continuous function $g_a(x)$ is conveyed by the samples $g[k]$ and a natural assumption is that $g[k] = g_a(k)$. We require the same from the new function $y_a(x)$, that is, for all $k_0 \in \boldsymbol{\kappa}$, $y_a(k_0) = g[k_0]$. This requirement imposes the so-called *interpolation constraint* for the kernel $h(x)$ as follows:

$$y_a(k_0) = \sum_k g[k]h(k_0 - k) \Longrightarrow h(k_0 - k) = \begin{cases} 1, & \text{for } k = k_0, \\ 0, & \text{otherwise.} \end{cases} \tag{7.8}$$

The interpolation kernel should behave like $h(k) = \delta(k)$, where $\delta(k)$ is the Kroneker symbol. We say that the kernel $h(x)$ is *interpolating*.

A more general model can be stated. Consider another convolution kernel $\varphi(x)$ which generates the approximating function $y_a(x)$:

$$y_a(x) = \sum_k d[k]\varphi(x-k). \tag{7.9}$$

The model coefficients $d[k]$ do not (necessarily) coincide with the initial signal samples $g[k]$ and the reconstruction kernel $\varphi(x)$ is not required to pass through zeros for the integer coordinates (*noninterpolating* kernel). Still, we require $y_a(k_0) = g[k_0]$ for all $k_0 \in \boldsymbol{\kappa}$, that is,

$$y_a(k_0) = \sum_k d[k]\varphi(k_0-k) = g[k_0]. \tag{7.10}$$

Equation (7.10) determines uniquely the model coefficients. It is in fact a discrete convolution between two sequences: the model coefficients and the interpolation function $\varphi(x)$ sampled at integers. Denote it by $p[k] = \varphi(k)$ and write the convolution (7.10) in z-domain:

$$Y_a(z) = D(z)P(z), \tag{7.11}$$

where $Y_a(z)$, $D(z)$, and $P(z)$ are the z-transforms of $y_a(k) = g[k]$, $d[k]$, and $p[k]$, respectively. The sequence of model coefficients is obtained by a recursive digital filtering, as follows:

$$D(z) = \frac{Y_a(z)}{P(z)} = \frac{G(z)}{P(z)}. \tag{7.12}$$

Provided that φ is a finitely supported and symmetric function (as in most cases of interest), its sampled version is a symmetric finite length sequence $p[k]$. The recursive digital filter $1/P(z)$ is regarded as corresponding to an infinite discrete sequence, $(p^{-1})[k]$ being the convolution inverse of the sequence $p[k]$, that is,

$$\left(p * p^{-1}\right)[k] = \delta[k]. \tag{7.13}$$

In time domain the filtering (7.12) can be formally expressed as

$$d[k] = \left(p^{-1} * g\right)[k]. \tag{7.14}$$

Consequently, the interpolation approach (7.9) involves the preliminary filtering step (7.14) to get the model coefficients $d[k]$. It is not a costly operation and

for a large class of reconstruction kernels, can be realized through an efficient IIR filtering [62, 69, 70, 72]. The benefit is that the function $\varphi(x)$ can be chosen with better properties, in particular it can be nonoscillatory by contrast to interpolating function $h(x)$.

The model (7.9) can be brought to the model (7.7) by replacing $d[k]$ in (7.9) with (7.14),

$$y_a(x) = \sum_k (p^{-1} * g)[k]\varphi(x-k) = \sum_k g[k](p^{-1} * \varphi)(x-k). \tag{7.15}$$

The convolution

$$h(x) = \sum_k (p^{-1})[k]\varphi(x-k) \tag{7.16}$$

results in an infinitely supported interpolating kernel having the following frequency response:

$$H(2\pi f) = \frac{\Phi(2\pi f)}{P(e^{j2\pi f})}, \tag{7.17}$$

where $\Phi(2\pi f)$ is the continuous frequency response of $\varphi(x)$ and $P(e^{j2\pi f}) = P(z)|_{z=e^{j2\pi f}}$.

After building the continuous model as given by (7.7) or (7.9), it is resampled at new discrete points to obtain $y(l) = y_a(x_l)$. Here, the current output coordinate x_l can be expressed as $x_l = n_l + \mu_l$, where $n_l = \lfloor x_l \rfloor$ is the coordinate (integer) of $g[n_l]$, occurring just before or at x_l. The interval $\mu_l = x_l - n_l$, in turn, is a *fractional interval* in the range $0 \le \mu_l < 1$. Given n_l and μ_l, the current output pixel can be generated following (7.7):

$$y[l] = \sum_k g[n_l - k]h(k + \mu_l). \tag{7.18}$$

From this equation, the interpolation kernel $h(x)$ can be interpreted also as a filter with varying impulse response that depends on the value of μ_l. That is why in the literature it appears under different names such as *interpolation kernel*, *interpolation filter*, *fractional delay filter*, and *reconstruction filter*, emphasizing some of its properties [77].

Despite its different names, the kernel should be as follows.

(i) The kernel is desired to be *symmetrical* to avoid introducing phase distortions.

(ii) The kernel's *support* should be *as short as possible* for the desired interpolation accuracy. This decreases the computational complexity, especially for high-dimensional cases. Regarding the model (7.9), the support of the reconstruction function φ is envisaged. In fact, for this model, the cardinal interpolation function is infinitely supported because of the IIR term, but still the computational complexity would be sufficiently low for short-length kernels.

(iii) The kernel should ensure good closeness of the approximating function $y_a(x)$ to the unknown function $g_a(x)$. This closeness can be measured via different measures, that is,

(1) in *frequency domain*, thus invoking the terms of *passband preservation* and *stopband attenuation* [55, 77];
(2) by the kernel's *approximation order*, measuring its capability to reproduce polynomials up to some degree M and characterizing the rate of decay of the approximation error when the sampling interval tends to zero [62, 63].

These terms and some adequate measures of goodness will be discussed in detail later in this chapter.

7.2.2.2. Interpolation problem in frequency domain

The Fourier transform of the initial sequence $g[k]$ is the periodical version of $G_a(f)$:

$$G(e^{j2\pi f}) = \sum_{n=-\infty}^{\infty} G_a(2\pi(f-n)). \tag{7.19}$$

The continuous function $y_a(x)$, as given by (7.7), can be expressed in the frequency domain as

$$Y_a(2\pi f) = H(2\pi f)G(e^{j2\pi f}) = H(2\pi f)\sum_{k=-\infty}^{\infty} G_a(2\pi(f-k)). \tag{7.20}$$

See Figure 7.3 for an illustration of the effects of sampling and reconstruction in frequency domain.

Sinc (ideal) reconstruction kernel. In the ideal case it is desired that $Y_a(2\pi f) = G_a(2\pi f)$. The classical assumption is that the signal $g_a(x)$ is band-limited (or had been filtered to be band-limited), that is, $G_a(2\pi f)$ is zero for higher frequencies. The classic Shannon theorem [57] states that a band-limited signal $g_a(x)$ can be restored from its samples taken at integer coordinates, provided that $G_a(2\pi f)$ is zero for $|f| > 1/2$. Otherwise the sampling introduces aliasing. The interpolation starts where sampling has finished: from the discrete sequence $g[k]$ having a periodical Fourier representation. The interpolation cannot restore frequencies distorted by aliasing. The best it can do is to try to restore a continuous function $y_{\text{ideal}}(x)$ having Fourier transform

$$Y_{\text{ideal}}(2\pi f) = \begin{cases} G(e^{j2\pi f}), & \text{for } 0 \le f \le \dfrac{1}{2}, \\ 0, & \text{for } f > \dfrac{1}{2}, \end{cases} \tag{7.21}$$

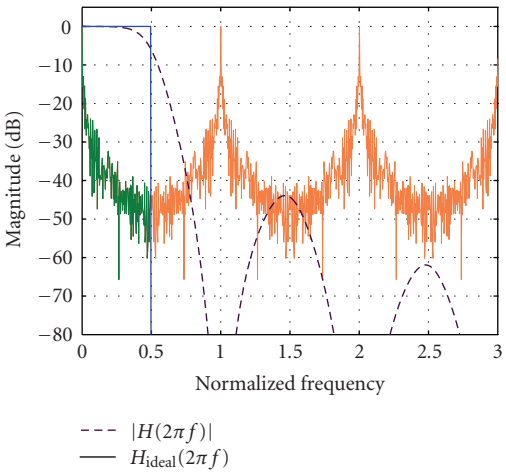


FIGURE 7.3. Effects of sampling and reconstruction in frequency domain. $|G(e^{j2\pi f})|$ is replicated version of $|G_a(2\pi f)|$. $H_{\text{ideal}}(2\pi f)$ (solid line) is finite supported in frequency domain, while $|H(2\pi f)|$ (dashed line) has side lobes.

and hope that the sampling has introduced no (little) aliasing. Hence, the ideal interpolation (reconstruction) is achieved by the filter

$$H_{\text{ideal}}(2\pi f) = \begin{cases} 1, & \text{for } 0 \le f \le \dfrac{1}{2}, \\ 0, & \text{for } f > \dfrac{1}{2}. \end{cases} \tag{7.22}$$

In the signal domain, the ideal filter is inversely transformed to the sinc function

$$h_{\text{ideal}}(x) = \int_{-\infty}^{\infty} H_{\text{ideal}}(2\pi f) e^{-j2\pi f x} df = \frac{\sin(\pi x)}{\pi x} = \text{sinc}(x). \tag{7.23}$$

The reconstruction equation for the case of the sinc reconstruction function is

$$y_{\text{ideal}}(x) = \sum_{k=-\infty}^{\infty} g[k]\,\text{sinc}(x-k). \tag{7.24}$$

In the real world, the ideal filtering cannot be achieved since the sinc reconstruction function is infinitely supported (note the summation limits in (7.24)). Even if the initial signal $g[k]$ is given on the finite interval $0 \le k \le M-1$ only, to reconstruct the continuous function $g_a(x)$ in this interval, we need an infinite summation. For signals given on a finite interval of M samples, the sinc-function should be replaced by a discrete the sinc-function $\text{sincd}(x) = \sin(\pi x)/(M \sin(\pi x/M))$ which is an ideal, to the accuracy of boundary effects, interpolator of discrete

signals with finite support [82, 83]. This interpolation method is considered in Chapter 8.

Nonideal reconstruction kernels. Finitely supported nonideal reconstruction kernels have infinite frequency response. Being attractive due to their easier implementation, they never cancel fully the replicated frequencies above the Nyquist rate. Figure 7.3 illustrates this effect.

An explicit relation between the output discrete signal $y[l]$ and the input signal $g[k]$ can be established, assuming a uniform resampling of $y_a(x)$ with the resampling step $\Delta < 1$ (finer than the initial one, i.e., the resampling generates more pixels). The Fourier transform of the output signal $y[l]$ can be expressed in the following forms:

$$Y\left(e^{j2\pi f\Delta}\right) = \frac{1}{\Delta}\sum_{n=-\infty}^{\infty} Y_a\left(2\pi\left(f - \frac{n}{\Delta}\right)\right), \tag{7.25}$$

$$Y\left(e^{j2\pi f\Delta}\right) = \frac{1}{\Delta}\sum_{n=-\infty}^{\infty} H\left(2\pi\left(f - \frac{n}{\Delta}\right)\right) G\left(e^{j2\pi(f-n/\Delta)}\right). \tag{7.26}$$

This clarifies that the interpolation process generates some frequency regions which are mirrors of the passband frequency region with respect to $2\pi/\Delta$.

As an example, let us take the simplest interpolator, the so-called nearest neighbor. Practically, this interpolator repeats the value at the closest integer coordinate to the needed new coordinate. Following (7.7), this repetition is modeled by a convolution between the given discrete sequence and a rectangular reconstruction function having support between $-1/2$ and $1/2$. The Fourier transform of this reconstruction function is a sinc function in the frequency domain, that is, $\sin(\pi f)/\pi f$. It has high magnitude side lobes that overlap (not suppress!) the periodical replicas in the product (7.20) and in (7.26), respectively. Moreover, it is not flat for the frequencies below half the sampling rate and it attenuates considerably the true (passband) frequencies of the original signal resulting in visible blur.

The two extreme cases, namely, sinc and nearest neighbor interpolators suggest directions for improving the performance of finite support interpolators. With some preliminary knowledge about the frequency characteristics of $g_a(x)$, the interpolator $h(x)$ can be designed in such a way that its frequency response suppresses effectively the undesired components of the interpolated signal. This is equivalent to designing some application-specific optimized approximation of the ideal reconstruction filter in frequency domain.

7.2.2.3. Interpolation artifacts

Before studying some quantitative measures of interpolators' performance let us summarize qualitatively the distortions caused by nonideal interpolators.

Ringing. Ringing is a result of the oscillatory type of interpolators combined with the Gibbs effects due to the finite terms approximation of continuous functions in Fourier domain. Ringing effect occurs even for the ideal sinc interpolators realized in Fourier domain. Actually, those are not true artifacts since they can arise together with the perfect recovering of the initial samples [82].

Blurring. Blurring is a result of the nonideality of the reconstruction function in the passband. Instead of preserving all frequencies in this region, nonideal interpolators suppress some of them, especially in the high-frequency area (close to half the sampling rate). As a result, the interpolated images appeared with no sharp details, that is, blurred.

Aliasing. Aliasing is an effect due to improper sampling. This is the effect of the appearing of unwanted frequencies (hence the term aliasing) as a result of the repetition of the original spectrum around multiples of the sampling rate. The aliasing artifacts resulting from insufficient sampling may appear as Moiré patterns. For small images, they may be hardly noticeable, but after interpolation on a finer grid they can become visible.

Imaging. This is the counterpart of aliasing, though the term "imaging" is somehow confusing. Consider the simple case of sampling rate expansion by an integer factor of L. It can be accomplished by first inserting $L - 1$ zeros between the given samples (up-sampling) and then smoothing the new sequence with a digital filter. The up-sampling causes "stretching" of the frequency axis [43]. As a result, in the passband the original spectrum appears together with its $L - 1$ "images." The role of the smoothing filter is to remove these unwanted frequencies. Following the model (7.7), we first apply the reconstruction operation (fitting the continuous model $y_a(x)$) and subsequently the resampling. Assuming this model, unwanted frequencies can interfere into the passband during the process of resampling as a result of nonsufficient suppression of the frequency replicas during the previous step of continuous reconstruction. Hence, this effect can be again characterized as aliasing, and we will use this term in order not to confuse "imaging" with digital images. The effects of possible sampling-caused and reconstruction-caused aliasings are almost undistinguishable since they appear simultaneously (and with blurring) in the resampled image. The reconstruction-caused aliasing effect is emphasized for short-length kernels (i.e., nearest neighbor or linear) and appears in the form of blocking (pixelation) in the magnified image [62, 82].

7.2.2.4. Interpolation error kernel

There are two sources of errors caused by the nonideality of the reconstruction kernel to be quantified. First, there is the nonflat magnitude response in the passband (causing blurring) and second, there is the nonsufficient suppression of the periodical replicas in the stopband (causing aliasing). As the blurring and aliasing errors both superimpose into the resampled image, it is good to have an integral measure for both of them. Such a measure would play a substantial role in optimizing and comparing different interpolators. We review the interpolation error

kernel as it has appeared in the work of Park and Schowengerdt [48]. Then, based on some approximation theory presumptions, we review a generalized form of this kernel, appropriate for the interpolation model (7.9), see [10, 64].

Sampling and reconstruction (SR) blur. Park and Schowengerdt [48] have investigated the influence of the phase of sampling on the sampling and reconstruction accuracy. Return to image acquisition model (7.1)-(7.2) and insert an additional parameter u, showing the relative position of the sampling device with respect to the function $s(x)$:

$$g_a(x-u) = (r * s)(x-u). \tag{7.27}$$

Sampling at integers and reconstructing by a continuous interpolation function give

$$y_a(x,u) = \sum_{k=-\infty}^{\infty} g_a(n-u)h(x-k). \tag{7.28}$$

Here, the reconstructed function also depends on the sampling phase and it is not simply a function of the difference $x-u$. The error between $g(x-u)$ and $y(x,u)$, called by the authors sampling and reconstruction (SR) blur, is a function of u as well:

$$\varepsilon^2_{\mathrm{SR}}(u) = \int_{-\infty}^{\infty} \big(g_a(x-u) - y(x,u)\big)^2 dx. \tag{7.29}$$

One can observe that the phase-dependent SR error is a periodic function of u with a period of unity and hence it can be represented as the Fourier series

$$\varepsilon^2_{\mathrm{SR}}(u) = \sum_{m=-\infty}^{\infty} a_m e^{j2\pi um}, \tag{7.30}$$

where

$$a_m = \int_0^1 \varepsilon^2_{\mathrm{SR}}(u) e^{-j2\pi um} du. \tag{7.31}$$

The coefficient a_0 plays the most important role since it quantifies the *average* value of the SR blur. It can be regarded as the expectation of the random error, depending on the arbitrary sampling phase. It can be obtained by taking the SR blur in frequency domain (Parseval's equality)

$$\begin{aligned}\varepsilon^2_{\mathrm{SR}}(u) = \int_{-\infty}^{\infty} &\left[\sum_{k=-\infty}^{\infty} \big(\delta_k - H(2\pi f)\big) G_a\big(2\pi(f-k)\big) e^{j2\pi ku}\right] \\ &\times \left[\sum_{n=-\infty}^{\infty} \big(\delta_n - H^*(2\pi f)\big) G_a^*\big(2\pi(f-n)\big) e^{-j2\pi nu}\right] df\end{aligned} \tag{7.32}$$

and integrating over u

$$a_0 = \int_{-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} (\delta_k - H(2\pi f)) G_a(2\pi(f-k)) \times (\delta_n - H^*(2\pi f)) G_a^*(2\pi(f-n)) \int_0^1 e^{j2\pi(k-n)u} du\, df. \tag{7.33}$$

After some simplifications we get

$$\begin{aligned} E\{\varepsilon_{\mathrm{SR}}^2\} = a_0 &= \int_{-\infty}^{\infty} |G_a(2\pi f)|^2 \left(1 - 2\,\mathrm{Re}\,(H(2\pi f)) + |H(2\pi f)|^2\right) \\ &\quad + |H(2\pi f)|^2 \sum_{k\neq 0} |G_a(2\pi(f-k))|^2 df \\ &= \int_{-\infty}^{\infty} \Big\{ |G_a(2\pi f)|^2 |(1 - H(2\pi f))|^2 \\ &\quad + |H(2\pi f)|^2 \sum_{k\neq 0} |G_a(2\pi(f-k))|^2 \Big\} df. \end{aligned} \tag{7.34}$$

The first term in the summation inside the integral is the error introduced by the nonideality of the reconstruction kernel (i.e., blurring term), while the second term represents the aliasing error due to the nonideality of the reconstruction kernel for the replicated frequencies. By changing the variable, this term can be rewritten in an equivalent form as follows:

$$\begin{aligned} &\int_{-\infty}^{\infty} |H(2\pi f)|^2 \sum_{k\neq 0} |G_a(2\pi(f-k))|^2 df \\ &\quad = \int_{-\infty}^{\infty} |G_a(2\pi f)|^2 \sum_{k\neq 0} |H(2\pi(f-k))|^2 df. \end{aligned} \tag{7.35}$$

The average SR blur takes a very compact form which separates the influence of the reconstruction kernel and the initial signal

$$E\{\varepsilon_{\mathrm{SR}}^2\} = \int_{-\infty}^{\infty} \eta^2(2\pi f) |G_a(2\pi f)|^2 df, \tag{7.36}$$

where

$$\eta^2(2\pi f) = |1 - H(2\pi f)|^2 + \sum_{k\neq 0} |H(2\pi(f-k))|^2. \tag{7.37}$$

Here, the nonnegative kernel η^2 quantifies the amount of errors introduced by the reconstruction kernel.

The total SR error is

$$\varepsilon_{\mathrm{SR}}^2(u) = \varepsilon_S^2(u) + \varepsilon_R^2(u) + \phi(u), \tag{7.38}$$

where

$$\begin{aligned} \varepsilon_S^2 &= \int_{-\infty}^{\infty} \left| G(2\pi f) \right|^2 \sum_{k\neq 0} \left| H(2\pi(f-k)) \right|^2 df, \\ \varepsilon_R^2 &= \int_{-\infty}^{\infty} \left| G(2\pi f) \right|^2 \left| 1 - H(2\pi f) \right|^2 df, \\ \phi(u) &= \sum_{m\neq 0} a_m e^{j2\pi m u}. \end{aligned} \tag{7.39}$$

The term ε_S^2 is regarded as the error due to insufficient sampling. It is also this term that is responsible for introducing the aliasing errors if the reconstructed signal is resampled into a finer grid. The term ε_R^2 shows the error contribution of a reconstruction function which is different from unity in the frequency areas of significant signal energy. The last, namely, the sampling phase-dependant term $\phi(u)$, determined by the higher harmonics in the Fourier series (7.30), is a zero mean random variable with variance $2\sum_{m=1}^{\infty} |a_m|^2$. It is always zero if the initial signal is essentially band-limited and sufficiently sampled [48]. For a particular *nonband-limited* signal, equations for a_m should be written and calculated (numerically). Many authors prefer just to study the behavior of the averaged error for different signals and different reconstruction kernels [40, 48, 49, 56].

A generalization of the error kernel (7.37) can be obtained based on the reconstruction model (7.9). Write the cardinal interpolation kernel (7.16) in frequency domain

$$H(2\pi f) = \frac{\Phi(2\pi f)}{P\left(e^{j2\pi f}\right)} = \frac{\Phi(2\pi f)}{\sum_{k=-\infty}^{\infty} \Phi(2\pi(f-k))}, \tag{7.40}$$

where $P(e^{j2\pi f}) = \sum_{k=-\infty}^{\infty} \Phi(2\pi(f-k))$ is the frequency response of the sampled version $p(k)$ of the reconstruction function $\varphi(x)$. The error kernel (7.37) becomes

$$\begin{aligned} \eta^2(2\pi f) &= \left| 1 - \frac{\Phi(2\pi f)}{P\left(e^{j2\pi f}\right)} \right|^2 + \sum_{k\neq 0} \left| \frac{\Phi(2\pi(f-k))}{P\left(e^{j2\pi(f-k)}\right)} \right|^2 \\ &= \frac{\left| P\left(e^{j2\pi f}\right) - \Phi(2\pi f) \right|^2 + \sum_{k\neq 0} \left| \Phi(2\pi(f-k)) \right|^2}{\left| P\left(e^{j2\pi f}\right) \right|^2} \end{aligned} \tag{7.41}$$

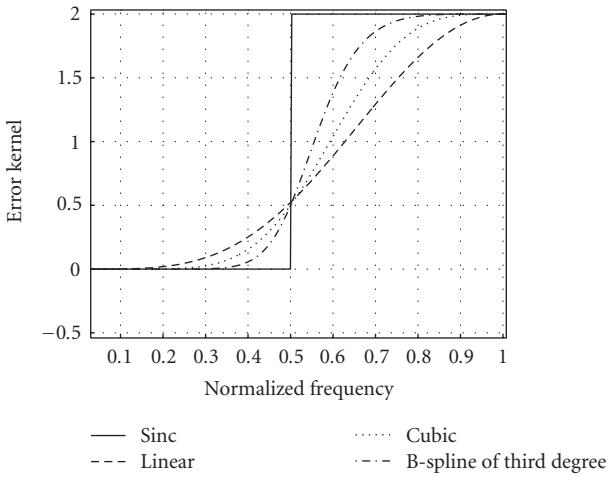


FIGURE 7.4. Error kernel for four classical interpolators.

or has the following equivalent form [64]:

$$\eta^2(j2\pi f) = \frac{\left|\sum_{k\neq 0}\Phi(2\pi(f-k))\right|^2 + \sum_{k\neq 0}\left|\Phi(2\pi(f-k))\right|^2}{\left|\sum_{k=-\infty}^{\infty}\Phi(2\pi(f-k))\right|^2}. \tag{7.42}$$

Figure 7.4 shows the error kernel for the ideal (sinc) interpolator and for the probably most used interpolating linear and cubic interpolators, and for the noninterpolating B-spline interpolator of third degree [33, 48, 49, 62]. The improvement of cubic versus linear interpolator is achieved thanks to the longer support and the higher degree. The B-spline has the same degree as the cubic convolution interpolator and the improvement is achieved thanks to the noninterpolating model applied. B-spline basis functions are described in the next section.

The role of the error kernel will be additionally highlighted from an approximation theory point of view.

How to compute the error kernel. Since the class of symmetrical, finitely supported interpolators is of major interest, we give some practical directions regarding how to compute the error kernel (7.42) for this class. Assume the function $\varphi(x)$ is supported over the interval $[-(N+1)/2, (N+1)/2]$ and is symmetrical, that is, $\varphi(-x) = \varphi(x)$. Its sampled version, given by the sequence $p(k) = \varphi(x)|_{x=k}$, is also symmetrical and has $2\lfloor N/2\rfloor + 1$ nonzero terms. The Fourier transform of this sequence is given by

$$P(z)\big|_{z=e^{j2\pi f}} = \varphi(0) + 2\sum_{k=1}^{\lfloor N/2\rfloor}\varphi(k)\cos(2\pi k f). \tag{7.43}$$

For such a reconstruction function $\varphi(x)$, (7.41) can be simplified,

$$\begin{aligned}
&\eta^2(2\pi f) \\
&= \frac{\left(P\left(e^{j2\pi f}\right) - \Phi(2\pi f)\right)^2 + \sum_{k\neq 0}\left(\Phi(2\pi(f-k))\right)^2}{\left(P\left(e^{j2\pi f}\right)\right)^2} \\
&= \frac{\left(P\left(e^{j2\pi f}\right)\right)^2 - 2P\left(e^{j2\pi f}\right)\cdot\Phi(2\pi f) + \left(\Phi(2\pi f)\right)^2 + \sum_{k\in\mathbb{Z}_*}\left(\Phi(2\pi(f-k))\right)^2}{\left(P\left(e^{j2\pi f}\right)\right)^2} \\
&= \frac{\left(P\left(e^{j2\pi f}\right)\right)^2 - 2P\left(e^{j2\pi f}\right)\cdot\Phi(2\pi f) + \sum_{k=-\infty}^{\infty}\left(\Phi(2\pi(f-k))\right)^2}{\left(P\left(e^{j2\pi f}\right)\right)^2}.
\end{aligned} \tag{7.44}$$

Although it is an infinite sum, the last term in the numerator can be calculated rather easily, realizing that it is the Fourier transform of the function $\varphi(x)$ continuously convolved by itself and subsequently sampled. The autoconvolution (which is in fact an autocorrelation) is also symmetrical and has a compact support of $2N + 2$. Hence, only N cosine terms take part in the considered infinite sum

$$\sum_{k=-\infty}^{\infty}\left(\Phi(2\pi(f-k))\right)^2 = (\varphi * \varphi)(0) + 2\sum_{k=1}^{N}(\varphi * \varphi)(k)\cos(2\pi k f). \tag{7.45}$$

The algorithm for calculating the error kernel can be summarized as follows.

(1) Calculate the Fourier transform $\Phi(2\pi f)$ of the continuous kernel $\varphi(x)$.
(2) Calculate the Fourier transform $P(e^{j2\pi f})$ of the discrete kernel $p(k) = \varphi(x)|_{x=k}$, as given by (7.43).
(3) Calculate the autocorrelation function of the symmetrical continuous function $\varphi(x)$, $a_\varphi(x) = (\varphi * \varphi)(x)$ at integer coordinates $k : a[k] = a_\varphi(x)|_{x=k}$.
(4) Find the Fourier transform of the sequence $a(k)$, according to (7.45).
(5) Calculate the kernel (7.44).

7.2.3. Shift-invariant function spaces and generating bases

The model (7.9) has, in fact, a wider scope than the interpolation we have used it for. For an arbitrary choice of the model sequence $d[k]$, the functions $\varphi(x-k)$ span a vector space. The function space reformulation of the model (7.9) allows considering other schemes for obtaining the sequence $d[k]$, apart from the simple interpolation [10].

Consider the following *shift-invariant* function space $V(\varphi)$ that is a closed subspace of L^2 and *generated* by a function φ as

$$V(\varphi) : \left\{ g_V(x) = \sum_{k=-\infty}^{\infty} d[k]\varphi(x-k) : c \in l^2 \right\}. \tag{7.46}$$

Any function from the space $V(\varphi)$ can be represented as a convolution between a discrete set of coefficients and the generating function φ. In other words, the integer shifts of φ form a basis for the space $V(\varphi)$ and the coefficients $d[k]$ are regarded as coordinates of the function $g_V(x)$ with respect to the given basis.

The generating function has to be chosen in an appropriate manner in order to ensure that the space $V(\varphi)$ is a closed subspace of L^2, that is, φ must form a *Riesz basis* [3]. By definition, the family of basis functions $\varphi(x-k)$ forms a Riesz basis if there exist two constants $B > A > 0$ such that

$$A\|d\|_{l^2}^2 \leq \left\| \sum_{k=-\infty}^{\infty} d[k]\varphi(x-k) \right\|_{L^2}^2 \leq B\|d\|_{l^2}^2. \tag{7.47}$$

The upper bound ensures that $V(\varphi)$ is a well-defined subspace of L^2, while the lower bound provides that it is a closed subspace of L^2.

A function in $V(\varphi)$ is entirely described by the discrete-time sequence $d[k]$. Any function $g_a(x) \in L^2$ can be approximated by a function in $V(\varphi)$. The interpolation is the simplest way of approximating $g_a(x)$ as was discussed in the previous sections. The least squares (LS) approximation is achieved by a preliminary filtering with an analysis (projecting) function. Since $V(\varphi)$ is closed, the LS approximation $\widetilde{g}_a(x)$ exists and it is equal to the orthogonal projection of $g_a(x)$ into $V(\varphi)$. In general, the basis $\varphi(x-k)$ is not orthogonal, but only linearly independent. Then, the orthogonal projection of $g_a(x) \in L^2$ into $V(\varphi)$ is obtained by

$$\widetilde{g}_a(x) = \sum_{k=-\infty}^{\infty} d[k]\varphi(x-k), \tag{7.48}$$

where

$$d[k] = \int_{-\infty}^{\infty} g_a(\tau)\widetilde{\varphi}(k-\tau)d\tau. \tag{7.49}$$

Here, $\widetilde{\varphi}(x)$ is the dual (biorthogonal) basis of φ. As it belongs to the same space V, it can be represented as a linear combination of the generating basis φ as well. Moreover, its coordinate coefficients are merely the autocorrelation sequence of φ, defined at the integer values as

$$\begin{aligned} \widetilde{\varphi} &\in V(\varphi), \\ \widetilde{\varphi}(x) &= \left((a)^{-1} * \varphi\right)(x), \\ a[k] &= \int_{-\infty}^{\infty} \varphi(\tau)\varphi(\tau-k)d\tau. \end{aligned} \tag{7.50}$$

The proof of this relation can be found, for example, in [3].

Taking the Fourier transform of (7.50) we obtain the dual basis in Fourier domain

$$\widetilde{\Phi}(2\pi f) = \frac{\Phi(2\pi f)}{A(e^{j2\pi f})} = \frac{\Phi(2\pi f)}{\sum_{n=-\infty}^{\infty} |\Phi(2\pi(f-n))|^2}. \tag{7.51}$$

7.2.4. Approximation theory impact

In order to quantify the approximation error between $g_a(x)$ and its approximation as a function of the sampling step Δ we write a scale-dependent convolution-based signal expansion:

$$y_\Delta(x) = \sum_k d[k]\varphi\left(\left(\frac{x}{\Delta}\right) - k\right). \tag{7.52}$$

This is an expansion of a signal belonging to the space generated by the reconstruction function φ parameterized by the sampling (scale) parameter Δ,

$$V_\Delta = \operatorname*{span}_{k\in\mathbb{Z}} \left\{\varphi\left(\left(\frac{x}{\Delta}\right) - k\right)\right\}. \tag{7.53}$$

The expansion coefficients $d[k]$ can be obtained either by interpolation or by projection of $g_a(x)$ onto V_Δ, for example, LS approximation. In both cases, the approximation theorists have been interested in quantifying the error between $g_a(x)$ and $y_\Delta(x)$ as a function of the decreasing and eventually vanishing sampling step Δ represented in the form

$$e^2(\Delta) = E\{\varepsilon^2(\Delta)\} = \int_{-\infty}^{\infty} \eta^2(2\pi\Delta f)\,|G_a(2\pi f)|^2 df. \tag{7.54}$$

Similar to [48], this error is expressed in [10, 62] as a prediction (in a probabilistic sense) of the true approximation mean square error between $g_a(x)$ and the approximating function $y_\Delta(x)$ (7.52). The error kernel $\eta^2(2\pi f)$ for the case of LS approximation can be obtained by transforming (7.48) and (7.49) to the Fourier domain and using (7.51). It takes the rather simple form [10, 12]:

$$\eta^2_{\mathrm{LS}}(2\pi f) = 1 - \frac{|\Phi(j2\pi f)|^2}{\sum_{n=-\infty}^{\infty} |\Phi(j2\pi(f-n))|^2}. \tag{7.55}$$

Further, we consider the concept of Lth-order basis function. It is important for both projection and interpolation cases. Lth order of approximation shows the rate of decay of the error as the sampling step tends to zero.

Without loss of generality we will consider the approximation error for the interpolation case and will just mark the differences for the LS case. The error

kernel (7.42) is an even function and can be developed in MacLaurin series:

$$\eta^2(2\pi f) = \sum_{n=0}^{\infty} \frac{\left(\eta^2(0)\right)^{(2n)}}{(2n)!} f^{2n}, \tag{7.56}$$

where the $(\eta^2(\cdot))^{(2n)}$ is the $2n$th derivative of the error kernel.

Approximation order of the basis function φ is the lowest order of differentiation L, for which $(\eta^2(\cdot))^{(2L)} \neq 0$, while $(\eta^2(\cdot))^{(2n)} = 0$ for all $n\ 0 \leq n \leq L-1$,

$$\eta^2(2\pi f) = \left(C_\varphi\right)^2 f^{2L} + \sum_{n=L+1}^{\infty} \frac{\left(\eta^2(0)\right)^{(2n)}}{(2n)!} f^{2n}, \tag{7.57}$$

where C_φ is a constant depending on φ as follows:

$$C_\varphi = \sqrt{\frac{\left(\eta^2\right)^{(2L)}(0)}{(2L)!}}. \tag{7.58}$$

It has been derived in [75] in the following form:

$$C_\varphi = \frac{1}{L!}\sqrt{\sum_{k=-\infty}^{\infty} \left|\Phi^{(L)}(2\pi k)\right|^2}. \tag{7.59}$$

The last term in (7.57) can be neglected when Δ is sufficiently small. Hence, for *oversampled* signals the approximation order plays a rather important role, since for such signals the approximation error decreases by order L when Δ decreases and tends to zero:

$$e^2(\Delta) \propto \left(C_\varphi\right)^2 \Delta^{2L} \int_{-\infty}^{\infty} f^{2L} \left|G_a(2\pi f)\right|^2 df. \tag{7.60}$$

The integrand is the norm of the Lth derivative of $g_a(x)$ and the above equation is compacted to the form

$$e^2(\Delta) \propto \left(C_\varphi\right)^2 \Delta^{2L} \left\|g^{(L)}\right\|_{L_2}^2. \tag{7.61}$$

Note that no restrictions on $g_a(x)$, that is, no band-limitedness, have been imposed. If the sampling step is sufficiently small to "kill" the effect of the last term in (7.57), the reconstruction function can be characterized by a constant and by a rate of decay Δ^L determined by the approximation order. Knowing these two parameters, the true approximation error can be well predicted by (7.61).

Strang-Fix conditions. In their paper [59], Strang and Fix formulated sufficient conditions for a basis function to have an approximation order of L. Later, those conditions have been widely applied in the wavelet functions design [60], while Unser [64] and Blu and Unser [10] have stressed their importance for the interpolation.

Here, a simplified form of those conditions, in the context of our problem and in a more engineering setting, as in [62], is presented.

There are three equivalent conditions leading to a reconstruction function, ensuring an approximation error in the form of (7.61):

(1) Lth-order zeros in the Fourier domain

$$\begin{aligned} \Phi(0) &= 1, \\ D^l\Phi(2\pi k) &= 0, \quad k \in \mathbb{Z}\backslash\{0\}\ 0 \le l \le L-1. \end{aligned} \tag{7.62}$$

(2) Reproduction of all monomials of degree $n \le N = L-1$,

$$\forall n \in [0, N] \quad \exists\{d_n[k]\} \mid \sum_{k=-\infty}^{\infty} d_n[k]\varphi(x-k) = x^n. \tag{7.63}$$

(3) Discrete moments

$$\sum_{k=-\infty}^{\infty} (x-k)^n \varphi(x-k) = M_n, \quad 0 \le n \le L-1. \tag{7.64}$$

The first condition for zero derivative is known as *partition of unity condition*. It implies that $\sum_k \varphi(x-k) = 1$, and in particular says that a good basis should preserve the DC signal value. The second condition shows that the reproduction of polynomials has not only an intuitive meaning (smooth data are best represented by polynomials) but it is also implicitly related with the rate of decay of the approximation error: reconstruction functions having an approximation order L can reproduce *exactly* polynomials up to degree $L-1$.

Interpolation and least squares projection. Both the interpolation and projection cases involve Lth-order basis functions complying with the Strang-Fix conditions for which the approximation error is predicted by a quantity such as

$$e^2(\Delta) = (C_\varphi)^2 \Delta^{2L} \|g^{(L)}\|_{L_2}^2 + o(\Delta^L). \tag{7.65}$$

However, the constant for different cases is different. It is smallest for the orthogonal projection case. It is shown in [10] that the constant for the case of LS approximation by an Lth-order basis is given by

$$C_{\varphi,\mathrm{LS}} = \frac{1}{L!} \sqrt{\sum_{k\neq 0} |\Phi^{(L)}(2\pi k)|^2}. \tag{7.66}$$

When comparing (7.66) with (7.59) it is seen that the approximation constant for the interpolation case is composed of two parts as follows:

$$C_{\varphi} = \sqrt{C_{\varphi,\mathrm{LS}}^2 + \left| \frac{1}{L!}\Phi^{(L)}(0) \right|^2}. \tag{7.67}$$

Using (7.42) and (7.55) the error kernels for the interpolation and LS approximation are related as follows:

$$\eta_{\mathrm{Int}}^2(2\pi f) = \eta_{\mathrm{LS}}^2(2\pi f) + \left| \frac{\sqrt{\sum_{n=-\infty}^{\infty} |\Phi(2\pi(f-n))|^2}}{\sum_{n=-\infty}^{\infty} \Phi(2\pi(f-n))} - \frac{\Phi(2\pi f)}{\sqrt{\sum_{n=-\infty}^{\infty} |\Phi(2\pi(f-k))|^2}} \right|^2. \tag{7.68}$$

7.3. Piecewise polynomial basis functions of minimal support

In Section 7.2, we reviewed the most commonly used convolution-based interpolation models and analytical tools for evaluating the interpolation function performance. In this section, we demonstrate how to design and analyze piecewise polynomial interpolation functions obtained as linear combinations of B-splines. Recently, the use of such, in general *noninterpolating*, functions has been favored in the light of modern approximation theory. The approximation order has been adopted as a key index that predetermines the interpolation quality. What makes the B-splines preferable for interpolation is their efficiency, since they can achieve a desired approximation order L with a minimum support of L. Furthermore, the Lth-order spline-based interpolators have been constructed as combinations of a B-spline of degree $L-1$ with its derivatives. The resulting functions have been called *splines of minimal support* [53] or MOMS (maximal order minimal support) [9]. They have been optimized to have the smallest possible approximation constant in order to achieve minimal asymptotic approximation error. Functions of this class have been reported as superior for image interpolation [9, 63].

Another approach has also adopted the use of uniform B-splines of different degrees as linearly combined bases for forming piecewise polynomial interpolators. However, the weighting parameters in the combination have been tuned by an optimization mechanism inspired by the digital filter design rather than the approximation theory. The design aim has been to approximate the ideal band-limited interpolator in frequency domain by clustering the frequency response zeros around the multiples of the sampling rate ($2k\pi$ in terms of angular frequency). In contrast with the Lth-order interpolators where $L-1$ zeros are placed exactly at $2k\pi$, this approach sacrifices the multiplicity, distributing the zeros among the stopband to provide better stopband attenuation.

Further in this chapter, we demonstrate how to design and analyze piecewise polynomial basis functions of minimal support which can be used in interpolation

and projection tasks. The design is based on linear combinations of uniform B-splines of different degrees. We compare the basis kernels thus designed in terms of error kernel behavior and by interpolation experiments with real images.

7.3.1. Piecewise polynomial interpolators and B-splines

Piecewise polynomial (PP) interpolation functions have been preferred for their efficient realization and explicit time and frequency expressions [33, 40, 49, 56, 77]. In general, a certain interpolator $h(x)$ of this class is formed by polynomial pieces

$$P_k(x) = \sum_{l=0}^{M} a_k[l]x^l, \quad \xi_k \le x < \xi_{k+1}, \tag{7.69}$$

joined together at some break points $\xi_0, \xi_1, \ldots, \xi_N$ with ensured continuity [11]. At the break points, the continuity can be ensured by making the function continuous from the right for example, that is, $h(\xi_i) = h(\xi_i^+) = P_i(x)$. For the case of *interpolation at uniform grid*, the break points are taken as $\xi_{k+1} = \xi_k + 1$, then

$$h(x) = \sum_{k=0}^{N-1} P_k(x). \tag{7.70}$$

Therefore, a PP interpolator is characterized by its extent (number of polynomial pieces) N, the polynomial degree M, and by the constraints for the break points, such as the interpolation constraints, symmetry around the origin, continuity, and continuity of some derivatives. Most of the $(M + 1)N$ coefficients are used to meet those constraints. The degrees of freedom (independent coefficients) left are used to improve the interpolation accuracy [33, 40, 49, 56, 77]. The power form of the PP function (7.69), (7.70) is not appropriate for computations. It is preferable to decompose it into a particular basis. The *truncated power basis* [11] has been widely used [19, 63, 77]. Recently, Vesma and Saramäki have proposed a *symmetrical PP basis* that deals properly with desired symmetry around the origin and reduces the computations considerably, while also improving the design of PP-based interpolators [77].

The uniform B-splines of degree M are piecewise polynomials of degree M and have a compact support of $N = M + 1$ while being $K = M - 1$ times differentiable. Thus, they are the most regular piecewise polynomials with the given support [11, 58, 73]. The centered B-splines are defined as follows:

$$\beta^M(x) = \sum_{i=0}^{M+1} \frac{(-1)^i}{M!} \binom{M+1}{i} \left(x + \frac{M+1}{2} - i\right)_+^M, \tag{7.71}$$

where x_+^n is the truncated power function $\max(0, x)^n$ [58]. The B-spline functions can be obtained by repetitive convolutions:

$$\beta^M = \beta^0 * \beta^{M-1}, \tag{7.72}$$

where

$$\beta^0(x) = \begin{cases} 1, & \text{for } |x| \leq \dfrac{1}{2}, \\ 0, & \text{for } |x| > \dfrac{1}{2}, \end{cases} \tag{7.73}$$

is the B-spline of zero degree. The corresponding frequency domain characteristics are given by

$$B^M(2\pi f) = \left(\frac{\sin(\pi f)}{\pi f}\right)^{M+1}. \tag{7.74}$$

The integer shifts of the B-spline function of degree M form a Riesz basis for the spline space $V(\beta^M)$, that is, all polynomial splines of degree M with integer knot step can be represented as a linear combination of B-spline basis functions of the same degree as

$$s^M(x) = \sum_{k\in\mathbb{Z}} d[k]\beta^M(x-k). \tag{7.75}$$

Often shift-invariant spaces are called *spline-like spaces*. B-splines also comply with the Strang-Fix conditions, that means that the B-splines of degree M having an approximation order $L = M + 1$ can represent exactly polynomials up to the Lth order.

Figure 7.5 shows the first four members of the B-spline family, whereas Figure 7.6 shows a cubic spline expanded as a combination of shifted and weighted B-splines of third degree.

As a consequence of (7.72), the derivative of B-spline is a B-spline of a lower degree, that is,

$$D\beta^M(x) = \beta^{M-1}\left(x + \frac{1}{2}\right) - \beta^{M-1}\left(x - \frac{1}{2}\right). \tag{7.76}$$

Hence, the gradient of a spline signal $s^M(x)$ can be computed very easily using the B-spline expansion, as given by (7.75):

$$Ds^M(x) = \sum_k d[k]D\beta^M(x-k) = \sum_k (d[k] - d[k-1])\beta^{M-1}\left(x - k + \frac{1}{2}\right). \tag{7.77}$$

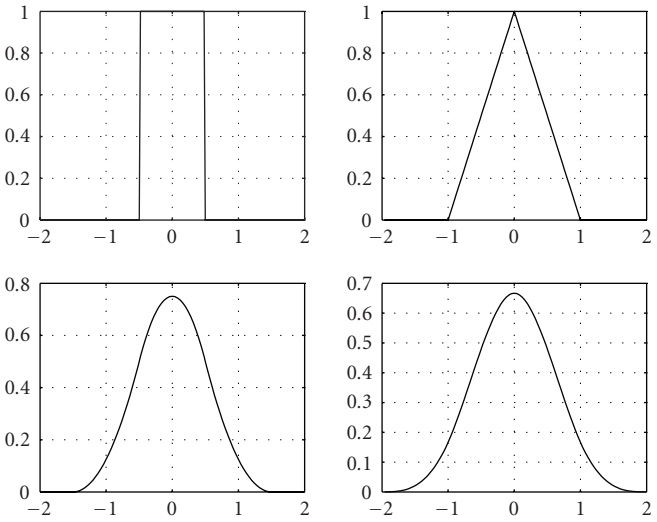


FIGURE 7.5. B-splines for $M = 0, 1, 2, 3$.

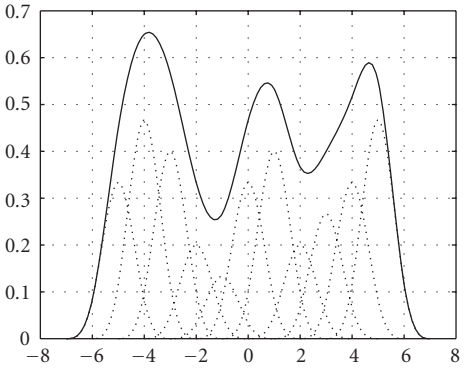


FIGURE 7.6. Cubic spline as B-spline expansion.

The convolution between two splines, $s^M(x)$ and $g^N(x)$ of degrees M and N, respectively, gives rise to a spline function of a higher degree $M + N + 1$ as follows [69, 70]:

$$\begin{aligned}(s^M * g^N)(x) &= \int_{-\infty}^{\infty} s^M(\xi) g^N(x-\xi)\, d\xi \\ &= \sum_{k=-\infty}^{\infty} (c * d)[k] \int_{-\infty}^{\infty} \beta^M(\xi)\beta^N(x-k-\xi)\, d\xi\end{aligned} \tag{7.78}$$

or

$$(s^M * g^N)(x) = \sum_{k=-\infty}^{\infty} (c * d)[k]\beta^{M+N+1}(x-k). \tag{7.79}$$

B-splines are strictly positive and for polynomial degrees $M > 1$ they do not go through zeros at integer coordinates. In order to use them for interpolation, they should be involved in the generalized interpolation model (7.9) as follows: the basis function in the model (7.9) is the B-spline of degree M itself $\varphi(x) = \beta^M(x)$, while the prefilter sequence is determined by the B-spline sampled version $p[k] = b^M[k] = \beta^M(k)$. The resulting interpolator frequency response is given by

$$H(2\pi f) = \frac{B^M(2\pi f)}{B^M(e^{j2\pi f})} = \frac{\left(\operatorname{sinc}(f)\right)^{M+1}}{b^M[0] + \sum_{k=1}^{\lfloor M/2 \rfloor} b^M[k]\cos(2\pi k f)}. \tag{7.80}$$

It specifies the impulse response of an *infinite* interpolator [69, 70] while being still computationally very efficient [72]. B-splines are also building blocks for designing optimized PP interpolation functions.

7.3.2. Piecewise polynomial basis functions of minimal support

We consider a subclass of PP basis functions complying with the following requirements.

(i) The functions are *noninterpolating* in the sense that they do not go through zeros for integer coordinates. This assumption favors the generalized interpolation model (7.9) instead of the classical model (7.7). We consider the latter as too restrictive for the design of efficient interpolators.

(ii) The functions are *symmetric* in order to preserve the phase relations.

(iii) The support M is chosen to be minimal, that is, $N = M + 1$, where M is the polynomial degree; thus the targeted approximation order is $L \leq N$.

These requirements target PP functions having an interpolation performance comparable and even better than the B-spline interpolation performance for the same degree M.

It is possible to construct the desired functions through the truncated power basis or through the symmetrical PP basis described in [77]. However, it has turned out that the B-splines are a better basis [18, 21, 22]. The required positivity and symmetry are their inherent features. The minimal support is ensured by taking a central B-spline of degree M and adding lower degree B-spline terms. Such a combination has explicit and easily manageable forms in both time and frequency domains and is very susceptible to optimization. The idea of constructing PP interpolators by combining B-splines of different degrees has appeared in different forms [8, 18, 53]. Engineers recognized the new construction as an opportunity to play with the additional degrees of freedom in a frequency domain specified optimization. Mathematicians recognized the construction as a convolution between a

B-spline and a distribution and were interested mainly in what the asymptotic behavior of the prospective approximating candidates is. Both approaches converged to a number of new optimized functions [9, 21, 39].

7.3.2.1. O-MOMS

The idea of combining *B*-splines of different degrees has appeared in a more formal setting in [8, 9, 53] with the aim to construct a basis function having an approximation order L and the *smallest possible* support. It turned out that such a function is merely a combination of the *B*-spline of degree $M = L - 1$ and its derivatives [9, 53]:

$$\varphi_L(x) = \beta^{L-1}(x) + \sum_{k=1}^{L-1} \lambda_k D^k \beta^{L-1}(x). \tag{7.81}$$

Blu et al. [9] have called this function class MOMS—maximal order of minimum support. It is characterized by the approximation order L, the (minimal) support of L, and the polynomial degree of $L - 1$. Putting all parameters λ_k equal to zero, we get the *B*-splines, which are the most regular functions in the class of MOMS. Further optimization can be achieved by adding some weighted derivative terms and tuning the weighting parameters. Derivatives of a *B*-spline are *B*-splines of lower degree and hence the construction (7.81) yields a PP interpolator of minimal support and of highest possible approximation order. Blu et al. [9] propose using the degrees of freedom to minimize the optimal approximation constant $C_{\varphi,\mathrm{LS}}$ (7.66). Their experiments have shown that this minimization gives better results than the global minimization of the constant C_φ (7.67). Minimization is carried out by an analytical method. The functions obtained are called O-MOMS (optimized MOMS). We give briefly their optimization results.

Transform the function (7.81) into Fourier domain

$$\Phi_L(2\pi f) = \Lambda(j2\pi f)\big(\operatorname{sinc}(f)\big)^L, \tag{7.82}$$

where $\Lambda(z) = \sum_{k=0}^{L-1} \lambda_k z^k$ is a polynomial of degree $L - 1$.

As a result of analytical optimization, the class of O-MOMS can be characterized by the following induction relation [9]:

$$\Lambda_{L+1}(z) = \Lambda_L(z) + \frac{z^2}{4\big(4L^2 - 1\big)} \Lambda_{L-1}(z) \tag{7.83}$$

with initial conditions $\Lambda_1(z) = \Lambda_2(z) = 1$. The minimal constant C_L is given by

$$C_L = \frac{L!}{(2L)!\sqrt{2L+1}}. \tag{7.84}$$

Several important observations are made in [9]: the polynomial in (7.83) is even with strictly positive coefficients of even powers. For L even, the highest polynomial degree is $L-2$. As a result, the function $\varphi_L(x)$ is *continuous* and has *a discontinuous first derivative*. For L odd, the highest polynomial degree is $L-1$. The resulting O-MOMS function $\varphi_L(x)$ is discontinuous. These observations indicate that regularity does not affect approximation accuracy. It is possible to construct interpolators which have good approximation properties but which nevertheless have low regularity and may even be discontinuous.

7.3.3. Modified B-splines

In this section, we present an alternative approach for obtaining optimized PP basis functions. We use B-splines as building blocks to design a kernel having a polynomial degree of M and a support of $N = M + 1$. The combination takes the following general form:

$$\beta^{\mathrm{mod}}(x) = \sum_{m=0}^{M} \sum_{n\in\mathbb{Z}} \gamma_{mn}\beta^m(x-n). \tag{7.85}$$

In our earlier publications [21, 22, 39] we have been calling such a combination *modified B-spline function*, emphasizing the fact that it is formed by a central B-spline of degree M combined with additional B-spline terms of lower degrees. It is also a *basis* function for some convolution-based spline-like signal space. From a computational point of view it is a PP function of minimal support, where the weighting coefficients γ_{mn} are optimized in a proper manner [18].

It is preferable to combine B-splines of *odd* degrees since they have knots at the integers, while the even degree B-splines have knots at half-integers and require an extra shift by half when making calculations.

We have studied extensively the following two cases: combining third and first degree B-splines and combining fifth- with third- and first-degree B-splines. This combination game can proceed further with higher degrees and with more parameters to be tuned. We have restricted it to these low-degree cases mainly for computational reasons. Moreover, when the B-spline degree is high enough, that is, higher than five, the performance is already sufficient for most of the image processing applications [62]. The optimization plays a substantial role in improving the performance precisely in the low-degree cases.

7.3.3.1. Third-degree combination

In this case, the modified B-spline basic function becomes

$$\beta^{\mathrm{mod}}(x) = \beta^{(3,1)}(x) = \beta^3(x) + \gamma_{10}\beta^1(x) + \gamma_{11}\beta^1(x+1) + \gamma_{11}\beta^1(x-1). \tag{7.86}$$

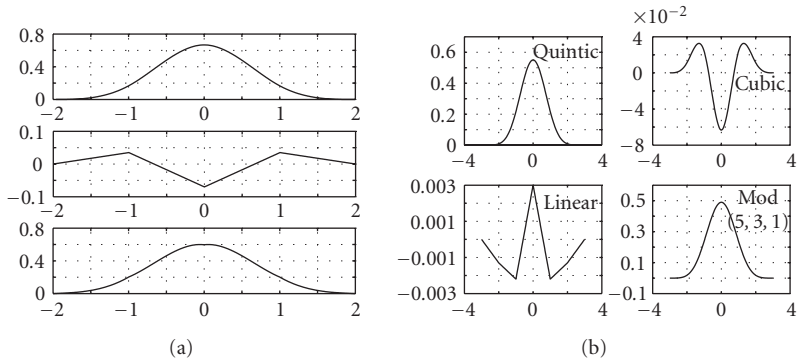


FIGURE 7.7. PP kernels obtained as a combination of B-splines. (a) third- and first-degree combinations. From top to bottom: cubic term, linear term, resulting kernel. (b) fifth-, third-, and first-degree combinations. From top to bottom and from left to right: quintic, cubic, linear terms, and resulting kernel.

This combination is of particular interest since the cubic term is smooth enough and has short support. We denote it as (3,1) showing the degrees of B-splines involved. Cubic convolution [33] and cubic spline [30] have been considered for a long time as the best cost/performance compromise for image interpolation. By adding linear terms, we aim at improving the performance for the same cost. The preservation of the partition of unity condition implies $\gamma_{10} = -2\gamma_{11}$ and (7.86) takes the form

$$\beta^{(3,1)}(x) = \beta^3(x) + \gamma_{11}[\beta^1(x-1) - 2\beta^1(x) + \beta^1(x+1)]. \tag{7.87}$$

It can be regarded as a combination of a smoothing function $\beta^3(x)$ and a difference part of zero average $(\beta^1(x-1) - 2\beta^1(x) + \beta^1(x+1))$ or, more broadly, a combination of a scaling function and a wavelet. The properly adjusted parameter γ_{11} gives the trade-off between smoothing and sharpening. Figure 7.7(a) shows the combination between a centered B-spline of third degree and centered and shifted B-splines of first degree.

The frequency response of the modified function is represented as

$$B^{(3,1)}(2\pi f) = \left(\frac{\sin(\pi f)}{\pi f}\right)^4 - 2\gamma_{11}(1-\cos(2\pi f))\left(\frac{\sin(\pi f)}{\pi f}\right)^2. \tag{7.88}$$

The sampled version $p^{(3,1)}[k] = \beta^{(3,1)}(x)|_{x=k}$ has the following z-transform:

$$P^{(3,1)}(z) = \left(\frac{4}{6} - 2\gamma_{11}\right) + \left(\frac{1}{6} + \gamma_{11}\right)(z + z^{-1}). \tag{7.89}$$

Finally, the frequency response of the interpolating function is obtained by

$$\begin{aligned} P^{(3,1)}(z)\big|_{z=e^{j2\pi f}} &= P^{(3,1)}\left(e^{j2\pi f}\right), \\ H^{(3,1)}(2\pi f) &= \frac{B^{(3,1)}(2\pi f)}{P^{(3,1)}\left(e^{j2\pi f}\right)}. \end{aligned} \tag{7.90}$$

7.3.3.2. Fifth-degree combination

We combine quintic, cubic, and linear B-spline in a combination (5,3,1). The modified B-spline takes the form

$$\begin{aligned} \beta^{(5,3,1)}(x) = {} & \beta^5(x) + \gamma_{30}\beta^3(x) + \gamma_{31}\left(\beta^3(x-1) + \beta^3(x+1)\right) \\ & + \gamma_{10}\beta^1(x) + \gamma_{11}\left(\beta^1(x-1) + \beta^1(x+1)\right) \\ & + \gamma_{12}\left(\beta^1(x-2) + \beta^1(x+2)\right). \end{aligned} \tag{7.91}$$

The kernel, as combined by quintic, cubic, and linear polynomial pieces, is shown in Figure 7.7(b). In the frequency domain it is represented as

$$\begin{aligned} B^{(5,3,1)}(2\pi f) = {} & \left(\frac{\sin(\pi f)}{\pi f}\right)^6 + \left(\gamma_{30} + 2\gamma_{31}\cos(2\pi f)\right)\left(\frac{\sin(\pi f)}{\pi f}\right)^4 \\ & + \left(\gamma_{10} + 2\gamma_{11}\cos(2\pi f) + 2\gamma_{12}\cos(4\pi f)\right)\left(\frac{\sin(\pi f)}{\pi f}\right)^2, \end{aligned} \tag{7.92}$$

while the sampled version $p^{(5,3,1)}[k] = \beta^{(5,3,1)}(x)|_{x=k}$ has the z-transform

$$\begin{aligned} P^{(5,3,1)}(z) = {} & \left(\frac{66}{120} + \frac{4}{6}\gamma_{30} + \frac{2}{6}\gamma_{31} + \gamma_{10}\right) + \left(\frac{26}{120} + \frac{1}{6}\gamma_{30} + \frac{4}{6}\gamma_{31} + \gamma_{11}\right)\left(z + z^{-1}\right) \\ & + \left(\frac{1}{120} + \frac{1}{6}\gamma_{31} + \gamma_{12}\right)\left(z^2 + z^{-2}\right). \end{aligned} \tag{7.93}$$

The frequency response of the interpolating function is obtained by

$$\begin{aligned} P^{(5,3,1)}(z)\big|_{z=e^{j2\pi f}} &= P^{(5,3,1)}\left(e^{j2\pi f}\right), \\ H^{(5,3,1)}(2\pi f) &= \frac{B^{(5,3,1)}(2\pi f)}{P^{(5,3,1)}\left(e^{j2\pi f}\right)}. \end{aligned} \tag{7.94}$$

The partition of unity condition is ensured by $\gamma_{30} + 2\gamma_{31} + \gamma_{10} + 2\gamma_{11} + 2\gamma_{12} = 1$.

7.3.3.3. Optimization

Consider an interpolation with a resampling step $\Delta < 1$. The interpolation process generates frequency components which are mirrors of the passband frequency region with respect to $2\pi/\Delta$. With some preliminary knowledge about the frequency

characteristics of $g(x)$, the interpolator $h(x) = (p^{-1} * \varphi)(x)$ can be designed in the continuous frequency domain in such a way that it suppresses effectively the undesired components of the interpolated signal (see Figure 7.3).

Our optimizing approach considers the possibility to express the edges (sharp transition in images) as a certain amount of *important high-frequency* components in the frequency characteristics of $g_a(x)$. Their mirrors in the stopband have to be suppressed. Suppression efficiency has to compromise with the flatness of the passband region of the interpolator's frequency response. It is useful to represent the frequency axis $F \subset [0, \infty)$ as consisting of disjoint regions: one passband and multiple stopbands.

We assume that most of the important signal components (frequencies) are in the frequency band $|f| \leq \alpha$ with $\alpha < 1/2$. The *passband region* is determined by

$$F_p = [0, \alpha] \tag{7.95}$$

and the *stopband regions* are determined by

$$F_s = \bigcup_{r=1}^{\infty} [r - \alpha, r + \alpha]. \tag{7.96}$$

We design $H(2\pi f) = \Phi(2\pi f)/P(e^{j2\pi f})$ to preserve the signal in the passband and to attenuate the undesired frequencies in the stopbands.

The optimization problem can be defined as follows [77].

Given the basis function support N, the polynomial degree M, and the fraction of the Nyquist frequency α; specify $D(2\pi f) = 1$ and $W(2\pi f) = w$ for $f \in F_p$, and $D(2\pi f) = 0$ and $W(2\pi f) = 1$ for $f \in F_s$, and find the unknown coefficients γ_{mn} to *minimize*

$$\delta_\infty = \max_{f \in F} \left| W(2\pi f)\left[H(2\pi f) - D(2\pi f)\right] \right|, \tag{7.97}$$

where $F = F_p \cup F_s$, in such a way that preserves the partition of unity condition. The function $D(2\pi f) \equiv H_{\text{ideal}}(2\pi f)$ is recognized as the desired impulse response, while the function $W(2\pi f)$ is a positive weighting function, specifying the tolerance for the desired function. As defined in (7.97), the optimized function $H(2\pi f)$ is a *minimax* approximation to the desired continuous function $D(2\pi f)$, that is, an approximation minimizing the L_∞ error norm [55]. The reason for choosing the minimax instead of least squares (L_2 error norm) minimization is twofold.

(1) Most of the images are oversampled and the information content is concentrated in the low frequencies. We are not worried too much about the amount of energy that can be transferred during the process of resampling from aliasing components that are close to the Nyquist frequency. Correspondingly, we disregard the requirement of minimal aliased energy, usually met by the LS optimization.

(2) We would like to treat the passband and stopbands *uniformly*. The uniform treatment is provided by the minimax optimization.

Table 7.1. SNRs for different optimized PP kernels obtained as combinations of third- + first- degree B-splines.

Mod B-spline (3,1)		SNR after successive rotations				
2α	$-\gamma_{01}$	Mandrill256 $10 \times 36°$	Geom2.256 $10 \times 36°$	Barbara512 $15 \times 24°$	Mandrill512 $15 \times 24°$	Lena512 $15 \times 24°$
0.5	0.0730	26.37	22.46	26.37	25.29	34.52
0.6; $w = .2$	0.0610	26.13	22.19	25.75	24.98	34.34
0.6; $w = .5$	0.0686	26.33	22.41	26.28	25.23	34.54
0.6; $w = 1$	0.0748	26.36	22.40	26.30	25.26	34.45
0.6; $w = 2$	0.0818	26.06	21.93	25.47	24.86	33.86
0.6; $w = 5$	0.0848	25.74	21.30	24.76	24.47	33.39
0.7; $w = .2$	0.0714	26.37	23.93	27.57	25.66	34.32
0.7; $w = .5$	0.8140	26.09	21.98	25.54	24.90	33.91
0.7; $w = 1$	0.0892	24.95	20.59	23.39	23.58	32.40
0.7; $w = 2$	0.0970	22.35	17.92	20.18	20.94	29.72
0.7; $w = 5$	0.1004	20.68	16.32	18.55	19.63	28.16
0.8 $w = .2$	0.0848	25.74	21.30	24.76	24.47	33.39
0.8; $w = .5$	0.0980	21.90	17.47	19.71	20.50	29.29
O-MOMS (3,1)	0.0476	25.67	23.14	25.19	24.63	33.60
B-spline (3)	0.0000	24.45	21.71	22.05	22.90	31.66

In numerical work, we take a sampled grid over F and convert the continuously defined optimization task to a multiobjective goal attainment problem solvable by the algorithm of Gembicki [20], as realized in the Matlab function `fgoalattain` [47].

We have sampled both the passband and first four stopbands by 500 equidistant points and proved experimentally that this grid is dense enough for our problem.

Third-degree and fifth-degree families of PP interpolators in the form of modified B-splines have been optimized for $M = 3$ and $M = 5$ for various values of α and various passband weights w. The optimized parameters obtained are listed in Tables 7.1 and 7.2.

7.3.3.4. Efficient implementation

The B-spline representation of a PP interpolator provides a simple frequency domain form which is appropriate for optimal design. As far as efficient implementation is concerned, it is advisable to transform the B-spline representation to the truncated power basis representation. It is especially suitable for the convolution-based interpolation models (7.7) and (7.9).

Table 7.2. Designs of PP interpolators of fifth degree.

Design no.	2α	w	γ_{10}	γ_{11}	γ_{12}	γ_{30}	γ_{31}
1	0.7	10	−0.0001	−0.0046	−0.0023	−0.1361	0.0750
2	0.7	Sqrt(10)	−0.0034	−0.0064	−0.0033	−0.133	0.0779
3	0.7	1	−0.0017	−0.0046	−0.0024	−0.1325	0.0741
4	0.7	Sqrt(0.1)	0.0005	−0.0026	−0.0013	−0.1323	0.0698
5	0.7	0.1	0.0003	−0.0022	−0.0013	−0.1289	0.0678
6	0.8	10	−0.0072	−0.0127	−0.006	−0.141	0.0928
7	0.8	Sqrt(10)	−0.0137	−0.0164	−0.0072	−0.1317	0.0963
8	0.8	1	−0.0093	−0.0119	−0.0055	−0.1347	0.0894
9	0.8	Sqrt(0.1)	−0.0038	−0.0077	−0.0036	−0.143	0.0847
10	0.8	0.1	−0.0005	−0.0085	−0.0024	−0.1633	0.0928
11	0.9	10	−0.0023	−0.0208	−0.0101	−0.1677	0.1159
12	0.9	Sqrt(10)	−0.0091	−0.0216	−0.0127	−0.1461	0.1119
13	0.9	1	−0.0492	−0.0438	−0.0173	−0.0918	0.1316
14	0.9	Sqrt(0.1)	−0.0335	−0.0341	−0.0127	−0.1237	0.1254
15	0.9	0.1	−0.0254	−0.0323	−0.0095	−0.156	0.1325
fifth-degree B-spline			0.0000	0.000	0.0000	0.0000	0.0000
O-MOMS (5,3,1)			0.00076	−0.0005	0.00013	−0.0606	0.0303

The truncated function is defined as

$$(x)_+ := \max\{0, x\}. \tag{7.98}$$

Further, the truncated power function is defined as

$$(x)_+^r := (x_+)^r, \quad r = 0, 1, 2, \ldots. \tag{7.99}$$

The shifted function $(x - \xi)_+^r$ is a piecewise polynomial of order $r + 1$ having $r - 1$ continuous derivatives and a jump in the rth derivative across ξ of size $r!$. To define the function for all values of r and x, that is, $x = 0$, we assume $0^0 = 0$. With this assumption, $(x - \xi)_+^r$ is a PP function for $r = 0$. For a given break sequence $\xi_0, \xi_1, \ldots, \xi_N$ we form the functions

$$\rho_{km}(x) := \begin{cases} (x - \xi_0)^m, & k = 0, \\ (x - \xi_k)_+^m, & k = 1, \ldots, N - 1, \end{cases} \tag{7.100}$$

for $m = 0, \ldots, M - 1$. They are linearly independent and they span the space of all piecewise polynomials of order M having breaks at $\xi_0, \xi_1, \ldots, \xi_N$. Consequently,

they form a basis and any PP function $h(x)$ from this space can be expressed as [11]:

$$h(x) = \sum_{k=0}^{N-1} \sum_{m=0}^{M} c_m[k]\rho_{km}(x), \tag{7.101}$$

where

$$c_m[k] = \begin{cases} \dfrac{D^m h(\xi_0)}{m!}, & k = 0, \\ \dfrac{\text{jump}_{\xi_k} D^m h(x)}{m!}, & k = 1, \ldots, N-1. \end{cases} \tag{7.102}$$

Now, the PP interpolation function is expressed as a sum of new truncated polynomial pieces, each represented in terms of a truncated power basis. For break points placed at the integers it is

$$h(x) = \sum_{k=0}^{N-1} Q_k(x-k), \tag{7.103}$$

where

$$Q_k(x) = \sum_{m=0}^{M} c_m[k] x_+^m. \tag{7.104}$$

The sequences $c_m[k]$, for $k = 0, \ldots, N-1$, can be considered as a bank of $M+1$ FIR filters having z-transforms

$$C_m(z) = \sum_{k=0}^{N-1} c_m[k] z^{-k} \tag{7.105}$$

and the corresponding continuous Fourier transform is obtained for $z = e^{-j2\pi f}$.

The interpolator expressed in the form of (7.103), (7.104) can be easily implemented, see (7.18). To interpolate at the new point $x_l = x_l + \mu_l$, we need $h(k+\mu_l)$, that is

$$h(k+\mu_l) = \sum_{m=0}^{M} c_m[k] \mu_l^m. \tag{7.106}$$

Substituting $h(k+\mu_l)$ in (7.18) and changing the order of summations give

$$y[l] = \sum_{k=0}^{N-1} g[n_l - k] \sum_{m=0}^{M} c_m[k] \mu_l^m = \sum_{m=0}^{M} v_m[n_l] \mu_l^m, \tag{7.107}$$

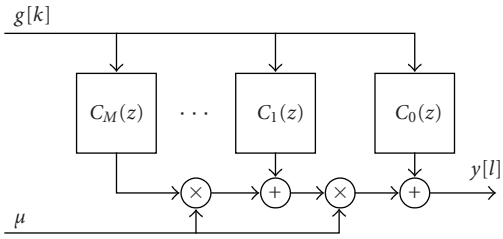


FIGURE 7.8. Farrow structure for PP interpolator.

where

$$v_m[n_l] = \sum_{k=0}^{N-1} g[n_l - k]c_m[k] \tag{7.108}$$

are the outputs of the FIR filters (7.105). The corresponding filter structure is known as *Farrow structure*[19] and is shown in Figure 7.8. This structure, originally proposed by Hou and Andrews for the case of B-splines [30], has been rediscovered and generalized as a form for efficient realizations of piecewise polynomial approximations. For M-degree polynomial it consists of $M+1$ parallel FIR branch filters with fixed coefficients.

The interpolant point can be obtained by multiplying the output of the mth FIR filter by μ^{m-1}. The only value that has to be loaded to the interpolator is the fractional interval μ_l.

The B-splines are naturally expressed as PP functions in the truncated power basis (cf. (7.71)). Adopting the Farrow structure notations, the coefficients $c_m[k]$ for the B-spline of degree M and support $N = M + 1$ are expressed as

$$c_m[k] = \sum_{i=-l}^{N/2} \frac{(-1)^{N/2-i} N!(i+k)^{M-m}}{(N/2+i)!(N/2-i)!(M-m)!m!}. \tag{7.109}$$

The Farrow structure is essentially the same, except the input sequence, which is formed by the model coefficients $d[k]$ instead of the initial samples $g[k]$.

The Farrow structure operation is clearly presented through the matrix notations. Using (7.108) and (7.109), for the case of cubic B-spline, where $N = 4$, $M = 3$, the interpolated sample $y(l)$ is determined as [30]

$$y[l] = y_a(n_l + \mu_l) = \frac{1}{6}\begin{bmatrix}\mu_l^3 & \mu_l^2 & \mu_l^1 & 1\end{bmatrix} \times \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} d[n_l - 2] \\ d[n_l - 1] \\ d[n_l] \\ d[n_l + 1] \end{bmatrix}. \tag{7.110}$$

The above equation clearly shows that a single interpolated sample is obtained by the local weighting of the neighbor polynomial pieces. The rows of the fixed matrix represent the FIR filter branches of the Farrow structure. Note that not the input signal $g[k]$ itself, but rather the model coefficients $d[k]$ are filtered. When the interpolation point occurs in the next integer interval between samples on the input grid, that is, $n = n + 1$, the model coefficient on the top $d[n_l - 2]$ drops out and the next coefficient $d[n_l + 2]$ enters from below, as in a conveyer.

For the modified case of B-splines of third and first degrees, the matrix is still fixed, though with different coefficients,

$$y[l] = \frac{1}{6}\begin{bmatrix}\mu_l^3 & \mu_l^2 & \mu_l^1 & 1\end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3-6\gamma_{11} & -6\gamma_{10}+6\gamma_{11} & 3+6\gamma_{10}-6\gamma_{11} & -6\gamma_{11} \\ 1+6\gamma_{11} & 4+6\gamma_{10} & 1+6\gamma_{11} & 0 \end{bmatrix} \begin{bmatrix} d[n_l-2] \\ d[n_l-1] \\ d[n_l] \\ d[n_l+1] \end{bmatrix}. \tag{7.111}$$

For the case of B-spline of fifth degree the matrix equation has the form

$$y[l] = \frac{1}{120}\begin{bmatrix}\mu_l^5 & \mu_l^4 & \mu_l^3 & \mu_l^2 & \mu_l^1 & \mu_l^0\end{bmatrix} \cdot \begin{bmatrix} -1 & 5 & -10 & 10 & -5 & 1 \\ 5 & -20 & 30 & -20 & 5 & 0 \\ -10 & 20 & 0 & -20 & 10 & 0 \\ 10 & 20 & -60 & 20 & 10 & 0 \\ -5 & -50 & 0 & 50 & 5 & 0 \\ 1 & 26 & 66 & 26 & 1 & 0 \end{bmatrix} \begin{bmatrix} d[n_l-3] \\ d[n_l-2] \\ d[n_l-1] \\ d[n_l] \\ d[n_l+1] \\ d[n_l+2] \end{bmatrix}. \tag{7.112}$$

For the case of combination of fifth-, third-, and first-degree B-splines, the values of the fixed matrix change. In fact, the first two rows, corresponding to the filters $c_0[k]$ and $c_1[k]$, are the same, while the last four rows have been modified by

the parameters γ's. Below, the filters are presented in the transposed form

$$\mathbf{C}^{(5,3,1)} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \\ \mathbf{c}_4 \\ \mathbf{c}_5 \end{bmatrix} \quad \text{where } \mathbf{c}_0^T = \begin{bmatrix} -1 \\ 5 \\ -10 \\ 10 \\ -5 \\ 1 \end{bmatrix}, \qquad \mathbf{c}_1^T = \begin{bmatrix} 5 \\ -20 \\ 30 \\ -20 \\ 5 \\ 0 \end{bmatrix},$$

$$\mathbf{c}_2^T = \begin{bmatrix} -10 - 20\gamma_{31} \\ 20 - 20\gamma_{30} + 60\gamma_{31} \\ -80\gamma_{31} + 60\gamma_{30} \\ -20 + 80\gamma_{31} - 60\gamma_{30} \\ 10 - 60\gamma_{31} + 20\gamma_{30} \\ 20\gamma_{31} \end{bmatrix}, \qquad \mathbf{c}_3^T = \begin{bmatrix} 10 + 60\gamma_{31} \\ 20 - 120\gamma_{31} + 60\gamma_{30} \\ -60 + 120\gamma_{31} - 120\gamma_{30} \\ 20 - 120\gamma_{31} + 60\gamma_{30} \\ 10 + 60\gamma_{31} \\ 0 \end{bmatrix},$$

$$\mathbf{c}_4^T = \begin{bmatrix} -5 - 60\gamma_{31} - 120\gamma_{12} \\ -50 + 120\gamma_{12} - 120\gamma_{11} - 60\gamma_{30} \\ -120\gamma_{10} + 120\gamma_{11} \\ 50 + 120\gamma_{10} + 60\gamma_{30} - 120\gamma_{11} \\ 5 - 120\gamma_{12} + 60\gamma_{31} + 120\gamma_{11} \\ 120\gamma_{12} \end{bmatrix}, \qquad \mathbf{c}_5^T = \begin{bmatrix} 1 + 20\gamma_{31} + 120\gamma_{12} \\ 26 + 80\gamma_{31} + 120\gamma_{11} + 20\gamma_{30} \\ 66 + 80\gamma_{30} + 40\gamma_{31} + 120\gamma_{10} \\ 26 + 20\gamma_{30} + 80\gamma_{31} + 120\gamma_{11} \\ 1 + 20\gamma_{31} + 120\gamma_{12} \\ 0 \end{bmatrix}. \tag{7.113}$$

7.3.4. Comparisons

Some elaborate surveys [35, 41] have compared a large number of PP-based and sinc-based image interpolators. Thirty two kernels have been studied in [35], while 126 kernels have been studied in [41]. The experiments included geometrical forward and backward transformations: subpixel translation, rotation, and rescaling over a large number of images of different kinds, including digital photographs and biomedical images. The reported results in [35, 41] show the superiority of B-spline basis functions in a cost-performance sense. In addition, Thévenaz et al. [62, 63] have analyzed a number of convolution-based interpolation kernels and favored the class of MOMS from an approximation theory point of view. They stressed the importance of the approximation order and approximation constant, and corroborated their theoretical considerations by involving the investigated

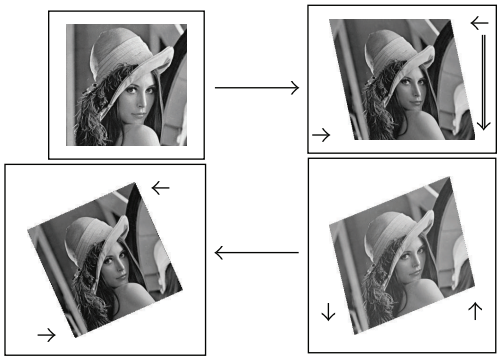


FIGURE 7.9. Rotation as three successive translations [76].

kernels in repetitive rotations experiments. In the rest of this chapter we will compare our optimized designs with the O-MOMS and B-splines, as functions belonging to the class of minimally supported PP basis functions.

7.3.4.1. The rotation experiment

We have adopted the experiment with successive image rotations [76]. A test image is rotated to a certain angle which is the result of an integer division of 360°, for example, $24° = 360°/15$, $36° = 360°/10$, and so forth. The rotation is repeated until the image reaches its starting position. Consider the matrix

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{7.114}$$

It determines a rotation of the image coordinates (x, y) by the angle θ. The matrix factorization

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & -\tan\left(\frac{\theta}{2}\right) \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -\tan\left(\frac{\theta}{2}\right) \\ 0 & 1 \end{bmatrix} \tag{7.115}$$

decomposes the rotation into three 1D translations, as illustrated in Figure 7.9 [76]. Each subpixel translation is an interpolation operation with no rescaling. The repetitive rotation experiment is very appropriate for accumulating the interpolation errors, thereby emphasizing the interpolator's performance [62].

Several well-known test images have been chosen because of their different frequency contents. Among them, the *Barbara*, *mandrill*, and *patterns* images that contain high-frequency and texture details, the *geometrical* image contains different radial frequencies, the *Lena* image is more like a low-frequency image, and the *bridge* contains some straight edges (Figure 7.10).

FIGURE 7.10. Test images with different frequency contents.

On the chosen set of images, we have performed 10 and 15 successive rotations of 36° and 24°, respectively, involving PP optimized interpolators, *B*-splines and O-MOMS of third and fifth degrees. As a measure of goodness, we have used the SNR between the initial image and the rotated one. Table 7.1 summarizes the results for the third-degree kernels, while Table 7.3 summarizes the results for the fifth-degree kernels. One can see that significant improvements are provided by some of the proposed modified *B*-splines for the relatively high-frequency image *Barbara*. Even improvements of more than 4 dB are achieved, compared to the classical fifth-degree *B*-spline and the O-MOMS. This argues the applicability of the chosen optimization technique, especially for high-frequency images. For relatively low-frequency images (like *Lena*), noticeable improvements can be achieved.

We illustrate the improvement of interpolation quality by two visual examples. A portion of the *Barbara* image, namely, the high-frequency trousers (Figure 7.11) is shown after $15 \times 24°$ rotations using three different PP interpolators of third degree. As can be seen in Figure 7.12 the most blurred image is the one obtained by the cubic *B*-spline. Our modified spline (3,1) gives the superior visual quality. To magnify the effect, the same images are enlarged by factor of two in Figure 7.13.

The second example is the central part of the *geometrical* image (Figure 7.14) containing high radial frequencies. The images, resulting from 15 successive rotations of 24° and by using three interpolation kernels of third degree, are shown in Figure 7.15, whereas Figure 7.16 shows the same images enlarged by factor of two. Again, more details are preserved in the image rotated using the Mod *B*-spline (3,1).

Table 7.3. SNRs for optimized PP kernels of fifth degree in Table 7.2 after rotation $10 \times 36°$.

Design no.	Bridge	Barbara	Geometrical	Lena
1	29.54	32.25	24.32	36.21
2	29.65	32.67	24.43	36.26
3	29.60	32.49	24.38	36.23
4	29.52	32.22	24.30	36.19
5	29.49	32.15	24.30	36.17
6	29.80	33.08	24.54	36.32
7	29.82	33.35	24.65	36.22
8	29.82	33.34	24.60	36.28
9	29.80	33.26	24.53	36.32
10	29.81	33.29	24.52	36.37
11	29.20	30.15	24.05	35.81
12	29.72	31.57	24.33	36.10
13	29.20	32.79	24.76	34.67
14	29.58	33.44	24.82	35.43
15	29.50	33.07	24.78	35.51
B-spline (5)	28.02	27.89	23.37	34.98
O-MOMS (5,3,1)	28.47	29.05	23.69	35.38

Figure 7.11. Portion of *Barbara* image: trousers (zoomed out).

Table 7.4 shows comparisons of different spline-based interpolations for four rotation experiments. Barbara image has been rotated clockwise 15 times by 24° and 10 times by 36°. The same successive rotations have been accomplished then backwards to allow for accumulating more interpolation errors. The differences between the original image and the rotated ones have been measured by peak signal-to-noise ratio (PSNR), signal-to-noise ratio (SNR), mean absolute error (MAE), mean square error, and maximum difference. The experiments have involved the interpolation kernels described in Section 7.3.4.2. The results in the

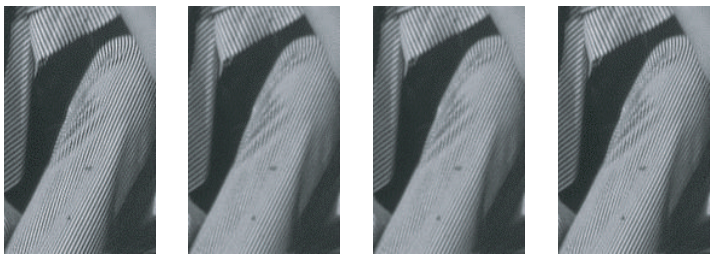


FIGURE 7.12. Results of successive rotations, $15 \times 24°$. From left to right: original image, rotated using cubic B-spline, rotated using O-MOMS (3,1), rotated using Mod B-spline (3,1).

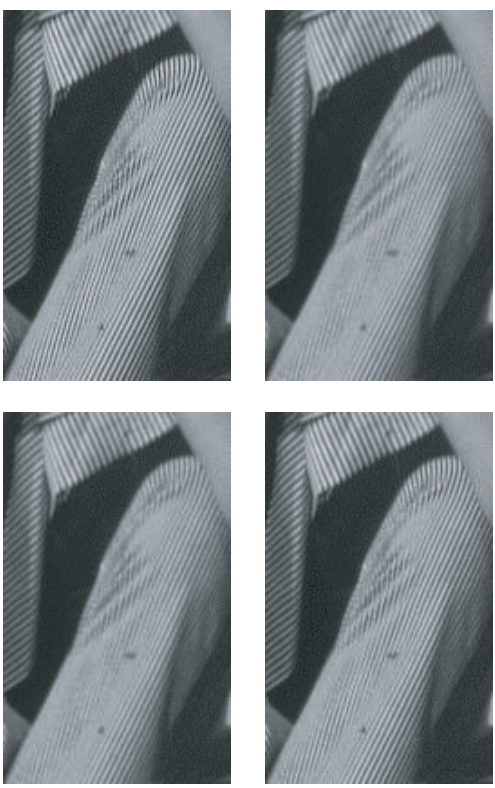


FIGURE 7.13. Same as in Figure 7.11 magnified by a factor of two. From left to right and from top to bottom: original, cubic B-spline rotated, O-MOMS (3,1) rotated, Mod B-spline (3,1) rotated.

table once again confirm the superior performance of the properly optimized kernels. Note, that the modified B-spline kernel of third degree already achieves better performance than some of the higher degree kernels.

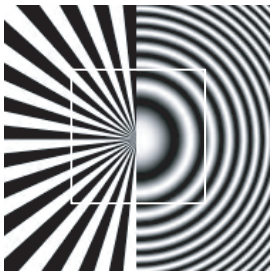


FIGURE 7.14. Portion of *geometrical* image: center (zoomed out).

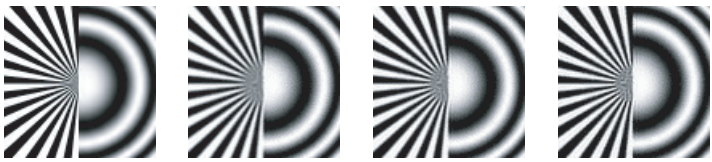


FIGURE 7.15. Results of successive rotations, $15 \times 24^{\circ}$. From left to right: original image, rotated using cubic B-spline, rotated using O-MOMS (3,1), rotated using Mod B-spline (3,1).

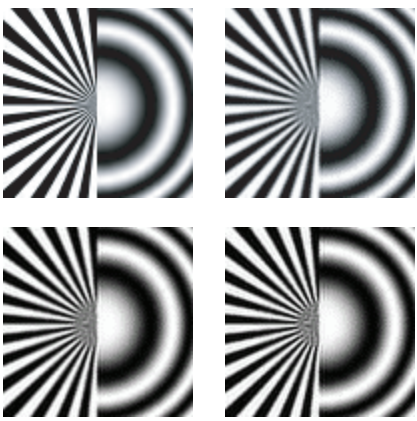


FIGURE 7.16. Same as in Figure 7.14 magnified by factor of two. From left to right and from top to bottom: original image, cubic B-spline rotated, O-MOMS (3,1) rotated, Mod B-spline (3,1) rotated.

7.3.4.2. Approximation error kernel

The performance of different interpolators can be compared theoretically based on the representation of the approximation error (7.42), (7.54) in frequency domain [62].

We present a comparison between two modified B-spline-based optimized kernels, of third and fifth degrees, respectively, and their O-MOMS counterparts.

TABLE 7.4. Trend of some quality measures for different spline-based interpolators. *Barbara* (512×512) image processed.

Interpolator	Rotations 15 × 24° clockwise				
	PSNR, [dB]	SNR, [dB]	MAE L_1 norm	MSE L_2 norm	MaxDiff L_∞ norm
B-spline (3)	27.02	22.07	5.5717	0.0276	70
O-MOMS (3,1)	30.20	25.25	3.6765	0.0191	58
Mod *B*-spline (3,1)	32.64	27.70	2.9757	0.0144	50
B-spline (5)	31.19	26.25	3.3074	0.0171	50
O-MOMS (5,3,1)	32.41	27.89	2.8853	0.0148	46
Mod *B*-spline (5,3,1)	36.47	31.53	1.9769	0.0093	28
Interpolator	Rotations 15 × 24° clockwise and back				
	PSNR, [dB]	SNR, [dB]	MAE L_1 norm	MSE L_2 norm	MaxDiff L_∞ norm
B-spline (3)	25.07	20.12	7.1015	0.0345	77
O-MOMS (3,1)	28.23	23.28	4.5958	0.0240	67
Mod *B*-spline (3,1)	30.40	25.46	3.9859	0.0019	60
B-spline (5)	28.98	24.04	4.2307	0.0220	60
O-MOMS (5,3,1)	30.21	25.27	3.6552	0.0191	55
Mod *B*-spline (5,3,1)	34.14	29.20	2.5497	0.0121	34
Interpolator	Rotations 10 × 36° clockwise				
	PSNR, [dB]	SNR, [dB]	MAE L_1 norm	MSE L_2 norm	MaxDiff L_∞ norm
B-spline (3)	28.64	23.69	4.5812	0.0229	63
O-MOMS (3,1)	31.92	26.97	3.0177	0.0157	55
Mod *B*-spline (3,1)	34.03	29.08	2.5076	0.0123	49
B-spline (5)	32.92	27.97	2.7346	0.0140	49
O-MOMS (5,3,1)	34.11	29.16	2.4038	0.0122	44
Mod *B*-spline (5,3,1)	37.89	32.95	1.7016	0.0079	26
Interpolator	Rotations 10 × 36° clockwise and back				
	PSNR, [dB]	SNR, [dB]	MAE L_1 norm	MSE L_2 norm	MaxDiff L_∞ norm
B-spline (3)	26.30	21.36	6.0891	0.0300	74
O-MOMS (3,1)	29.76	24.82	3.8270	0.0201	69
Mod *B*-spline (3,1)	31.65	26.71	3.3366	0.0162	72
B-spline (5)	30.51	25.56	3.5545	0.0185	59
O-MOMS (5,3,1)	31.77	26.83	3.0745	0.0160	56
Mod *B*-spline (5,3,1)	35.81	30.87	2.1270	0.0100	31

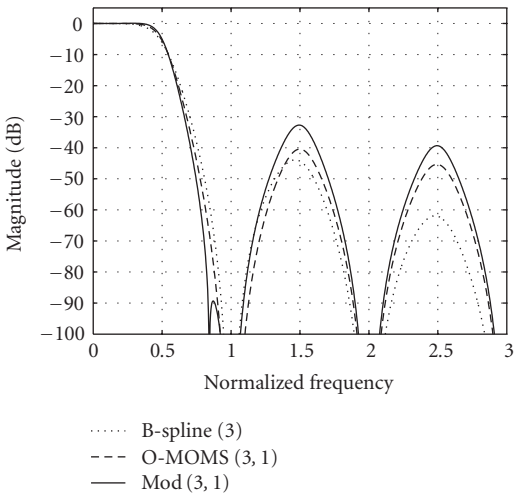


FIGURE 7.17. Magnitude responses of three PP interpolators of third degree.

The O-MOMS interpolation kernels of the form (7.81) have been optimized to have minimal interpolation constant [9]. This has yielded interpolator functions of the type of (7.81) as follows:

$$\begin{aligned}\beta^{\text{O-MOMS}(3,1)}(x) &= \beta^3(x) + \left(\frac{1}{42}\right)D^2\beta^3(x),\\ \beta^{\text{O-MOMS}(5,3,1)}(x) &= \beta^5(x) + \left(\frac{1}{33}\right)D^2\beta^5(x) + \left(\frac{1}{7920}\right)D^4\beta^5(x).\end{aligned} \tag{7.116}$$

Here, the abbreviations O-MOMS(3, 1) and O-MOMS(5, 3, 1) denote the degrees of the B-splines involved in the combination.

The modified kernels have been optimized to have a passband up to $\alpha = 0.35$ with maximal ripples of 0.1–0.2% and to suppress the frequencies in the stopband as much as possible. The optimization has resulted in the kernels

$$\begin{aligned}\beta^{\text{mod}(3,1)}(x) &= \beta^3(x) - 0.0714\beta^1(x) + 0.0357\big(\beta^1(x-1) + \beta^1(x+1)\big),\\ \beta^{\text{mod}(5,3,1)}(x) &= \beta^5(x) - 0.1289\beta^3(x) + 0.0003\beta^1(x)\\ &\quad + 0.0678\big(\beta^3(x-1) + \beta^3(x+1)\big)\\ &\quad - 0.0022\big(\beta^1(x-1) + \beta^1(x+1)\big)\\ &\quad - 0.0013\big(\beta^1(x-2) + \beta^1(x+2)\big).\end{aligned} \tag{7.117}$$

Again, the notations Mod B-spline (3,1) and Mod B-spline (5,3,1) stand for the degrees of the B-splines involved in the corresponding combinations.

Figure 7.17 compares the magnitude response of the optimized interpolation kernel of third degree (solid line), as of (7.90), with the O-MOMS (dashed line) and B-spline (dotted line) of same degree. It is well visible that the zeros of the

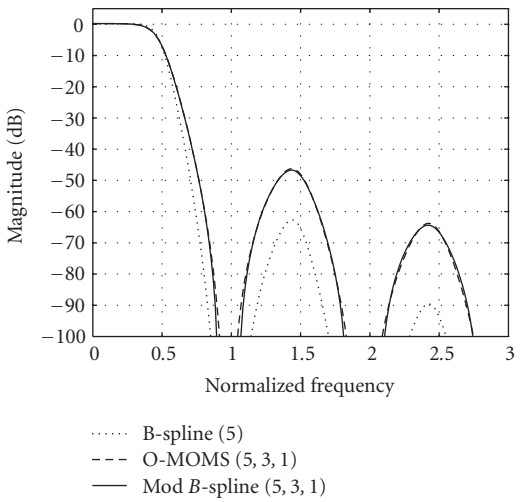


FIGURE 7.18. Magnitude responses of three PP interpolators of fifth degree.

modified kernel are not all located at the multiples of the sampling rate. This results in a better stopband attenuation.

The magnitude response of the fifth-degree modified B-spline interpolator (solid line), as of (7.94), is shown in Figure 7.18 together with O-MOMS (dashed line) and quintic B-spline (dotted line). Here, the effect of zero clustering for the case of modified B-spline interpolator is not indicated down to -100 dB.

Much more can be seen from the curve of the error approximation kernel, as of (7.42). As can be seen in Figure 7.19, the O-MOMS kernels (dashed lines) perform better for low frequencies, while the modified B-spline kernels (solid lines) are superior for higher frequencies (the curves to the left are for the third-degree kernels and the curves to the right are for the fifth-degree kernels). Note, however, that for low frequencies the predictive approximation error in the case of modified B-splines is less than 10^{-6}! This theoretical performance comparison concurs with the rotation experiment and shows that the modified kernels are potentially more promising because of their better high-frequency performance and reasonable and acceptable low-frequency error.

7.3.4.3. Total squared error

The total squared approximation error can be predicted using (7.54) for different interpolators and for different image models [63]. We have computed the error (7.54) for the following image models: white noise, Markov process, and an edge (unit step function), involving the interpolators from Section 7.3.4.2.

White noise. The power spectrum of the white noise is uniform and constant over the interval of integration. For this image model, the total squared error is obtained by integrating the error kernel (7.42) over the chosen interval. We

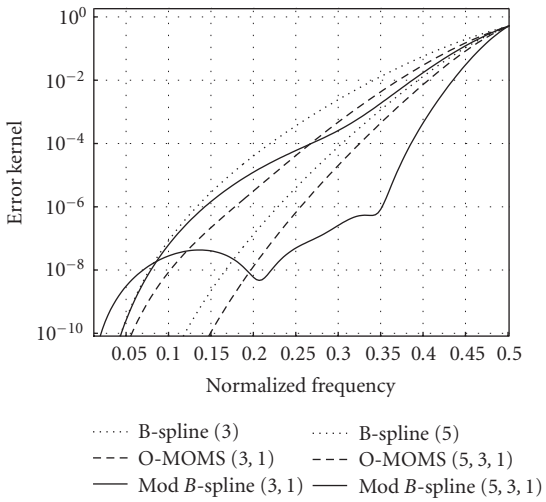


FIGURE 7.19. Approximation error kernels for different PP interpolators.

TABLE 7.5. Predictive squared error for different image models and different interpolators.

Interpolator	ε^2, [DB]	
	White noise	Markov process
B-spline (3)	13.15	28.91
O-MOMS (3,1)	14.03	29.98
Mod (3,1)	14.58	30.63
B-spline (5)	14.94	31.02
O-MOMS (5,3,1)	15.47	31.62
Mod (5,3,1)	17.02	33.62

have adopted the settings in [63], namely, the integration interval has been set to $[-\pi, \pi]$ (band-limited white noise). The results are summarized in Table 7.5.

Markov process. The Markov process has been chosen as it models a large class of real images [63]. In frequency domain, its power spectrum is described by

$$\left| G_a(2\pi f) \right|^2 = \frac{-2\log(\rho)}{(2\pi f)^2 + \log^2(\rho)} \tag{7.118}$$

with a correlation factor ρ less than but close to unity. To be consistent with the experiments in [63], we have set $\rho = 0.9$. The results are presented in the last column of Table 7.5. Even for such a low-frequency process the modified interpolation kernels play better, a fact emphasizing once again the importance of a proper compromise in specifying the optimization criteria.

Edge (step function). The step function is a model of sharp changes in the image intensity, that is, image edges. It is not a band-limited function and the accuracy of its reconstruction by means of interpolation depends on the phase of the sampling process, that is, the relative position of the samples around the step. For this image model, the measure (7.42) gives the amount of the approximation errors averaged over all possible sampling phases.

Assume an ideal rectangular scanning device aperture of unity length. It transforms the step function into the ramp function of unity width. Its spectrum is

$$\left| G_a(2\pi f) \right|^2 = \frac{\operatorname{sinc}^2(\pi f)}{(2\pi f)^2}. \tag{7.119}$$

One can represent the averaged error as function of the sampling frequency. The curves for some traditional interpolators, such as sinc (the ideal interpolator for band-limited functions), linear, and piecewise cubic, have been obtained in [49]. We add here the curves for the cubic B-spline interpolator (Figure 7.20(a)) and for the optimized O-MOMS (3,1) and Mod (3,1) (Figure 7.20(b)). As can be seen from the figures, for such a nonband-limited model the sinc interpolator is outperformed by the spline-based interpolators for a large range of sampling ratios. Also, the frequency domain specified optimization criteria that lead only to marginal improvement in this particular case.

7.4. Conclusions

In this section, we reviewed new spline-like basis functions, namely, PP functions of minimal support. We demonstrated how to improve the interpolation performance by taking optimized combinations of B-splines of different degrees. The combinations of third- and first-degree B-splines and of fifth-, third-, and first-degree B-splines were especially studied because they provide a good opportunity to deal with both smooth and sharp image areas and because they offer acceptable computational complexity. Two optimization techniques were presented: first one based on approximation theoretic assumptions and second one operating in frequency domain just like in a classical filter design. By the latter technique, the adjusted parameters were optimized in such a way that the resulting interpolation function sufficiently suppresses the imaging frequencies. As a result, the obtained PP interpolation kernels achieved better frequency characteristics than the classical B-splines, that is, flatter in the passband, steeper in the transition band, and with sufficient attenuation in the stopbands. While being very similar in construction to the so-called O-MOMS functions, the main difference between two optimized families is in the position of frequency response zeros around the multiples of the sampling rate ($2k\pi$ in terms of angular frequency). MOMS functions have multiple zeros at those points complying with the Strang-Fix conditions for Lth-order approximation functions. In the alternative design, the Lth order of approximation is traded for a better frequency domain stopband performance. In fact, as a result of the minimax optimization procedure the obtained functions have zeros

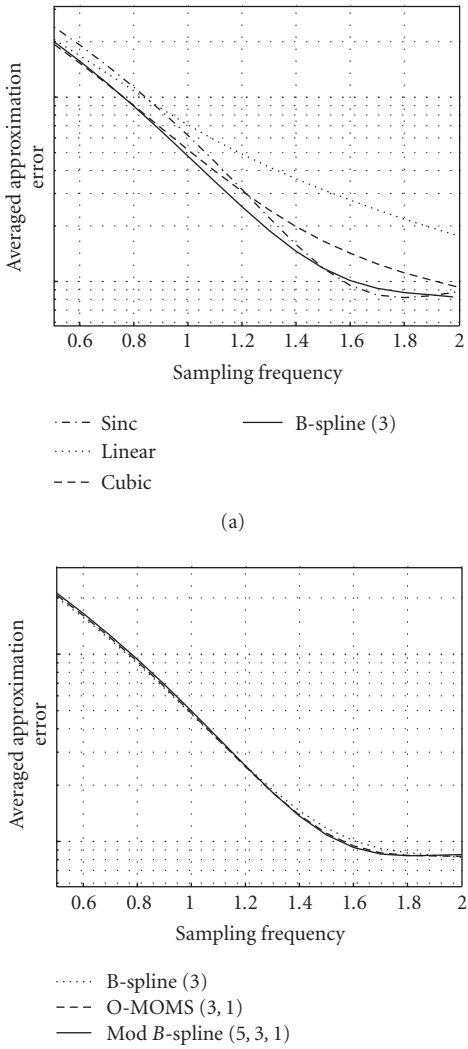


FIGURE 7.20. Averaged approximation error for image model "edge scanned with ideal aperture" and for different interpolators.

clustered around $(2k\pi)$, thus forming the desired stopbands. We showed that this optimization technique, working entirely in frequency domain, gives very promising results. This is done thanks to the good compromise between the increased interpolation capabilities for relatively high frequencies and the controllable low-frequency error. We experimentally proved the best interpolation capabilities of the functions thus constructed, especially for preserving high-frequency image details.

As far as the computational complexity is concerned, we described efficient filtering structures at arbitrary scale, based on the so-called Farrow structure.

Thanks to their susceptibility to optimization and computationally efficient structures, the modified B-spline kernels can considerably increase the interpolation quality and, therefore, they can be applied in image rescaling and rotation tasks and in approximation of multiscale operators as well.

Bibliography

[1] A. Aldroubi and M. Unser, "Families of multiresolution and wavelet spaces with optimal properties," *Numerical Functional Analysis and Optimization*, vol. 14, no. 5-6, pp. 417–446, 1993.

[2] A. Aldroubi and M. Unser, "Families of wavelet transforms in connection with Shannon's sampling theory and the Gabor transform," in *Wavelets: A Tutorial in Theory and Applications*, C. K. Chui, Ed., vol. 2, pp. 509–528, Academic Press, Boston, Mass, USA, 1992.

[3] A. Aldroubi and M. Unser, "Sampling procedures in function spaces and asymptotic equivalence with Shannon's sampling theory," *Numerical Functional Analysis and Optimization*, vol. 15, no. 1-2, pp. 1–21, 1994.

[4] A. Aldroubi, M. Unser, and M. Eden, "Cardinal spline filters: stability and convergence to the ideal sinc interpolator," *Signal Processing*, vol. 28, no. 2, pp. 127–138, 1992.

[5] D. Babic, J. Vesma, T. Saramäki, and M. Renfors, "Implementation of the transposed Farrow structure," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 4, pp. 5–8, Phoenix, Ariz, USA, May 2002.

[6] G. Battle, "A block spin construction of ondelettes. I. Lemarié functions," *Communications in Mathematical Physics*, vol. 110, no. 4, pp. 601–615, 1987.

[7] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, Pa, USA, 1996.

[8] T. Blu, P. Thévenaz, and M. Unser, "Minimum support interpolators with optimum approximation properties," in *Proceedings of International Conference on Image Processing (ICIP '98)*, vol. 3, pp. 242–245, Chicago, Ill, USA, October 1998.

[9] T. Blu, P. Thévenaz, and M. Unser, "MOMS: maximal-order interpolation of minimal support," *IEEE Transactions on Image Processing*, vol. 10, no. 7, pp. 1069–1080, 2001.

[10] T. Blu and M. Unser, "Quantitative Fourier analysis of approximation techniques. I. Interpolators and projectors," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2783–2795, 1999.

[11] C. de Boor, *A Practical Guide to Splines*, Springer, New York, NY, USA, 2nd edition, 2001.

[12] C. de Boor, R. A. DeVore, and A. Ron, "Approximation from shift-invariant subspaces of $L_2(\mathbb{R}^d)$," *Transactions of the American Mathematical Society*, vol. 341, no. 2, pp. 787–806, 1994.

[13] P. J. Burt, "Fast filter transform for image processing," *Computer Graphics and Image Processing*, vol. 16, no. 1, pp. 20–51, 1981.

[14] S. Chaudhuri, Ed., *Super-Resolution Imaging*, Kluwer Academic, Boston, Mass, USA, 2001.

[15] C. K. Chui, *Multivariate Splines*, vol. 54 of *CBMS-NSF Regional Conference Series in Applied Mathematics*, SIAM, Philadelphia, Pa, USA, 1988.

[16] C. K. Chui and J.-Z. Wang, "On compactly supported spline wavelets and a duality principle," *Transactions of the American Mathematical Society*, vol. 330, no. 2, pp. 903–915, 1992.

[17] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 45, no. 5, pp. 485–560, 1992.

[18] K. Egiazarian, T. Saramäki, H. Chugurian, and J. Astola, "Modified B-spline filters and interpolators," in *Proceedings of IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP '96)*, vol. 3, pp. 1743–1746, Atlanta, Ga, USA, May 1996.

[19] C. Farrow, "A continuously variable digital delay element," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '88)*, pp. 2641–2645, Espoo, Finland, June 1988.

[20] F. W. Gembicki, *Vector optimization for control with performance and parameter sensitivity indices*, Ph.D. thesis, Case Western Reserve University, Cleveland, Ohio, USA, 1974.

[21] A. Gotchev, J. Vesma, T. Saramäki, and K. Egiazarian, "Digital image resampling by modified B-spline functions," in *Proceedings of IEEE Nordic Signal Processing Symposium (NORSIG '00)*, pp. 259–262, Kolmarden, Sweden, June 2000.

[22] A. Gotchev, J. Vesma, T. Saramäki, and K. Egiazarian, "Multiscale image representations based on modified B-splines," in *Proceedings of International Workshop on Spectral Techniques and Logic Design (SPEGLOG '00)*, pp. 431–452, Tampere, Finland, June 2000.

[23] A. Gotchev, K. Egiazarian, J. Vesma, and T. Saramäki, "Edge-preserving image resizing using modified B-splines," in *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP '01)*, vol. 3, pp. 1865–1868, Salt Lake City,Utah, USA, May 2001.

[24] A. Gotchev, K. Egiazarian, and T. Saramäki, "Comparing the performance of optimized piecewise polynomial image interpolators," in *Proceedings of IEEE Nordic Signal Processing Symposium (NORSIG '02)*, Hurtigruta, Norway, October 2002, CDROM Proceedings SatAmOr4.

[25] A. Gotchev, T. Saramäki, and K. Egiazarian, "Spline and Wavelet Based Techniques for Signal and Image Processing," doctoral thesis, Tampere University of Technology, Tampere, Finland, September 2003.

[26] A. Gotchev and K. Egiazarian, "Spline-based resampling of noisy images," in *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis (ISPA '01)*, pp. 580–585, Pula, Croatia, June 2001.

[27] A. Gotchev, G. Marchokov, T. Saramäki, and K. Egiazarian, "Efficient algorithms and structures for image size reduction," in *Proceedings of TICSP International Workshop on Spectral Methods and Multirate Signal Processing (SMMSP '02)*, pp. 251–258, Toulouse, France, September 2002.

[28] A. Gotchev, K. Egiazarian, G. Marchokov, and T. Saramäki, "A near least squares method for image decimation," in *Proceedings of International Conference on Image Processing (ICIP '03)*, vol. 3, pp. 929–932, Barcelona, Spain, September 2003.

[29] T. Hentschel and G. Fettweis, "Continuous-time digital filters for sample-rate conversion in reconfigurable radio terminals," in *Proceedings of the European Wireless Conference (EW '00)*, pp. 55–59, Dresden, Germany, September 2000.

[30] H. S. Hou and H. C. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 6, pp. 508–517, 1978.

[31] R. Hummel, "Sampling for spline reconstruction," *SIAM Journal on Applied Mathematics*, vol. 43, no. 2, pp. 278–288, 1983.

[32] A. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1989.

[33] R. G. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[34] R. Lagendijk and J. Biemond, *Iterative Identification and Restoration of Images*, Kluwer Academic, Boston, Mass, USA, 1991.

[35] T. M. Lehmann, C. Gonner, and K. Spitzer, "Survey: interpolation methods in medical image processing," *IEEE Transactions on Medical Imaging*, vol. 18, no. 11, pp. 1049–1075, 1999.

[36] P. Lemarié, "Ondelettes à localisation exponentielle," *Journal de Mathématiques Pures et Appliquées*, vol. 67, no. 3, pp. 227–236, 1988.

[37] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[38] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, Calif, USA, 1998.

[39] G. Marchokov, A. Gotchev, J. Vesma, T. Saramäki, and K. Egiazarian, "Efficient polynomial-based methods for image resizing," in *Proceedings of the 3rd International Conference on Information, communications, and Signal Processing (ICICS '01)*, p. 5, Singapore, October 2001, CD-ROM Processing, 2D1.3.

[40] E. H. W. Meijering, K. J. Zuiderveld, and M. A. Viergever, "Image reconstruction by convolution with symmetrical piecewise *n*th-order polynomial kernels," *IEEE Transactions on Image Processing*, vol. 8, no. 2, pp. 192–201, 1999.

[41] E. H. W. Meijering, W. J. Niessen, and M. A. Viergever, "Quantitative evaluation of convolution-based methods for medical image interpolation," *Medical Image Analysis*, vol. 5, no. 2, pp. 111–126, 2001.

[42] A. Mertins, *Signal Analysis*, John Wiley & Sons, New York, NY, USA, 1999.

[43] S. Mitra, *Digital Signal Processing: A Computer-Based Approach*, McGraw-Hill, New York, NY, USA, 2001.

[44] A. Muñoz, T. Blu, and M. Unser, "Least-squares image resizing using finite differences," *IEEE Transactions on Image Processing*, vol. 10, no. 9, pp. 1365–1378, 2001.

[45] N. Nguyen, P. Milanfar, and G. Golub, "A computationally efficient superresolution image reconstruction algorithm," *IEEE Transactions on Image Processing*, vol. 10, no. 4, pp. 573–583, 2001.

[46] A. Oppenheim and R. Shafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1989.

[47] *Optimization Toolbox User's Guide*, MathWorks, 2002.

[48] S. K. Park and R. A. Schowengerdt, "Image sampling, reconstruction, and the effect of sample-scene phasing," *Applied Optics*, vol. 21, no. 17, pp. 3142–3151, 1982.

[49] S. K. Park and R. A. Schowengerdt, "Image reconstruction by parametric cubic convolution," *Computer Vision, Graphics, and Image Processing*, vol. 23, no. 3, pp. 258–272, 1983.

[50] S. Pissanetzky, *Sparse Matrix Technology*, Academic Press, London, UK, 1984.

[51] G. Ritter and J. Wilson, *Handbook of Computer Vision Algorithms in Image Algebra*, CRC Press, Boca Raton, Fla, USA, 2000.

[52] B. Romeny, Ed., *Geometry-Driven Diffusion in Computer Vision*, Kluwer Academic, Dordrecht, The Netherlands, 1994.

[53] A. Ron, "Factorization theorems for univariate splines on regular grids," *Israel Journal of Mathematics*, vol. 70, no. 1, pp. 48–68, 1990.

[54] J. Russ, *The Image Processing Handbook*, CRC Press, Boca Raton, Fla, USA, 2002.

[55] T. Saramäki, "Finite impulse response filter design," in *Handbook for Digital Signal Processing*, S. Mitra and J. Kaiser, Eds., pp. 155–277, John Wiley & Sons, New York, NY, USA, 1993.

[56] A. Schaum, "Theory and design of local interpolators," *CVGIP: Graphical Models and Image Processing*, vol. 55, no. 6, pp. 464–481, 1993.

[57] C. E. Shannon, "Communication in the presence of noise," *Proceedings of IEEE*, vol. 86, no. 2, pp. 447–457, 1998.

[58] I. Schoenberg, *Cardinal Spline Interpolation*, SIAM, Philadelphia, Pa, USA, 1973.

[59] G. Strang and G. Fix, "A Fourier analysis of the finite-element variational method," in *Constructive Aspect of Functional Analysis*, pp. 796–830, Edizioni Cremonese, Rome, Italy, 1971.

[60] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, Mass, USA, 1996.

[61] P. Thévenaz, T. Blu, and M. Unser, "Complete parameterization of piecewise polynomial interpolators according to degree, support, regularity, and order," in *Proceedings of International Conference on Image Processing (ICIP '00)*, vol. 2, pp. 335–338, Vancouver, BC, Canada, September 2000.

[62] P. Thévenaz, T. Blu, and M. Unser, "Image interpolation and resampling," in *Handbook of Medical Imaging, Processing and Analysis*, I. N. Bankman, Ed., pp. 393–420, Academic Press, San Diego Calif, USA, 2000.

[63] P. Thévenaz, T. Blu, and M. Unser, "Interpolation revisited," *IEEE Transactions on Medical Imaging*, vol. 19, no. 7, pp. 739–758, 2000.

[64] M. Unser, "Approximation power of biorthogonal wavelet expansions," *IEEE Transactions on Signal Processing*, vol. 44, no. 3, pp. 519–527, 1996.

[65] M. Unser, A. Aldroubi, and M. Eden, "Fast B-spline transform for continuous image representation and interpolation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 277–285, 1991.

[66] M. Unser and A. Aldroubi, "A general sampling theory for non-ideal acquisition devices," *IEEE Transactions on Signal Processing*, vol. 42, no. 11, pp. 2915–2925, 1994.

[67] M. Unser, A. Aldroubi, and M. Eden, "Polynomial spline signal approximations: filter design and asymptotic equivalence with Shannon's sampling theorem," *IEEE Transactions on Information Theory*, vol. 38, no. 1, pp. 95–103, 1992.

[68] M. Unser, A. Aldroubi, and M. Eden, "On the asymptotic convergence of B-spline wavelets to Gabor functions," *IEEE Transactions on Information Theory*, vol. 38, no. 2, part 2, pp. 864–872, 1992.

[69] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing. I. Theory," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 821–833, 1993.

[70] M. Unser, A. Aldroubi, and M. Eden, "B-spline signal processing. II. Efficient design and applications," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 834–848, 1993.

[71] M. Unser, A. Aldroubi, and M. Eden, "A family of polynomial spline wavelet transforms," *Signal Processing*, vol. 30, no. 2, pp. 141–162, 1993.

[72] M. Unser, A. Aldroubi, and S. J. Schiff, "Fast implementation of continuous wavelet transforms with integer scale," *IEEE Transactions on Signal Processing*, vol. 42, no. 12, pp. 3519–3523, 1994.

[73] M. Unser, "Splines: a perfect fit for signal and image processing," *Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, 1999.

[74] M. Unser and T. Blu, "Wavelet theory demystified," *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 470–483, 2003.

[75] M. Unser and I. Daubechies, "On the approximation power of convolution-based least squares versus interpolation," *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1697–1711, 1997.

[76] M. Unser, P. Thévenaz, and L. P. Yaroslavsky, "Convolution-based interpolation for fast, high-quality rotation of images," *IEEE Transactions on Signal Processing*, vol. 4, no. 10, pp. 1371–1381, 1995.

[77] J. Vesma and T. Saramäki, "Polynomial-based interpolation filters—part I: filter synthesis," to appear in *Circuits, Systems and Signal Processing*, vol. 26, no. 1, 2007.

[78] M. Veterli and H. Kovacevic, *Wavelets and Subband Coding*, Prentice-Hall, New York, NY, USA, 1995.

[79] G. Wahba, "Practical approximate solutions to linear operator equations when the data are noisy," *SIAM Journal on Numerical Analysis*, vol. 14, no. 4, pp. 651–667, 1977.

[80] Y.-P. Wang and S. L. Lee, "Scale-space derived from B-splines," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1040–1055, 1998.

[81] A. P. Witkin, "Scale-space filtering," in *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI '83)*, pp. 1019–1022, Karlsruhe, Germany, August 1983.

[82] L. P. Yaroslavsky and M. Eden, *Fundamentals of Digital Optics*, Birkhäuser, Boston, Mass, USA, 1996.

[83] L. P. Yaroslavsky, "Fast signal sinc-interpolation methods for signal and image resampling," in *Image Processing: Algorithms and Systems*, vol. 4667 of *Proceedings of SPIE*, pp. 120–129, San Jose, Calif, USA, January 2002.

Atanas Gotchev: Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland

Email: atanas.gotchev@tut.fi

Karen Egiazarian: Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland

Email: karen.egiazarian@tut.fi

Tapio Saramäki: Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland

Email: tapio.saramaki@tut.fi

8 Fast discrete sinc-interpolation: a gold standard for image resampling

L. Yaroslavsky

Numerous image processing tasks and applications require digital image resampling and geometrical transformations. These operations always assume building, from available image samples, a "continuous" image model, which is then resampled into the required new sample positions. For generating the "continuous" image model, available image samples are interpolated using, usually, convolution interpolation kernels. In the design of this process, a compromise is sought between the interpolation accuracy and computational complexity. In this chapter, we describe a family of discrete sinc-interpolation algorithms that can be regarded as a gold standard for interpolation of sampled signals defined by a final number of their samples. We prove that discrete sinc-interpolation is the only convolution-based interpolation method that preserves signal content in its baseband defined by the sampling rate, provide experimental evidence of its prevalence, outline its various algorithmic implementations based on fast transform, illustrate its applications for signal/image fractional shifts, image rotation, image resizing, signal integration and differentiation, and show that the availability of fast Fourier and fast DCT-based discrete sinc-interpolation algorithms makes discrete sinc-interpolation a very competitive alternative to other interpolation techniques in applications that are sensitive to interpolation accuracy and require image resampling in a regular equidistant sampling grid. For image resampling in irregular sampling grid, we suggest sliding window discrete sinc-interpolation methods that provide the best, for the given window size, interpolation accuracy and are, in addition, capable of simultaneous image resampling and filtering for restoration or enhancement and of local adaptation of the interpolation kernel.

8.1. Introduction

Accurate and fast signal and image resampling is a key operation in many digital image processing applications such as image reconstruction from projections, multimodality data fusion, image superresolution from image sequences, stabilization of video images distorted by atmosphere turbulence, target location and tracking with subpixel accuracy, and so forth, to name a few.

Resampling assumes interpolation of available signal/image samples to obtain samples in between the given ones. In some applications, for instance, in computer graphics and print art, simple interpolation methods, such as nearest neighbor or linear (bilinear), can provide satisfactory results. In applications that are more demanding in terms of the interpolation accuracy, higher-order spline interpolation methods gained popularity. The interpolation accuracy of spline interpolation methods is proportional to the spline order. In the same way, their computational complexity grows.

Meanwhile, there exists a unique discrete signal interpolation method that is capable, given finite set of signal samples, to secure virtually error-free signal interpolation. This method is the discrete sinc-interpolation. Discrete sinc-interpolation has been known for quite a lot of time but still remains to be a sort of an "ugly duckling" in the family of discrete signal interpolation methods. Virtually in all text and reference books on digital signal and digital image processing, one can find only brief mentioning discrete sinc-interpolation solely in an association with DFT spectrum zero-padding, which is rather inefficient and inflexible if compared, computationwise, with other interpolation methods. It is also most frequently confused with the continuous sinc-interpolation that, theoretically, secures error-free reconstruction of band-limited signals and least mean square error reconstruction of arbitrary continuous signals, provided infinitely large number of signal samples is available and that cannot be implemented in reality when only a finite number of signal samples is available.

The chapter intends to advocate broad use of the discrete sinc-interpolation in signal and especially in image processing. To this goal, we, in Section 8.2, provide a theoretical proof that discrete sinc-interpolation is the only convolution-based interpolation method that preserves signal content in its frequency baseband defined by the signal sampling rate. In Section 8.3, we support this proof by experimental comparison of discrete sinc-interpolation with other interpolation methods in terms of the interpolation accuracy. In Section 8.4, we describe various fast Fourier and fast DCT transform-based algorithms for signal and image subsampling, arbitrary fractional shift, zooming, rotation and scaling, and numerical integration and differentiation with discrete sinc-interpolation that uses all available signal samples. As these algorithms are computationally efficient only if resampling in a regular equidistant sampling grid is required, we, in Section 8.5, suggest sliding window DCT domain discrete sinc-interpolation algorithms that are suitable for signal resampling in arbitrary irregular sampling grid, provide the best, for the given window size, interpolation accuracy, are capable, additionally to interpolation, of simultaneous signal/image restoration and enhancement, and allow easy local adaptation of the interpolation kernel. Necessary mathematical details and auxiliary materials are entered in the appendices.

8.2. Optimality of the discrete sinc-interpolation: a mathematical formulation

For the purposes of the resampling filter design, one can regard signal coordinate shift as a general resampling operation. This is justified by the fact that samples of

the resampled signal for any arbitrary signal resampling grid can be obtained one by one through the corresponding signal shift to the given sample position. In this section, we introduce, for signals defined by the finite number of their samples, an optimal signal shifting resampling filter that preserves signal content in its frequency baseband defined by the signal sampling rate and show that the filter point spread function (impulse response) is the discrete sinc-function.

8.2.1. Preliminaries and denotations

Let $\{a_k\}$ be a set of samples of an analog signal $a(x)$, $N = 0, 1, \ldots, N-1$:

$$a_k = \int_{-\infty}^{\infty} a(x)\varphi^{(s)}(x - \tilde{k}^{(s)}\Delta x)\,dx, \tag{8.1}$$

where $\{\varphi^{(s)}(\cdot)\}$ is a set of sampling basis functions, $\tilde{k}^{(s)} = k + u^{(s)}$, and $u^{(s)}$ is a shift, in units of the sampling interval Δx of the signal sampling grid with respect to signal coordinate system.

We assume that, for signal shifting resampling, resampled signal samples $\{\tilde{a}_k^{\tilde{x}}\}$ are obtained by means of digital convolution

$$\tilde{a}_k^{\tilde{x}} = \sum_{n=0}^{N-1} h_n^{(\mathrm{intp})}(\tilde{x})a_{k-n} \tag{8.2}$$

of the initial signal samples $\{a_k\}$ with a certain interpolation kernel $h_n^{(\mathrm{intp})}(\tilde{x})$ (resampling filter point spread function), where $\tilde{x}$ is a required signal shift with respect to the sampling grid of samples $\{a_k\}$.

We assume also that samples $\{\tilde{a}_k^{\tilde{x}}\}$ correspond to a continuous $\tilde{x}$-shifted signal:

$$\tilde{a}(x) = \sum_{k=0}^{N-1} \tilde{a}_k^{\tilde{x}}\varphi^{(r)}(x - \tilde{k}^{(r)}\Delta x), \tag{8.3}$$

where $\{\varphi^{(r)}(\cdot)\}$ is a set of reconstruction basis functions, as, for instance, those that describe image display devices, $\tilde{k}^{(r)} = k + u^{(r)}$, and $u^{(r)}$ is a shift, in units of the sampling interval Δx of the reconstructed signal sampling grid with respect to reconstructed device coordinate system.

8.2.2. Continuous impulse and frequency responses of digital resampling filters

Signal resampling is a linear signal transformation. As any linear transformation, it is completely characterized by its impulse response, or point spread function (PSF). In order to find PSF of the resampling transformation, let us establish alink

between reconstructed resampled $\widetilde{x}$-shifted signal $\widetilde{a}(x)$ of (8.3) and initial signal $a(x)$. To this goal, combine (8.1)–(8.3) and obtain

$$\begin{aligned}\widetilde{a}(x) &= \sum_{k=0}^{N-1}\sum_{n=0}^{N-1} h_n^{(\mathrm{intp})}(\widetilde{x})\varphi^{(r)}\left[x-(k+u^{(r)})\Delta x\right]\int_{-\infty}^{\infty} a(\xi)\varphi^{(s)}\left[\xi-(k-n+u^{(s)})\Delta x\right]d\xi \\ &= \int_{-\infty}^{\infty} a(\xi)\left\{\sum_{k=0}^{N-1}\sum_{n=0}^{N-1} h_n^{(\mathrm{intp})}(\widetilde{x})\varphi^{(r)}\left[x-(k+u^{(r)})\Delta x\right]\varphi^{(s)}\left[\xi-(k-n+u^{(s)})\Delta x\right]\right\}d\xi \\ &= \int_{-\infty}^{\infty} a(\xi)\widetilde{h}_{\widetilde{x}}^{(\mathrm{intp})}(x,\xi)\,d\xi.\end{aligned} \tag{8.4}$$

Function

$$\widetilde{h}_{\widetilde{x}}^{(\mathrm{intp})}(x,\xi) = \sum_{k=0}^{N-1}\sum_{n=0}^{N-1} h_n^{(\mathrm{intp})}(\widetilde{x})\varphi^{(r)}\left[x-(k+u^{(r)})\Delta x\right]\varphi^{(s)}\left[\xi-(k-n+u^{(s)})\Delta x\right] \tag{8.5}$$

in (8.4) can be treated as the *overall point spread function* (*OPSF*) of the numerical signal $\widetilde{x}$-shifting resampling process described by (8.2).

Define *overall frequency response* of the shifting resampling process as 2D Fourier transform of its *OPSF*:

$$H_{\widetilde{x}}^{(\mathrm{intp})}(f,p) = \iint_{-\infty}^{\infty} h_{\widetilde{x}}^{(\mathrm{intp})}(x,\xi)\exp\left[i2\pi(fx-p\xi)\right]dx\,d\xi. \tag{8.6}$$

Replace, in (8.6), $h^{(\mathrm{intp})}(x,\xi)$ with its expression given by (8.5), and obtain

$$\begin{aligned}&H_{\widetilde{x}}^{(\mathrm{intp})}(f,p) \\ &\quad= \iint_{-\infty}^{\infty}\sum_{k=0}^{N-1}\sum_{n=0}^{N-1} h_n^{(\mathrm{intp})}(\widetilde{x})\varphi^{(r)}\left[x-(k+u^{(r)})\Delta x\right] \\ &\qquad\qquad\times\varphi^{(s)}\left[\xi-(k-n+u^{(s)})\Delta x\right]\exp\left[i2\pi(fx-p\xi)\right]dx\,d\xi \\ &\quad= \sum_{k=0}^{N-1}\sum_{n=0}^{N-1} h_n^{(\mathrm{intp})}(\widetilde{x})\int_{-\infty}^{\infty}\varphi^{(r)}\left[x-(k+u^{(r)})\Delta x\right]\exp(i2\pi fx)dx \\ &\qquad\times\int_{-\infty}^{\infty}\varphi^{(s)}\left[\xi-(k-n+u^{(s)})\Delta x\right]\exp(-i2\pi p\xi)\,d\xi\end{aligned}$$

$$
\begin{aligned}
&= \sum_{k=0}^{N-1} \sum_{n=0}^{N-1} h_n^{(\mathrm{intp})}(\tilde{x}) \exp\left[i2\pi f\left(k + u^{(r)}\right)\Delta x\right] \\
&\quad \times \exp\left[-i2\pi p\left(k + u^{(r)} - u^{(r)} - n + u^{(s)}\right)\Delta x\right] \\
&\quad \times \int_{-\infty}^{\infty} \varphi^{(r)}(x) \exp(i2\pi f x)dx \int_{-\infty}^{\infty} \varphi^{(s)}(\xi) \exp(-i2\pi p\xi)d\xi \\
&= \sum_{n=0}^{N-1} h_n^{(\mathrm{intp})}(\tilde{x}) \exp\left[i2\pi p\left(n + u^{(r)} - u^{(s)}\right)\Delta x\, d\xi\right] \sum_{k=0}^{N-1} \\
&\quad \times \exp\left[i2\pi(f-p)\left(k + u^{(r)}\right)\Delta x\right] \\
&\quad \times \int_{-\infty}^{\infty} \varphi^{(r)}(x) \exp(i2\pi f x)dx \int_{-\infty}^{\infty} \varphi^{(s)}(\xi) \exp(-i2\pi p\xi)d\xi.
\end{aligned}
\tag{8.7}
$$

The first term in this product

$$
\overline{H}_{\tilde{x}}^{(\mathrm{intp})}(p) = \sum_{n=0}^{N-1} h_n^{(\mathrm{intp})}(\tilde{x}) \exp\left[i2\pi p\left(n + u^{(r)} - u^{(s)}\right)\Delta x\right] \tag{8.8}
$$

is determined solely by the resampling filter coefficients $\{h_n^{(\mathrm{intp})}(\tilde{x})\}$ and signal sampling interval Δx. We call this term "*continuous frequency response* (*CFrR*) *of the resampling filter*."

The second term can be represented as

$$
\begin{aligned}
\mathrm{SV}_N(f-p) &= \sum_{k=0}^{N-1} \exp\left[i2\pi(f-p)\left(k + u^{(r)}\right)\Delta x\right] \\
&= \frac{\exp\left[i2\pi(f-p)N\Delta x\right] - 1}{\exp\left[i2\pi(f-p)\Delta x\right] - 1} \exp\left[i2\pi(f-p)u^{(r)}\Delta x\right] \\
&= \frac{\sin\left[\pi(f-p)N\Delta x\right]}{\sin\left[\pi(f-p)\Delta x\right]} \exp\left[i\pi(N-1)(f-p)\left(N-1+2u^{(r)}\right)\Delta x\right] \\
&= N\,\mathrm{sincd}\left[N;\pi(f-p)N\Delta x\right] \exp\left[i\pi(N-1)(f-p)\left(N-1+2u^{(r)}\right)\Delta x\right],
\end{aligned}
\tag{8.9}
$$

where

$$
\mathrm{sincd}(N;x) = \frac{\sin(x)}{\sin(x/N)} \tag{8.10}
$$

is the *discrete sinc-function*.

It is only natural to maintain the following relationship between the number of signal samples N and sampling grid shift $u^{(r)}$ in the display coordinate system:

$$u^{(r)} = -\frac{N-1}{2}. \tag{8.11}$$

In this case, the phase term in (8.9) becomes zero, and

$$\mathrm{SV}_N(f-p) = N\,\mathrm{sincd}\,[N;\pi(f-p)N\Delta x]. \tag{8.12}$$

This term describes how the overall frequency response of the resampling process is affected by the finiteness of the number of signal samples. Specifically, it shows that, for any finite number of signal samples N, numerical signal resampling is not shift invariant process and it is subjected to boundary effects. The contribution of the boundary effects decays when the number of signal samples N grows. In the limit, when $N \to \infty$,

$$\lim_{N\to\infty} \mathrm{SV}_N(f-p) = \lim_{N\to\infty} N\,\mathrm{sincd}\,[N;\pi(f-p)N\Delta x] = \delta[\pi(f-p)N\Delta x], \tag{8.13}$$

and the numerical signal resampling process tends to become shift invariant.

Two last terms in (8.7),

$$\begin{aligned} \Phi^{(s)}(p) &= \int_{-\infty}^{\infty} \varphi^{(s)}(\xi)\exp(-i2\pi p\xi)d\xi, \\ \Phi^{(r)}(f) &= \int_{-\infty}^{\infty} \varphi^{(r)}(x)\exp(i2\pi f x)dx, \end{aligned} \tag{8.14}$$

are frequency responses of signal sampling and reconstruction devices.

Given the number of signal samples and signal sampling and reconstruction devices, the quality of the numerical resampling process can be evaluated in terms of the continuous frequency response (*CFrR*) $\overline{H}_{\tilde{x}}^{(\mathrm{intp})}(p)$ of the resampling filter defined by (8.3).

In order to facilitate analysis of the CFrR, it is useful to link it with samples $\{\eta_r^{(\mathrm{intp})}\}$ of a shifted discrete Fourier transform (SDFT) of the resampling filter coefficients $\{h_n^{(\mathrm{intp})}(x)\}$. For SDFT $(u,0)$, these sets of the coefficients are related through the equation

$$h_n^{(\mathrm{intp})}(\tilde{x}) = \frac{1}{\sqrt{N}}\sum_{r=0}^{N-1} \eta_r^{(\mathrm{intp})}(\tilde{x})\exp\left[-i2\pi\frac{(n+u)r}{N}\right], \tag{8.15}$$

where u is a signal domain transform shift parameter (see Chapter 3). We will refer to the set of coefficients $\eta_r^{(\mathrm{intp})}(\tilde{x})$ as to *discrete frequency response* of the resampling filter.

Replacing $\{h_n^{(\mathrm{intp})}(x)\}$ in (8.3) with its expression in (8.15), obtain

$$\begin{aligned}
H^{(\mathrm{intp})}(p) &= \sum_{n=0}^{N-1} \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \eta_r^{(\mathrm{intp})}(\tilde{x}) \exp\left[-i2\pi \frac{(n+u)r}{N} \right] \\
&\quad \times \exp\left[i2\pi p (n + u^{(r)} - u^{(s)}) \Delta x \right] \\
&= \sum_{r=0}^{N-1} \eta_r^{(\mathrm{intp})}(\tilde{x}) \left\{ \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \exp\left[-i2\pi \frac{(n+u)r}{N} \right] \right. \\
&\qquad \left. \times \exp\left[i2\pi p (n + u^{(r)} - u^{(s)}) \Delta x \right] \right\} \\
&= \sum_{r=0}^{N-1} \eta_r^{(\mathrm{intp})}(\tilde{x}) \left\{ \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \exp\left[i2\pi \left(p\Delta x - \frac{r}{N} \right) n \right] \right\} \\
&\quad \times \exp\left(-i2\pi \frac{ur}{N} \right) \exp\left[i2\pi p (u^{(r)} - u^{(s)}) \Delta x \right] \qquad (8.16) \\
&= \sum_{r=0}^{N-1} \eta_r^{(\mathrm{intp})}(\tilde{x}) \frac{1}{\sqrt{N}} \frac{\exp\left[i2\pi (p\Delta x - r/N) N \right] - 1}{\exp\left[i2\pi (p\Delta x - r/N) N \right] - 1} \\
&\quad \times \exp\left(-i2\pi \frac{ur}{N} \right) \exp\left[i2\pi p (u^{(r)} - u^{(s)}) \Delta x \right] \\
&= \sum_{r=0}^{N-1} \eta_r^{(\mathrm{intp})}(\tilde{x}) \frac{\sqrt{N}}{N} \frac{\sin\left[\pi (p\Delta x - r/N) N \right]}{\sin\left[i2\pi (p\Delta x - r/N) \right]} \\
&\quad \times \exp\left[-i2\pi \frac{r}{N} \left(u + \frac{(N-1)}{2} \right) \right] \\
&\quad \times \exp\left[i2\pi p \left(u^{(r)} - u^{(s)} + \frac{(N-1)}{2} \right) \Delta x \right].
\end{aligned}$$

Provided reconstruction shift parameter is determined by (8.11), the natural selections of signal sampling shift parameter $u^{(s)}$ and SDFT shift parameter u are

$$u^{(s)} = 0, \qquad u = -\frac{N-1}{2}. \qquad (8.17)$$

Then, obtain finally

$$H^{(\mathrm{intp})}(p) \propto \sum_{r=0}^{N-1} \eta_r^{(\mathrm{intp})}(\tilde{x}) \operatorname{sinc}\left[N; \pi \left(p\Delta x - \frac{r}{N} \right) N \right]. \qquad (8.18)$$

With this, we arrive to the following theorems.

Theorem 8.1. *SDFT*$(-(N-1)/2, 0)$ *coefficients of resampling filter point spread function* $\{h_n^{(\mathrm{intp})}(\cdot)\}$*, or its discrete frequency response* $\eta_r^{(\mathrm{intp})}(\cdot)$*, are samples of the filter continuous frequency response* $H^{(\mathrm{intp})}(p)$.

Theorem 8.2. *Continuous frequency response* $H^{(\mathrm{intp})}(p)$ *of the resampling filter is a function, discrete-sinc interpolated from its samples* $\eta_r^{(\mathrm{intp})}(\cdot)$*, taken with sampling interval* $1/N\Delta x$*, where* N *is the number of signal samples involved in the resampling and* Δx *is signal sampling interval.*

Note that, generally, these theorems are applicable to any digital filter and its discrete and continuous frequency responses.

8.2.3. Optimal resampling filter

We define the optimal shifting resampling filter as the filter that generates a shifted copy of the input signal and preserves samples of the analog signal spectrum in its baseband defined by its sampling rate and the number of available signal samples. According to this definition, continuous frequency response of the optimal shifting resampling filter for the coordinate shift $\delta\widetilde{x}$ is, by virtue of the Fourier transform shift theorem,

$$H^{(\mathrm{intp})}(p) = \exp(i2\pi p\delta\widetilde{x}). \tag{8.19}$$

Then, according to Theorem 8.1, for odd number of signal samples N, samples $\{\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}(\delta\widetilde{x})\}$ of CFrR of the optimal resampling filter that define its point spread function $\{h_n^{(\mathrm{intp})}(\delta\widetilde{x})\}$ should be selected as

$$\begin{aligned} \eta_{r,\mathrm{opt}}^{(\mathrm{intp})}(\delta\widetilde{x}) &= \frac{1}{\sqrt{N}}\exp\left(i2\pi\frac{r\delta\widetilde{x}}{N\Delta x}\right), \quad r = 0, 1, \ldots, \frac{N-1}{2}, \\ \eta_{r,\mathrm{opt}}^{(\mathrm{intp})}(\delta\widetilde{x}) &= \eta_{N-r,\mathrm{opt}}^{*(\mathrm{intp})}(\delta\widetilde{x}), \quad r = \frac{N+1}{2}, \ldots, N-1. \end{aligned} \tag{8.20}$$

For even number of signal samples N, coefficient $\eta_{N/2,\mathrm{opt}}^{(\mathrm{intp})}(\delta\widetilde{x})$, which corresponds to the signal higher frequency in its baseband, requires a special treatment. As it is shown in Appendix A, this coefficient must be an imaginary number in order to keep filter point spread function $\{h_n^{(\mathrm{intp})}(\delta\widetilde{x})\}$ to be real valued. This limitation inevitably prevents the $N/2$th signal spectral coefficient from being exact sample of the ideal frequency response and assumes its arbitrary choice. The most

natural one is the assignment

$$\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}(\delta\widetilde{x}) = \begin{cases} \exp\left(i2\pi\dfrac{r\delta\widetilde{x}}{N\Delta x}\right), & r = 0, 1, \ldots, \dfrac{N}{2} - 1, \\ -Ai\sin\left(\pi\dfrac{\delta\widetilde{x}}{\Delta x}\right), & r = \dfrac{N}{2}, \end{cases} \tag{8.21}$$

$$\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}(\delta\widetilde{x}) = -\left(\eta_{N-r,\mathrm{opt}}^{(\mathrm{intp})}(\delta\widetilde{x})\right)^*, \quad r = \frac{N}{2} + 1, \ldots, N - 1,$$

where A is a weight coefficient that defines signal spectrum shaping at its highest frequency component. In what follows, we will consider, for even N, the following three options for A:

$$\begin{aligned} &\text{Case 0} : A = 0, \\ &\text{Case 1} : A = 1, \\ &\text{Case 2} : A = 2. \end{aligned} \tag{8.22}$$

It is shown in Appendix B that, for odd N, point spread function of the optimal resampling filter defined by (8.15) is

$$h_n^{(\mathrm{intp})}(\delta\widetilde{x}) = \mathrm{sincd}\left\{N, \pi\left[n - \frac{N-1}{2} - \frac{\delta\widetilde{x}}{\Delta x}\right]\right\}. \tag{8.23a}$$

For even N, Case 0 and Case 2, optimal resampling point spread functions are

$$h_n^{(\mathrm{intp}\,0)}(\delta\widetilde{x}) = \overline{\mathrm{sincd}}\left\{N; N-1; \pi\left[n - \frac{N-1}{2} - \frac{\delta\widetilde{x}}{\Delta x}\right]\right\}, \tag{8.23b}$$

$$h_n^{(\mathrm{intp}\,2)}(\delta\widetilde{x}) = \overline{\mathrm{sincd}}\left\{N; N+1; \pi\left[n - \frac{N-1}{2} - \frac{\delta\widetilde{x}}{\Delta x}\right]\right\}, \tag{8.23c}$$

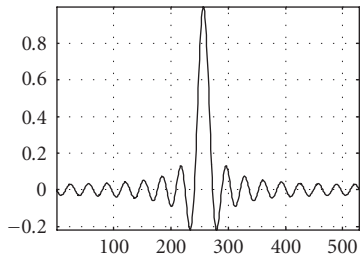
correspondingly, where a modified sincd-function $\overline{\mathrm{sincd}}$ is defined as

$$\overline{\mathrm{sincd}}(N; M; x) = \frac{\sin(Mx/N)}{N\sin(x/N)}. \tag{8.24}$$

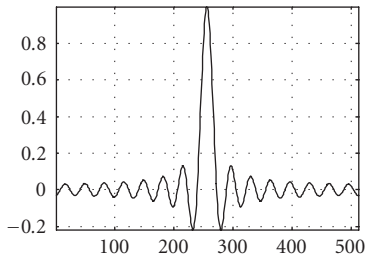
One can easily see that Case 1 is just a combination of Case 0 and Case 2:

$$\begin{aligned} h_n^{(\mathrm{intp}\,2)}(\delta\widetilde{x}) &= \frac{h_n^{(\mathrm{intp}\,0)}(\delta\widetilde{x}) + h_n^{(\mathrm{intp}\,2)}(\delta\widetilde{x})}{2} \\ &= \overline{\mathrm{sincd}}(\pm 1; N; x) = \frac{\overline{\mathrm{sincd}}(N-1; N; x) + \overline{\mathrm{sincd}}(N+1; N; x)}{2}. \end{aligned} \tag{8.23d}$$

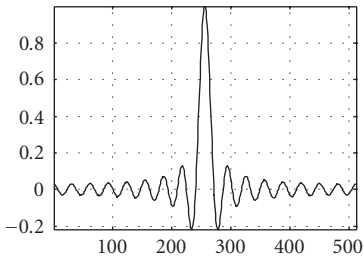
These four functions $h_n^{(\mathrm{intp})}(\delta\widetilde{x})$ are illustrated in Figure 8.1 for $N = 512$. As one can see, Case 1-discrete sinc-function decays to zero much faster than three other



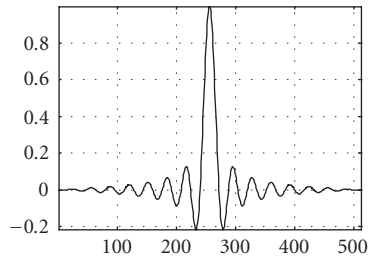
(a) $\operatorname{sincd}(N;x) = \overline{\operatorname{sincd}}(N;N;x)$



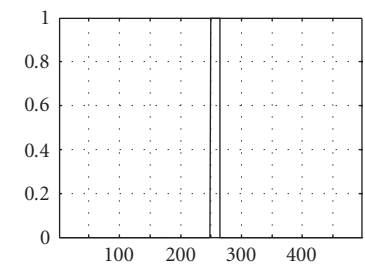
(b) $\overline{\operatorname{sincd}}(N-1;N;x)$



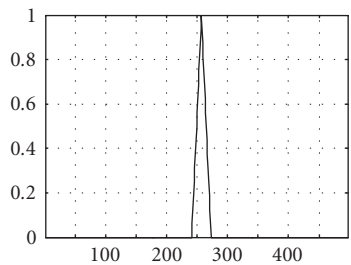
(c) $\overline{\operatorname{sincd}}(N+1;N;x)$



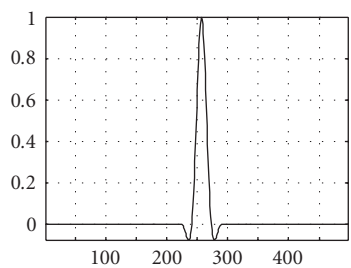
(d) $\overline{\operatorname{sincd}}(\pm 1;N;x)$



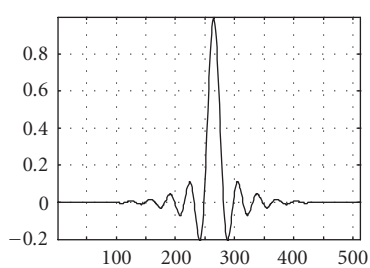
(e) Nearest neighbor



(f) Linear



(g) Cubic



(h) Mems(5, 3, 1)

FIGURE 8.1. Four versions of discrete sinc-function of (8.23a)–(8.23d) ((a)–(d)) and point spread functions of four spline interpolation methods ((e)–(h)).

versions and therefore is less liable to boundary effects. We call signal resampling, by means of filters defined by (8.23), *discrete sinc-interpolation*.

Such result can be formulated as the follows.

Theorem 8.3. *For analog signals defined by their N samples, discrete sinc-interpolation is, for odd N, the only discrete convolution-based signal resampling method that does not distort signal spectrum samples in its baseband specified by the signal sampling rate Δx. For even N, discrete sinc-interpolation distorts only the highest N/2th frequency spectral component.*

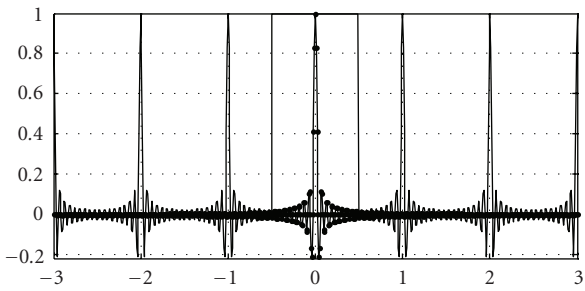
In conclusion of this section, note that the reasoning concerning the design of the optimal resampling filter discussed in this section can also be used for the design of numerical implementation of any shift invariant signal linear transformations. In Section 8.4.5, we will use it for the design and comparison of algorithms for signal numerical differentiation and integration.

8.2.4. Optimal resampling filter: discussion and interpretation; discrete sinc- and continuous sinc-functions

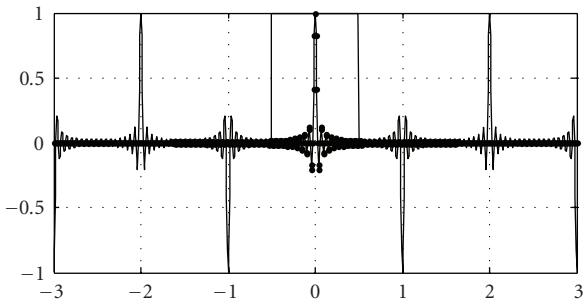
Discrete sinc functions sincd and $\overline{\text{sincd}}$ defined by (8.10) and (8.24) are discrete point spread functions of the ideal digital lowpass filter, whose discrete frequency response is a rectangular function. Depending on whether the number of signal samples N is odd or even number, they are periodic or antiperiodic with period N, as it is illustrated in Figures 8.2(a) and 8.2(b). According to Theorem 8.2, continuous frequency response of discrete sinc-interpolators is a function, discrete sinc-interpolated from its discrete frequency response. Both continuous and discrete frequency responses of discrete sinc-interpolators are illustrated in Figure 8.2(c).

As one can see from the figure, discrete sinc-inerpolation does preserve signal spectrum in spectrum sampling points, while in between sampling points spectrum oscillations are possible due to the continuous frequency response interpolation. However, these oscillations are caused solely by the finite number of signal samples and depend solely on signal boundary conditions. Specifically in image processing, this means that they can be significantly weakened by an appropriate windowing of the interpolated discrete signal at its boundaries without any change of the interpolation PSF. It is in this sense one can say that the preservation of signal spectral samples in the process of signal resampling is the necessary and sufficient condition of signal spectrum preservation as far as it concerns the selection of the resampling interpolation function. In Section 8.3, we will provide an experimental evidence in favor of this statement.

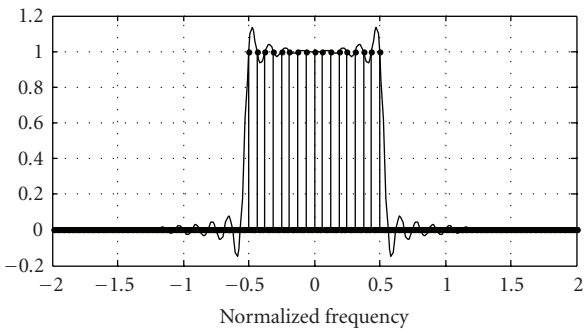
Discrete sinc-functions are plotted in Figure 8.2 along with the continuous sinc-function $\text{sinc}\, x = \sin x/x$. It is a Fourier transform of the rectangular function, which is the frequency response of the ideal lowpass filter for continuous signals. Discrete sinc-functions are discrete analogs of the continuous sinc-function, however, these functions must not be confused.



(a)



(b)



(c)

FIGURE 8.2. (a)-(b) Continuous (bold circles) versus discrete (thin line) sinc-functions for odd and even N, respectively; (c) continuous frequency response of the sincd-interpolator (solid line) and samples of the ideal lowpass filter frequency response (bold circles).

Sinc-function is an aperiodic function with infinite support. It is a convolution kernel that preserves spectra of continuous signals in their baseband defined by the sampling rate. For band-limited signals, it provides signal perfect reconstruction from their samples provided infinite number of the samples is available. For signals that are not band-limited, it provides least mean square error signal reconstruction under the same condition.

Discrete sinc-interpolation, in its turn, provides perfect, for the given number of signal samples, discrete signal resampling with preservation of corresponding continuous signal spectra in their sampling points. Continuous frequency response of discrete-sinc interpolators coincides with that of continuous sinc-interpolator in spectral sampling points defined by the number of the discrete signal samples, but deviates from it between sampling points, deviations being dependent on discrete signal boundary conditions.

Continuous sinc-function is a mathematical model that cannot be implemented in reality due to its infinite support. As a practical substitute for sinc-function, a windowed sinc-function is considered frequently for interpolation purposes. As a convolutional interpolation kernel, it has no indisputable advantages over other interpolation kernels for the same number of samples. Discrete sinc-functions must not be confused with a windowed sinc-function either.

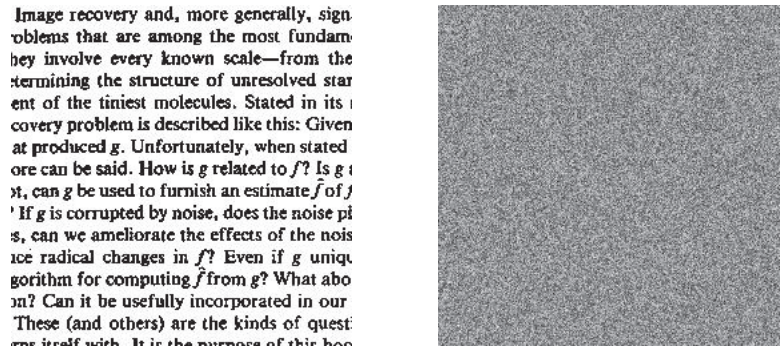
8.3. Discrete sinc-interpolation versus other interpolation methods: performance comparison

In this section, we provide experimental evidence that the discrete sinc-interpolation outperforms other known interpolation methods in terms of the interpolation accuracy and signal preservation and is capable of virtually error-free signal interpolation. Compared with discrete sinc-interpolation are nearest neighbor interpolation, linear (bilinear) interpolation, cubic (bicubic) spline interpolation, and higher-order spline interpolation methods: Mems(5, 3, 1) (see [1]), and Beta11 spline (see [2]). Point spread functions of the first 4 interpolation methods are shown in Figures 8.1(e)–8.1(h), correspondingly, where they can be compared with point spread functions of the above described modifications of the discrete sinc interpolation ((a)–(d)).

Mentioned interpolation methods were compared in two applications: image rotation and image zooming. In image rotation, several rotations of test images were performed using mentioned interpolation methods so as the total rotation angle be multiple of 360°. Then, the rotation error was evaluated as a difference between initial and rotated images, and its DFT power spectrum was evaluated and displayed. For image zooming, test images were subjected to 4 times zooming and then DFT spectra of zoomed images were computed and displayed in order to evaluate spectra aliasing artifacts introduced by the methods.

For discrete sinc-interpolation, the version with point spread function shown in Figure 8.1(d) was used. In experiments with nearest neighbor, bilinear, and bicubic interpolation, Matlab programs **imrotate.m** and **interp2.m** from image processing tool box were used. In experiments with Mems(5, 3, 1)-interpolation, a software package kindly provided by Dr. Gotchev (see [3]) was used. Discrete sinc-interpolation used in the experiments was implemented in Matlab using standard Matlab tools.

As test images, two images were used: an image of a printed text ("Text" image, Figure 8.3(a)) and a computer generated pseudorandom image with uniform DFT spectrum ("PRUS" image, Figure 8.3(b)). The "Text" image was used to evaluate



(a) (b)

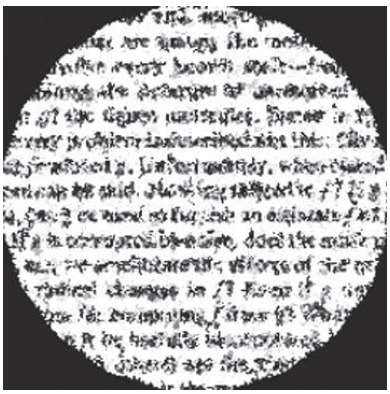
FIGURE 8.3. Test images used in the experiments: (a) "Text" image; (b) "PRUS" image.

readability of the text for different rotation method. The "PRUS" image was used for demonstrating distortions of signal spectra caused by the resampling.

Results of comparison, for rotation of test image "Text," of three standard methods of image resampling, nearest neighbor, bilinear and bicubic ones, and that of discrete-sinc interpolation in application to image rotation, are shown in Figures 8.4(a)–8.4(d).

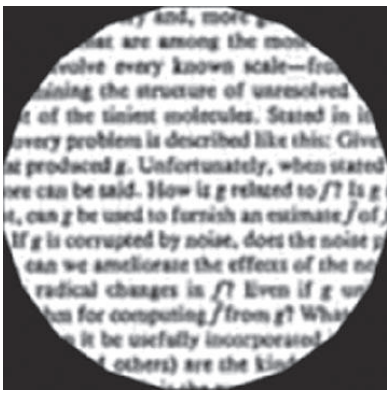
As it was mentioned, for the standard methods, Matlab program **imrotate.m** from image processing tool box was used. For discrete sinc-interpolated rotation, a Matlab code was written using standard Matlab tools that implements the 3-step rotation algorithm (see [4]) through DFT-based fractional shift algorithm described in the next section. Images shown in the figure clearly show that, after 60 rotations though 18° each, standard interpolation methods completely destroy the readability of the text, while discrete sinc-interpolated rotated image is virtually not distinguishable from the original one.

Nearest neighbor, bilinear, and bicubic interpolations are spline interpolations of the first, second, and third order. The higher the spline order, the higher interpolation accuracy, and the higher their computational complexity. However, higher-order spline interpolators still underperform the discrete sinc-interpolation as one can see from Figures 8.5(a) and 8.5(b) where results of 1000 rotations, through 18° each rotation, of the "Text" image using Mems(5, 3, 1)-based rotation program and the above mentioned Matlab code that implements 3-step rotations with discrete sinc-interpolation. While rotated test image obtained with Mems(5, 3, 1)-interplolation, is substantially blurred, image, rotated with discrete sinc-interpolation remains unchanged. Figures 8.5(c) and 8.5(d) are reference images that represent the test image low pass filtered to 0.5 (image (c)) and to 0.4 (image (d)) of the baseband. Comparing them with Mems(5, 3, 1)-rotated image, one can say that, after 1000 rotations, Mems(5, 3, 1)-rotated image is equivalent, from the point of view of the text readability, to a lowpass filtered image with the

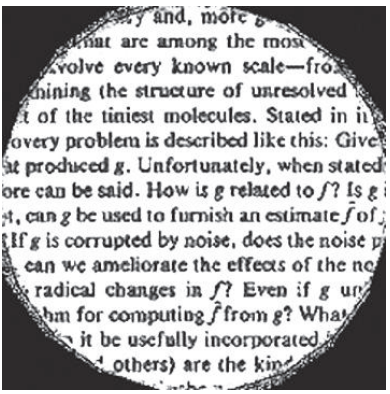


(a) 1080 degrees rotated image in 60 steps (NearNeighb, $T = 7.268$)

(b) 1080 degrees rotated image in 60 steps (Bilinear, $T = 11.1$)



(c) 1080 degrees rotated image in 60 steps (Bicubic, $T = 17.67$)

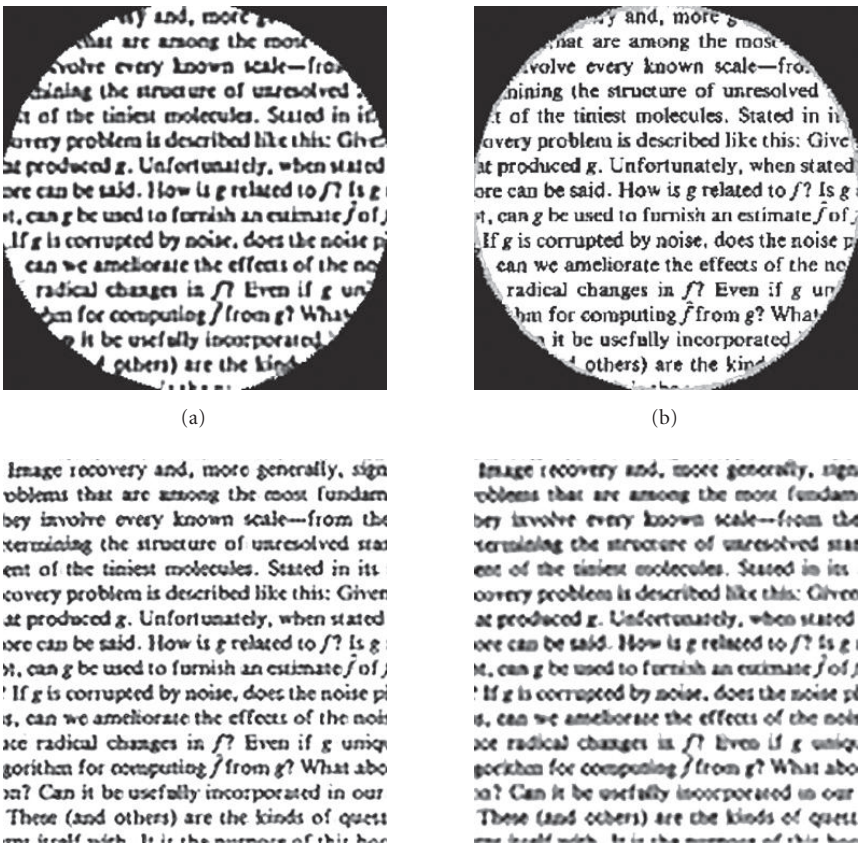


(d) 1080 degrees rotated image in 60 steps (Discrete sinc, $T = 14.158$)

FIGURE 8.4. Discrete sinc-interpolation versus conventional interpolation methods: results of multiple rotation of the test image "Text." Numbers in brackets indicate processing time.

residual bandwidth of 0.4-0.5 of the initial one. Note that in these experiments, both compared methods had comparable run time on the same computer.

In the analysis of the interpolation errors, it is very instructive to compare their power spectra. Figures 8.6(a)–8.6(d) present power spectra of rotation errors obtained for the test image "PRUS" rotated 10 times through 36° each time, using bicubic interpolation (Matlab program **imrotate.m**), Mems(5, 3, 1), Beta11, and discrete sinc-interpolation, correspondingly. Rotation results obtained with Beta11 spline interpolation were kindly provided by Dr. Thévenaz (see [2]). Rotation error spectra for Mems(5, 3, 1) and discrete sinc-interpolation and larger number (100) of rotations through multiple of 360° are shown for comparison in Figures 8.7(a) and 8.7(b), correspondingly. All spectra in Figures 8.6 and 8.7 are centered around zero frequency (dc) component; left-right and top-bottom



(a) (b)

(c) (d)

FIGURE 8.5. Discrete sinc-interpolation versus Mems(5, 3, 1) interpolation method: results of image rotations through 18 000° in 1000 steps using Mems(5, 3, 1) interpolation (a) and discrete sinc-interpolation (b) and reference test images lowpass filtered to 0.5 (c) and to 0.4 (d) of the baseband.

borders of the images correspond to the highest horizontal and, correspondingly, vertical spectral components.

These spectra clearly show that, of those four methods, only discrete sinc-interpolation preserves image content in its baseband. For discrete sinc-interpolation, only spectral components outside the circle with radius equal to the highest horizontal/vertical frequency (frequency index $N/2$) are distorted. These distortions are, in accordance with Theorem 8.3, associated with the implementation of the discrete sinc-interpolation for even number of signal samples. Plots of horizontal and vertical spectra sections (Figures 8.6(e), 8.6(f)) illustrate spectra of rotation error in more quantitative terms.

In Figures 8.6 and 8.7, one can also see that all interpolation methods produce, in image rotation, errors outside a circle inscribed into square that corresponds to the image baseband area. These errors are rotation aliasing errors. They

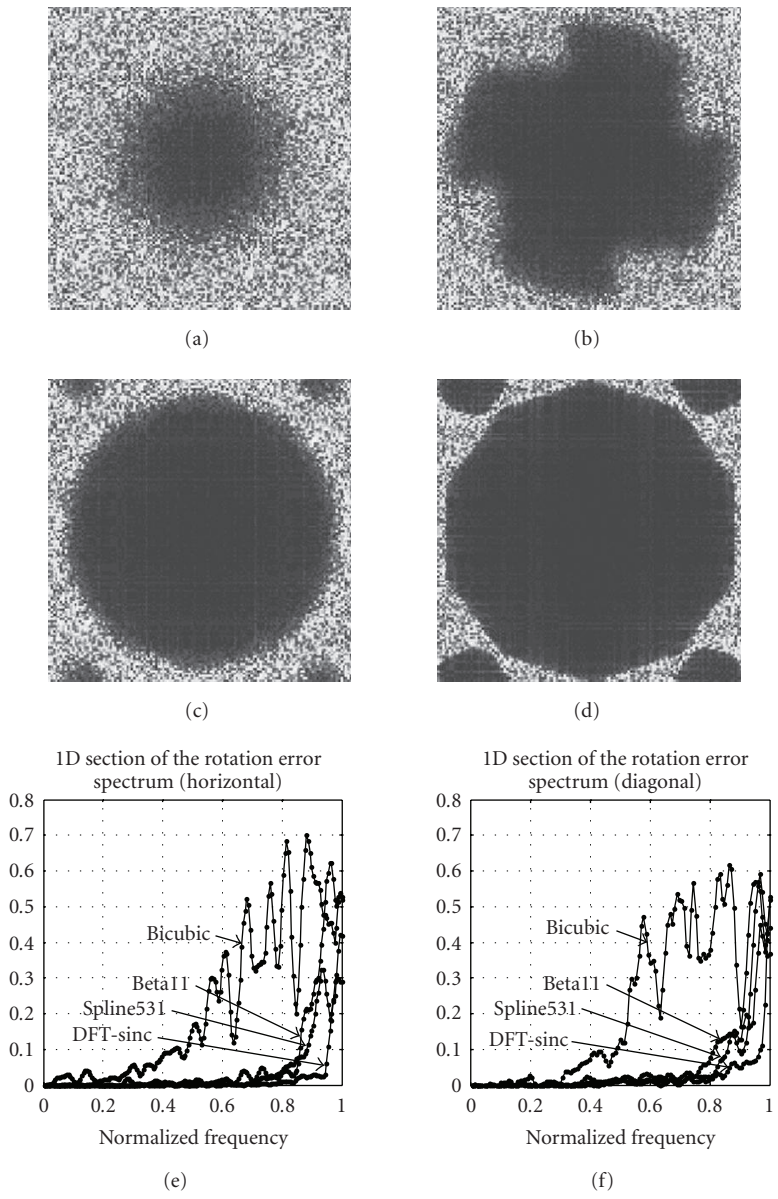
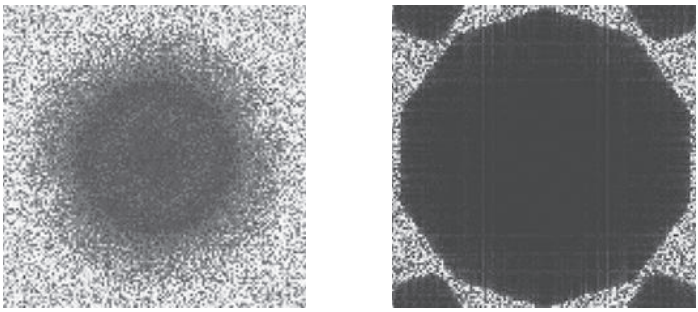


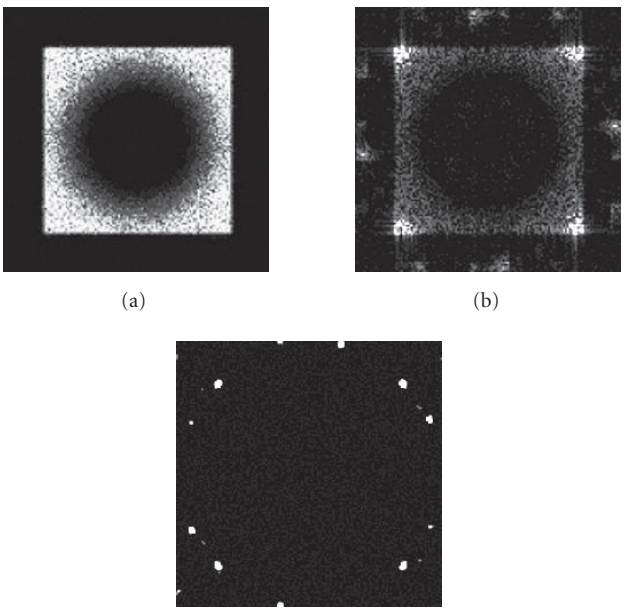
FIGURE 8.6. (a)–(c): rotation error spectra for bicubic (a), Mems(5, 3, 1) (b), Beta11 (c), and discrete sinc-interpolation (d) interpolation methods; (e)–(f): vertical and horizontal cross-sections of the 2D spectra (dots: spectra samples; solid lines: continuous spectra). All spectra are shown centered at spectrum zero frequency (dc) component, image lightness being proportional to spectra intensity.

originate from image diagonal spectral components that should, due to the rotation, go outside the baseband square and are put inside it because of the cyclicity of the DFT. Figure 8.8 shows rotation error spectra for $10 \times 36^\circ$ rotations, using



(a) (b)

FIGURE 8.7. Rotation error spectra for $100 \times 36^\circ$ rotations of "PRUS"-image using Mems(5, 3, 1)-interpolation (a) and discrete sinc-interpolation (b). All spectra are shown centered at spectrum zero frequency (dc) component image lightness being proportional to spectra intensity.



(a) (b)

(c)

FIGURE 8.8. Rotation error spectra for $10 \times 36^\circ$ rotations of PRUS image low pass filtered to 0.7 of its baseband for bicubic (a), Mems(5, 3, 1) (b) and discrete sinc-interpolation (c) methods. All spectra are shown centered at spectrum zero frequency (dc) component, image lightness being proportional to spectra intensity.

bicubic, spline351, and sinc-interpolation, of PRUS image that was lowpass pre-filtered to 0.7 of the baseband in horizontal and vertical dimensions. As a result of such a lowpass filtering, aliasing error components are excluded, and one can see from the figure that discrete sinc-interpolation, as opposite to other methods,

does not produce, in this case, rotation errors at all (few components still visible in Figure 8.8(c) are intentionally left as reference points that keep, for display purposes, dynamic range of the error for all compared methods).

Experimental results presented in Figures 8.4–8.8 illustrate interpolation errors within the image baseband. Aliasing interpolation errors outside the image baseband for different interpolation methods, in spectra of 4x-zooming "PRUS" image are illustrated in Figure 8.9 for bilinear, bicubic, Mems(5, 3, 1)- and discrete sinc-interpolation methods. One can clearly see that spectrum of discrete-sinc-interpolated zoomed image (Figure 8.9(d)) does not contain any aliasing components while spectra images zoomed with other methods do. Corresponding 1D horizontal sections of these spectra shown in Figures 8.9(d)-8.9(e) illustrate this quantitatively.

8.4. Global image resampling: fast discrete sinc-interpolation algorithms and applications

8.4.1. Signal subsampling (zooming in) by DFT and DCT spectrum zero-padding

Fast discrete sinc-interpolation algorithms are all based on fast transforms (fast Fourier transform or fast DCT transform). The most well-known signal resampling algorithm that implements discrete sinc-interpolation is the signal subsampling algorithm by means of zero-padding of its DFT spectrum. Given subsampling (zoom) factor M, the algorithm generates, from signal of N samples, a signal of MN samples in which every Mth samples are corresponding samples of the original signal and the rest $(M-1)N$ samples are discrete sinc-interpolated from original samples. The algorithm can be described by the equation

$$\widetilde{a}_{\widetilde{k}} = \mathrm{IFFT}_{MN}\left\{\mathrm{DFT\_ZP}_{M}\left[\mathrm{FFT}_{N}\left(a_k\right)\right]\right\}, \tag{8.25}$$

where $\{a_k\}, k = 0, 1, \ldots, N-1$, are input signal samples, $\{\widetilde{a}_{\widetilde{k}}\}, \widetilde{k} = 0, 1, \ldots, NM-1$, are output signal samples, M is a signal zoom (subsampling) factor, $\mathrm{FFT}_N(\cdot)$ and $\mathrm{FFT}_{MN}(\cdot)$ are direct and inverse fast Fourier transform operators of N- and, correspondingly, MN-point. $\mathrm{DFT\_ZP}_M[\cdot]$ in (8.25) is a zero-padding operator that forms, from N-points sequence of samples, an MN-points sequence by padding the former with $(M-1)N$ zeros. When N is an odd number, zeros are placed between $(N-1)/2$th and $(N+1)/2$th samples of the N-points sequence. When N is an even number, then, either

(i) $(M-1)N+1$ zeros are placed between $N/2-1$th and $N/2+1$th samples of the N-points sequence and $N/2$th sample is discarded, or

(ii) $(M-1)N$ zeros are paced after $N/2$th sample and then the sequence repeated beginning of its $N/2$th sample, or

(iii) $(M-1)N$ zeros are paced after $N/2$th sample of the sequence, this sample is halved and then $N/2$th through $(N-1)$th samples are placed, $N/2$th sample being halved.

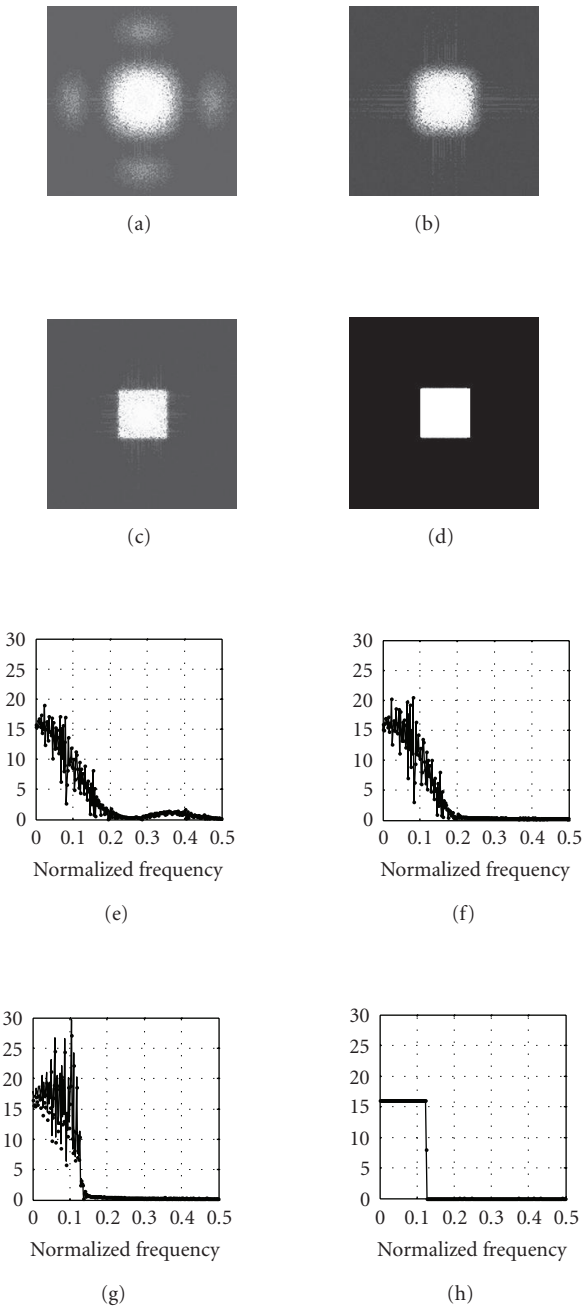


FIGURE 8.9. Spectra of 4x-zoomed "PRUS" image for bilinear (a) bicubic (b), Mems(5, 3, 1) (c), and discrete sinc-interpolation (d) methods and corresponding central horizontal sections of the spectra ((e)–(h)). All spectra are shown centered at spectrum zero frequency (dc) component, image lightness being proportional to spectra intensity. Note speckle patterns in spectra (a)–(c) caused by the interpolation inaccuracy.

Cases (i)–(iii) implement above described Case 0, Case 2, and Case 1 discrete sinc-interpolation, respectively.

Although discrete sinc-interpolation is optimal in preserving signal spectrum, it has one major drawback. As interpolation kernel discrete sinc-function decays to zero relatively slow, discrete sinc-interpolation heavily suffers from boundary effects that may propagate rather far from signal boundaries. It is especially the case in image processing when image size rarely exceeds, in each dimension, 1000 samples. With the use of FFT in the resampling algorithm, images are treated as being virtually periodic. Therefore, samples at their left and right and, correspondingly, upper and bottom borders are, virtually, immediate neighbors in the interpolation process. Therefore, any discontinuity between opposite border samples will cause heavy discrete sinc-function oscillations that propagate far away from the borders.

A simple and very efficient solution of this problem is working in the domain of discrete cosine transform (DCT) instead of DFT:

$$\widetilde{a}_{\widetilde{k}} = \mathrm{IDCT}_{MN}\left\{\mathrm{DCT\_ZP}_{M}\left[\mathrm{DCT}_{N}\left(a_k\right)\right]\right\}, \tag{8.26}$$

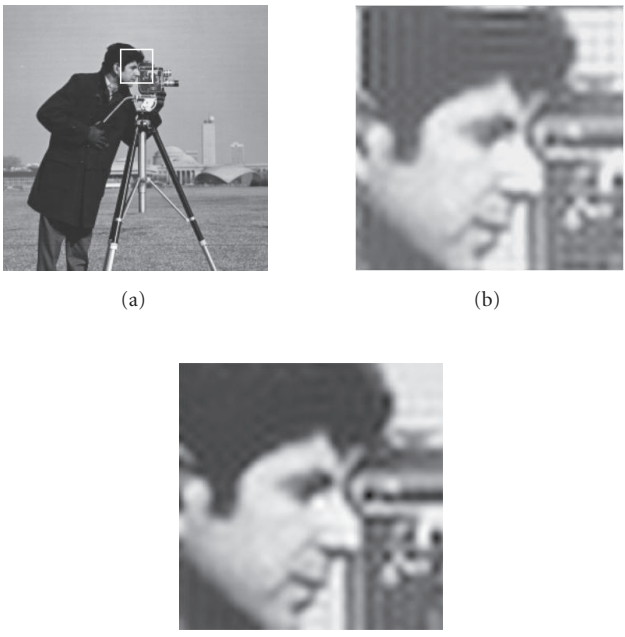
where $\mathrm{DCT}_N(\cdot)$ and $\mathrm{IDCT}_{MN}(\cdot)$ are N-points fast direct and MN-points inverse discrete cosine transforms and $\mathrm{DCT\_ZP}_M[\cdot]$ is a DCT spectrum zero-padding operator that places $(M-1)N$ zeros after the last $(N-1)$th DCT spectrum sample. For faster decay of the interpolation kernel, it is also advisable to halve the last two samples that, for DCT, represent signal highest frequency component.

Figure 8.10 demonstrates virtual absence of boundary effect oscillations for image zooming by DCT spectrum zero-padding algorithm compared to DFT spectrum zero-padding.

As it is shown in Appendix C, point spread function of the DCT spectrum zero-padding with halving its last two components is defined by equation

$$\begin{aligned}\widetilde{\widetilde{a}}_{\widetilde{k}} = \frac{\alpha_0^{\mathrm{DCT}}}{\sqrt{2LN}} + \frac{1}{N\sqrt{L}}\sum_{n=0}^{N-1} a_n \Bigg\{ & \frac{\sin\left[\pi\left((N+1)/2NL\right)(\widetilde{n}L-\widetilde{k})\right]}{\sin\left[\pi(\widetilde{n}L-\widetilde{k})/2NL\right]}\cos\left[\frac{\pi(\widetilde{n}L-\widetilde{k})}{2L}\right] \\ & + \frac{\sin\left[\pi\left((N+1)/2NL\right)(\widetilde{n}L+\widetilde{k})\right]}{\sin\left[\pi(\widetilde{n}L+\widetilde{k})/2NL\right]}\cos\left[\frac{\pi(\widetilde{n}L+\widetilde{k})}{2L}\right] \\ & + \frac{\sin\left[\pi\left((N-1)/2NL\right)(\widetilde{n}L-\widetilde{k})\right]}{\sin\left[\pi(\widetilde{n}L-\widetilde{k})/2NL\right]}\cos\left[\pi\frac{N-2}{2NL}(\widetilde{n}L-\widetilde{k})\right] \\ & + \frac{\sin\left[\pi\left((N-1)/2NL\right)(\widetilde{n}L+\widetilde{k})\right]}{\sin\left[\pi(\widetilde{n}L+\widetilde{k})/2NL\right]}\cos\left[\pi\frac{N-2}{2NL}(\widetilde{n}L+\widetilde{k})\right]\Bigg\}.\end{aligned} \tag{8.27}$$

One can see from (8.27) that DCT zero-padding interpolation is not a cyclic shift invariant and is implemented by two pairs of sincd-function interpolation kernels 1/2-shifted and mirror reflected with respect to each other. Its behavior is illustrated in Figure 8.11 for sampling positions at boundaries and at the middle of the signal extent.



(a) (b)

(c)

FIGURE 8.10. Zooming of an image fragment outlined by white box in (a): zero-padding its DFT spectrum (b) and zero-padding DCT spectrum (c). Note that heavy oscillations in the vicinity of image borders in image (a) disappear in image (b).

As it is immediately seen from (8.27), point spread function of the DCT zero-padding interpolation approximates the discrete sinc-function very closely. This close similarity is illustrated in Figure 8.12 that shows PSFs (Figure 8.12(a)) and discrete frequency responses (Figure 8.12(b)) of DFT and DCT zero-padding for signal 4x-zooming.

Computationwise, signal L-times zooming by zero-padding its DFT or DCT spectra requires $O(\log N)$ operations per each signal N samples for direct transform and $O(L \log NL)$ operation for inverse transform of zero pad spectra which is quite inefficient for large L. Moreover, when conventional radix 2 FFT and fast DCT algorithms are used, zooming factor L should be selected to be an integer power of 2. These two facts limit applicability of the spectra zero-padding algorithms. However, in certain cases the use of above described algorithms is a good practical solution. This might be, for instance, the case when one works in a certain software environment, such as Matlab. Image zooming by means of DCT spectra zero-padding can be naturally used when images are represented in a compressed form such as in JPEG compression. In this case, zooming can be carried out without the need to decompress images. In the next sections, much more computationally efficient and flexible discrete sinc-interpolation algorithms will be described

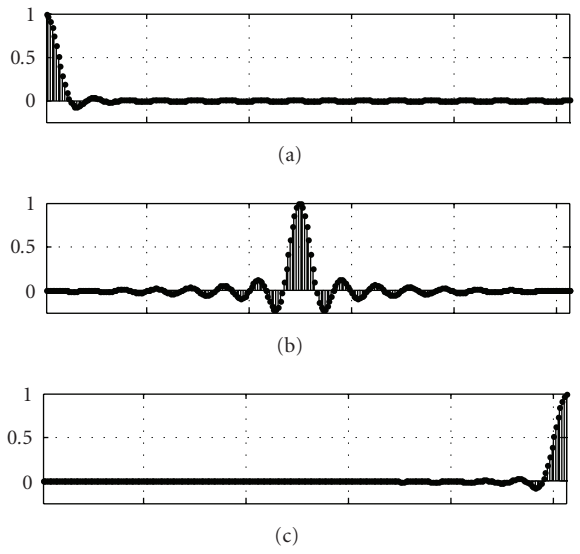


FIGURE 8.11. Point spread function of DCT zero-padding for three different sampling positions.

that can compete, both in terms of the interpolation accuracy and computational complexity, with other interpolation methods.

In conclusion, note that signal interpolation by DFT and DCT spectra zero-padding can be naturally extended to interpolation by zero-padding of signal spectra in other bases, such as Walsh or wavelet ones. This option is discussed in Appendix D.

8.4.2. DFT- and DCT-based signal fractional shift algorithms

In this section, we describe signal fractional shift algorithms that provide a computationally efficient and competitive solution for virtually error-free signal resampling. Consider first the DFT-based algorithm that computes, for signal samples $\{a_k\}$, discrete sinc-interpolated samples $\{a_k^{(u)}\}$ of a signal shifted with respect to the initial signal by an arbitrary distance u times signal sampling interval Δx. The algorithm is described by the equation

$$\widetilde{a}_{\widetilde{k}} = \mathrm{IFFT}_N \left\{ \{\eta^{(\mathrm{intp})}(u\Delta x)\} \bullet [\, \mathrm{FFT}_N \, (a_k)\,] \right\}, \tag{8.28}$$

where $\mathrm{FFT}_N(\cdot)$ and $\mathrm{IFFT}_N(\cdot)$ are direct and inverse N-point fast Fourier transforms, $\bullet$ symbolizes pointwise (Hadamard) matrix product and $\{\eta_r^{(\mathrm{intp})}(u\Delta x)\}$ is a set of coefficients defined, for odd N by the above (8.20) in which $\delta\widetilde{x}$ should be replaced by $u\Delta x$. For even N, (8.21) is applicable with a slight modification that $\eta_{N/2}^{(\mathrm{intp})}(u\Delta x)$ should be taken equal to $A\cos(2\pi ru/N)$ because simple DFT is used in the algorithm rather than SDFT assumed in the derivation of (8.21) and, therefore,

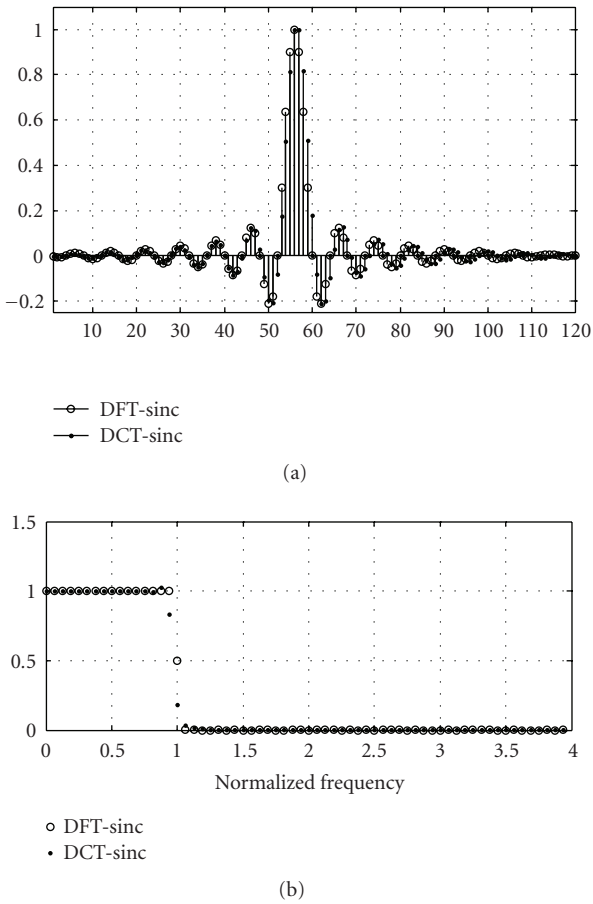


FIGURE 8.12. Point spread functions (a) and frequency responses (b) of DCT spectrum zero-padding (8.27) and DFT zero-padding (sincd-function $\mathrm{sincd}(\pm 1, N, x)$ of (8.23d)) for 4x-zooming.

set of coefficients $\{\eta_r^{(\mathrm{intp})}(u\Delta x)\}$ must obey complex conjugate symmetry property of DFT spectra of real sequences.

We will call this algorithm "*DFT-based signal fractional shift algorithm*." This algorithm can be recommended for numerous applications. First of all, it allows generating arbitrarily shifted discrete sinc-interpolated copies of signals. For instance, it is especially useful for implementing image shearing in 3-step image rotation algorithm (see [4]) in which, at each of the three stages, image shearing (correspondingly, vertical, horizontal, and another vertical) is performed. Above presented experimental results for image rotation with discrete sinc-interpolation were obtained using DFT-based fractional shift algorithm.

Another option is signal/image zooming-in with an arbitrary integer zoom factor. For zoom factor L, signal/image zooming can be implemented through combination of $L-1$ signal/image copies shifted by corresponding multiple of $1/L$

shifts. Above presented experimental results with discrete sinc-interpolated image zooming were obtained using this algorithm.

One of the applications in which image rescaling with high interpolation accuracy is required is real time location and tracking of moving targets in video sequences. In this case, template images of the target with arbitrary orientation and scale can be very rapidly computed using bilinear interpolation or similar fast resampling techniques applied to a reference image of the target highly oversampled using DFT/DCT-based zooming. Paper [5] reports such a hybrid resampling for an implementation of the direct Fourier method of tomographic reconstruction (see also [6, Section 9.4.5]).

Yet another application option is signal/image subsampling in the vicinity of a selected sample; this feature enables realization of continuous spectrum analysis and computing signal correlation with subpixel accuracy for target location.

Being a cyclic convolution, the "DFT-based" signal fractional shift algorithm suffers from same boundary effects as the DFT zero-padding algorithm does. The efficient practical solution of the problem is to implement convolution in DCT domain instead of DFT domain [7, 8]. Signal convolution in DCT domain is equivalent to shifted DFT(1/2,0) convolution of an auxiliary signal obtained from the initial signal by its extension to double length by means of its mirrored copy to eliminate discontinuities at the signal boundaries. For signal $u\Delta x$-shift, the algorithm is described by the equation

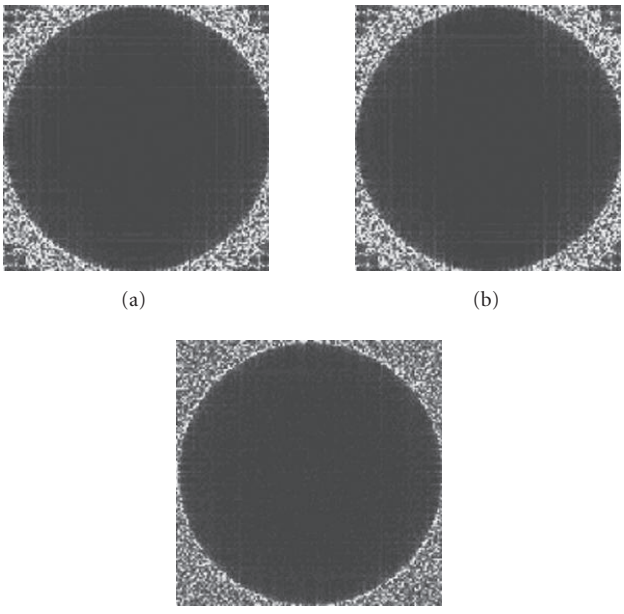
$$\begin{aligned}\widetilde{a}_{\widetilde{k}} = {} & \mathrm{IDCT}_N \left\{ \left[\eta_r^{(\mathrm{intp}),\mathrm{re}}(u\Delta x) \right] \bullet \left[\mathrm{DCT}_N \left(a_k \right) \right] \right\} \\ & - \mathrm{IDcST}_N \left\{ \left[\eta_r^{(\mathrm{intp}),\mathrm{im}}(u\Delta x) \right] \bullet \left[\mathrm{DCT}_N \left(a_k \right) \right] \right\},\end{aligned} \tag{8.29}$$

where $\mathrm{DCT}_N(\cdot)$ and $\mathrm{IDCT}_N(\cdot)$ are operators of direct and inverse DCT transforms, $\mathrm{IDcST}_N(\cdot)$ is operator of the inverse discrete cosine-sine transform (for definitions of DCT and DcST, see Chapter 3), and $\{\eta_r^{(\mathrm{intp}),\mathrm{re}}(u\Delta x)\}$, $\{\eta_r^{(\mathrm{intp}),\mathrm{im}}(u\Delta x)\}$ are real and imaginary parts of coefficients $\{\eta_r^{(\mathrm{intp})}(u\Delta x)\}$ defined by (8.20) in which N should be replaced by $2N$:

$$\begin{aligned}\{\eta_r^{(\mathrm{intp}),\mathrm{re}}(u\Delta x)\} &= \left\{ \cos\left(\frac{\pi u r}{N}\right) \right\}, \\ \{\eta_r^{(\mathrm{intp}),\mathrm{im}}(u\Delta x)\} &= \left\{ \sin\left(\frac{\pi u r}{N}\right) \right\}, \quad r = 0, 1, \ldots, N-1,\end{aligned} \tag{8.30}$$

and $\bullet$ symbolizes elementwise (Hadamard) product of vectors.

With respect to the boundary effects, DCT-based fractional shift algorithm is as efficient in eliminating boundary effects as the above-described DCT spectrum zero-padding algorithm. In terms of the interpolation accuracy, it is equivalent to the above DFT-based fractional shift algorithm as it implements convolution with discrete sinc-function. Experimental evidence of the equivalency is illustrated in Figures 8.13(a) and 8.13(b).



(a) (b)

(c)

FIGURE 8.13. Error spectra for $60\times6^\circ$ rotations of PRUS-image test image. (a): DFT-sinc interpolation; (b): DCT-sinc interpolation; (c): RotDFT-algorithm. All spectra are shown centered at spectrum zero frequency (dc) component. Standard deviation of the rotation error is practically the same for all cases: 20.0, 19.4 and 21.1, respectively.

8.4.3. Signal zooming using "scaled" DFT

DFT- and DCT-based fractional shift algorithms originate from the concept of shifted discrete fourier transforms (SDFT) that are obtained, for sampled signals, from Fourier integral transform in the assumption of an arbitrary shift of the signal and its Fourier spectra sampling grid shifts with respect to continuous signal and its spectrum coordinate systems (see [9] and also Chapter 3). Conventional discrete Fourier transform can be regarded as a special case of SDFTs when no signal and its spectrum sampling grids are assumed. It is also assumed in the derivation of DFTs that signal and its Fourier spectrum sampling intervals, say Δx and Δf, are linked with the number N of signal samples, according to the sampling theorem, by the "*cardinal sampling*" condition

$$\Delta f = \frac{1}{N\Delta x}. \tag{8.31}$$

As it is shown in Chapter 3, if one assumes a certain σ-times oversampling or undersampling, with respect to the cardinal sampling, with sampling interval $\Delta x = 1/\sigma N\Delta f$ in σ-scaled coordinates and sampling grid shifts (u_σ, v_σ) in,

respectively, signal and transform domains, *scaled discrete Fourier transforms* (ScDFT$(u_\sigma, v_\sigma; \sigma)$)

$$\alpha_r^{u,v;\sigma} = \frac{1}{\sqrt{\sigma N}} \sum_{k=0}^{N-1} a_k \exp\left[i2\pi \frac{(k+u_\sigma)(r+v_\sigma)}{\sigma N} \right] \tag{8.32}$$

can be obtained.

Similar to the case of shifted DFT, where the availability of arbitrary shift parameters enables, as it was described above, arbitrary fractional shift of sampled signal with discrete sinc-interpolation, the availability of an arbitrary scale parameter in ScDFT enables discrete signal resampling into an arbitrary scale by means of applying to the signal direct and inverse ScDFTs with different scale parameters. The following equations prove that this resampling is carried out with discrete-sinc interpolation. Let

$$\alpha_r = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} a_n \exp\left[i2\pi \frac{(n+u_0)(r+v_0)}{N} \right] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} a_n \exp\left[i2\pi \frac{\widetilde{n}(r+v_0)}{N} \right], \tag{8.33}$$

where $\widetilde{n} = n + u_0$, be SDFT(u_0, v_0) coefficients of a signal specified by its samples $\{a_n\}$. Applying to these coefficients ScDFT$(u_\sigma, v_\sigma; \sigma)$ of (8.32), obtain

$$\widetilde{a}_k = \frac{1}{\sqrt{\sigma N}} \sum_{r=0}^{N-1} \alpha_r \exp\left[i2\pi \frac{(k+u_\sigma)(r+v_\sigma)}{\sigma N} \right] = \frac{1}{\sqrt{\sigma N}} \sum_{r=0}^{N-1} \alpha_r \exp\left(i2\pi \frac{\widetilde{k}\widetilde{r}}{\sigma N} \right), \tag{8.34}$$

where $\widetilde{k} = k + u_\sigma$ and $\widetilde{r} = r + v_\sigma$. Then,

$$\begin{aligned}
\widetilde{a}_k &= \frac{1}{N\sqrt{\sigma}} \sum_{r=0}^{N-1} \left\{ \sum_{n=0}^{N-1} a_n \exp\left(i2\pi \frac{r+v_0}{N} \widetilde{n} \right) \right\} \exp\left(i2\pi \frac{r+v_\sigma}{\sigma N} \widetilde{k} \right) \\
&= \frac{1}{N\sqrt{\sigma}} \sum_{n=0}^{N-1} a_n \exp\left[i2\pi \left(\frac{\widetilde{n}v_0 + \widetilde{k}v_\sigma/\sigma}{N} \right) \right] \sum_{r=0}^{N-1} \exp\left(i2\pi \frac{\widetilde{n} + \widetilde{k}/\sigma}{N} r \right) \\
&= \frac{1}{N\sqrt{\sigma}} \sum_{n=0}^{N-1} a_n \frac{\sin\left[\pi(\widetilde{n} + \widetilde{k}/\sigma)\right]}{\sin\left[\pi(\widetilde{n} + \widetilde{k}/\sigma)/N\right]} \\
&\quad \times \exp\left\{ i2\pi \left[\frac{\widetilde{n}(v_0 + (N-1)/2) + (\widetilde{k}/\sigma)(v_\sigma + (N-1)/2)}{N} \right] \right\}.
\end{aligned} \tag{8.35}$$

With settings

$$\nu_0 = \nu_\sigma = -\frac{N-1}{2}, \tag{8.36}$$

we finally obtain

$$\widetilde{a}_k = \frac{1}{\sqrt{\sigma}} \sum_{n=0}^{N-1} a_n \frac{\sin\left[\pi(\widetilde{n}+\widetilde{k}/\sigma)\right]}{N\sin\left[\pi(\widetilde{n}+\widetilde{k}/\sigma)/N\right]} = \sum_{n=0}^{N-1} a_n \operatorname{sincd}\left\{N, \pi\left[(n+u_0) + \frac{k+u_\sigma}{\sigma}\right]\right\}, \tag{8.37}$$

which means discrete sinc-interpolated signal resampling and rescaling with respect to points u_0, u_σ, respectively.

As it is shown in Chapter 3, scaled DFT can be represented as a digital convolution, and therefore, by the convolution theorem, can be computed using FFT. In this way, we arrive at the *ScDFT-based image rescaling algorithm*:

$$\widetilde{a}_{\widetilde{k}} = \overrightarrow{\left\{\exp\left(-i\pi\frac{\widetilde{k}^2}{\sigma N}\right)\right\}} \bullet \operatorname{IFFT}_{[N\sigma]}\left\{\begin{array}{l} \operatorname{FFT}_{[N\sigma]}\left\{\operatorname{ZP}_{[N\sigma],N}\left\{\operatorname{SDFT}_{N,u_0}(a_k)\right\} \bullet \overrightarrow{\left\{\exp\left(-i\pi\frac{\widetilde{r}^2}{\sigma N}\right)\right\}}\right\} \\ \bullet\ \operatorname{FFT}_{[N\sigma]}\left[\overrightarrow{\left\{\exp\left(i\pi\frac{\widetilde{r}^2}{\sigma N}\right)\right\}}\right] \end{array}\right\}, \tag{8.38}$$

where $[N\sigma]$ is the smallest integer larger than N, $\operatorname{FFT}_{[N\sigma]}\{\cdot\}$ and $\operatorname{IFFT}_{[N\sigma]}\{\cdot\}$ are direct and inverse $[N\sigma]$-points FFT, $\operatorname{SDFT}_{N,u_0}(a_k)$ is shifted DFT defined by (8.33), $\bullet$ denotes elementwise (Hadamard) product of vectors, and $\operatorname{ZP}_{[N\sigma],N}\{\cdot\}$ is a zero-padding operator. If $\sigma > 1$, it places $([N\sigma] - N)$ zeros in the middle of the input sequence, as in the zero-padding algorithm, and if $\sigma < 1$ it nulls $(N - [N\sigma])$ corresponding spectral coefficients to implement lowpass filtering required for signal undersampling. To the best of the author's knowledge, such an algorithm of image rescaling was first published in [10]. Figure 8.14 illustrates examples of image resizing using ScDFT.

8.4.4. Image rotation and scaling using rotated 2D DFT

Second signal dimension adds additional degrees of freedom to signal transformations. For 2D signals, sampling of signals and their Fourier spectra can be carried out in the coordinates that are shifted, scaled, and rotated with respect to the signal and/or their spectra coordinates system. Taking this into account, one can, as

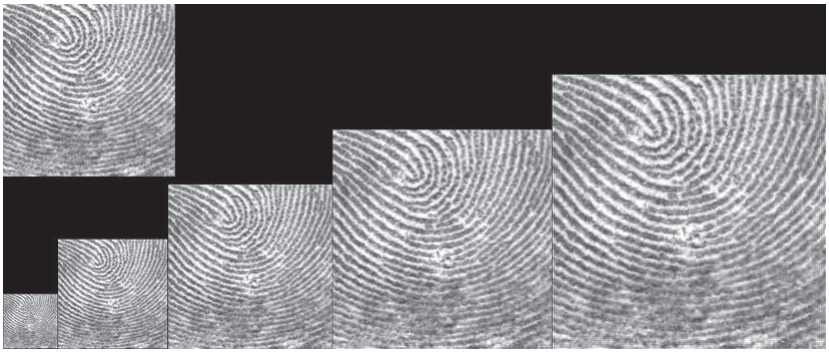


FIGURE 8.14. Examples of rescaling, using ScDFT algorithm of (8.38), of an image, shown in the left top corner, with scale factors $1/\pi$, $2/\pi$, $3/\pi$, $4/\pi$ and $5/\pi$ (bottom images, left to right).

it is shown in Chapter 3, arrive at "*rotated and scaled discrete Fourier transform*" (**RotScDFT**):

$$\alpha^{\theta}_{r,s} = \frac{1}{\sigma N}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} a_{k,l}\exp\left[i2\pi\left(\frac{\widetilde{k}\cos\theta+\widetilde{l}\sin\theta}{\sigma N}\widetilde{r} - \frac{\widetilde{k}\sin\theta-\widetilde{l}\cos\theta}{\sigma N}\widetilde{s}\right)\right], \tag{8.39}$$

where

$$\widetilde{k} = k + u^{(k)}_{\sigma}, \qquad \widetilde{l} = l + u^{(l)}_{\sigma}, \tag{8.40}$$

$u^{(k)}_{\sigma}$, $u^{(l)}_{\sigma}$, θ, and σ are signal sampling grid shifts, rotation angle, and scale factor with respect to the signal coordinate system, and

$$\widetilde{r} = r + v^{(r)}_{\sigma}, \qquad \widetilde{s} = s + v^{(s)}_{\sigma}, \tag{8.41}$$

$v^{(r)}_{\sigma}$ and $v^{(s)}_{\sigma}$ are Fourier spectrum sampling grid shifts.

Similar to the above-described signal shift or signal rescaling using shifted DFT or scaled DFT, one can use rotated DFT for image rescaling and rotation by applying RotDFT with appropriate scale and rotation angle parameters to signal SDFT spectrum. The following equations prove that in this rotation and rescaling separable discrete sinc-interpolation in rotated and scaled image coordinate is implemented.

Let

$$\alpha_{r,s} = \frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} a_{m,n}\exp\left[i2\pi\left(\frac{\widetilde{\widetilde{m}}\widetilde{\widetilde{r}}}{N} + \frac{\widetilde{\widetilde{n}}\widetilde{\widetilde{s}}}{N}\right)\right], \tag{8.42}$$

where $\widetilde{\widetilde{m}} = m + u^{(n)}_0$, $\widetilde{\widetilde{n}} = n + u^{(n)}_0$, $\widetilde{\widetilde{r}} = r + v^{(r)}_0$, $\widetilde{\widetilde{s}} = s + v^{(s)}_0$ are SDFTs of an image represented by its samples $\{a_{m,n}\}$. Apply to this spectrum rotated DFT defined by

(8.39), and obtain

$$\begin{aligned}
\widetilde{a}_{k,l} &= \frac{1}{\sigma N}\sum_{r=0}^{N-1}\sum_{s=0}^{N-1}\left\{\frac{1}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} a_{m,n}\exp\left[i2\pi\left(\frac{\widetilde{\widetilde{m}}\widetilde{\widetilde{r}}}{N}+\frac{\widetilde{\widetilde{n}}\widetilde{\widetilde{s}}}{N}\right)\right]\right\} \\
&\quad\times\exp\left[i2\pi\left(\frac{\widetilde{k}\cos\theta+\widetilde{l}\sin\theta}{\sigma N}\widetilde{r}-\frac{\widetilde{k}\sin\theta-\widetilde{l}\cos\theta}{\sigma N}\widetilde{s}\right)\right] = \frac{1}{\sigma N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} a_{m,n} \\
&\quad\times\left\{\sum_{r=0}^{N-1}\sum_{s=0}^{N-1}\exp\left[i2\pi\left(\frac{\widetilde{\widetilde{m}}+(\widetilde{k}\cos\theta+\widetilde{l}\sin\theta)/\sigma}{N}r+\frac{\widetilde{\widetilde{n}}-(\widetilde{k}\sin\theta-\widetilde{l}\cos\theta)/\sigma}{N}s\right)\right]\right\} \\
&\quad\times\exp\left[i2\pi\left(\frac{\widetilde{\widetilde{m}}v_0^{(r)}+(\widetilde{k}\cos\theta+\widetilde{l}\sin\theta)v_\sigma^{(r)}/\sigma}{N}+\frac{\widetilde{\widetilde{n}}v_0^{(s)}+(\widetilde{k}\sin\theta-\widetilde{l}\cos\theta)v_\sigma^{(s)}/\sigma}{N}\right)\right] \\
&= \frac{1}{\sigma N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} a_{m,n}\frac{\sin\{\pi[\widetilde{\widetilde{m}}+(\widetilde{k}\cos\theta+\widetilde{l}\sin\theta)/\sigma]\}}{\sin[\pi((\widetilde{\widetilde{m}}+(\widetilde{k}\cos\theta+\widetilde{l}\sin\theta)/\sigma)/N)]} \\
&\quad\times\frac{\sin\{\pi[\widetilde{\widetilde{n}}+(\widetilde{l}\cos\theta-\widetilde{k}\sin\theta)/\sigma]\}}{\sin[\pi((\widetilde{\widetilde{n}}+(\widetilde{l}\cos\theta-\widetilde{k}\sin\theta)/\sigma)/N)]} \\
&\quad\times\exp\left[i2\pi\frac{\widetilde{\widetilde{m}}(v_0^{(r)}+(N-1)/2)+(\widetilde{k}\cos\theta+\widetilde{l}\sin\theta)(v_\sigma^{(r)}+(N-1)/2)/\sigma}{N}\right] \\
&\quad\times\exp\left[i2\pi\frac{\widetilde{\widetilde{n}}(v_0^{(s)}+(N-1)/2)+(\widetilde{k}\sin\theta-\widetilde{l}\cos\theta)(v_\sigma^{(s)}+(N-1)/2)/\sigma}{N}\right],
\end{aligned} \tag{8.43}$$

or, with natural settings

$$\begin{aligned}
v_0^{(r)} &= v_\sigma^{(r)} = v_0^{(s)} = v_\sigma^{(s)} = -\frac{N-1}{2}, \\
\widetilde{a}_{k,l} &= \frac{1}{\sigma}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} a_{m,n}\,\mathrm{sincd}\left\{N;\pi\left[\widetilde{\widetilde{m}}+\frac{\widetilde{k}\cos\theta+\widetilde{l}\sin\theta}{\sigma}\right]\right\} \\
&\quad\times\mathrm{sincd}\left\{N;\pi\left[\widetilde{\widetilde{n}}+\frac{\widetilde{l}\cos\theta-\widetilde{k}\sin\theta}{\sigma}\right]\right\},
\end{aligned} \tag{8.44}$$

which means signal resampling in rotated and scaled coordinates with separable discrete sinc-interpolation. Figure 8.13(c) provides an experimental evidence of virtual equivalency, in terms of the interpolation accuracy, of image rotation using **RotDFT** and using DFT or DCT signal fractional shift-based three-pass rotation algorithm.

It is shown in Chapter 3 that **RotDFT** can, similar to **ScDFT**, be represented as a digital convolution and, therefore, can be efficiently computed using FFT. In

this way we arrive at the following image rescaling and rotation algorithm:

$$\{\widetilde{a}_{k,l}\} = \mathrm{IFFT}\,2_{\sigma N}\{\,\mathrm{FFT}\,2_{\sigma N}[\,\mathrm{ZP}_{[N\sigma],N}\,\{\,\mathrm{SDFT}_N\,(a_k)\}\bullet A_{r,s}]\bullet \mathrm{FFT}\,2_{\sigma N}[\,\mathrm{ChF}(r,s)]\}, \tag{8.45}$$

where $\mathrm{FFT}\,2_{[N\sigma]}[\cdot]$ and $\mathrm{IFFT}\,2_{[N\sigma]}[\cdot]$ are operators of direct and inverse $[N\sigma]$-point 2D FFT, $[N\sigma]$ is the smallest integer larger than $N\sigma$, $\mathrm{SDFT}_N(a_k)$ is defined by (8.33) and $\mathrm{ZP}_{[N\sigma],N}\{\cdot\}$ is the same zero-padding operator which was involved in above-presented image zooming algorithm and $\bullet$ denotes, as everywhere above, elementwise, or Hadamard product of vectors. A nonscale version of the described image rotation algorithm was suggested in [11].

To conclude this section, mention that, as this image rotation and rescaling algorithm is implemented through processing in DFT domain, one can combine rotation and scaling with adaptive image restoration and enhancement through nonlinear modification of its spectrum such as soft/hard thresholding, Pth low dynamic range compression and alike as it is described in Chapter 5. Such a combined algorithm is described by the equation

$$\begin{aligned}\{\widetilde{a}_{k,l}\} = \mathrm{IFFT}\,2_{\sigma N}\{\,\mathrm{PWNLT}\,\{\,\mathrm{FFT}\,2_{\sigma N}[\,\mathrm{ZP}_{[N\sigma],N}\,\{\,\mathrm{SDFT}_N\,(a_k)\}\bullet A_{r,s}]\\ \bullet\,\mathrm{FFT}\,2_{\sigma N}[\,\mathrm{ChF}(r,s)]\}\},\end{aligned} \tag{8.46}$$

where $\mathrm{PWNLT}\{\cdot\}$ is the above-mentioned pointwise nonlinear spectrum transformation.

Figure 8.15 illustrates this option. It shows a result of a simultaneous image rotation and scaling and a result of rotation, scaling, denoising by hard thresholding in spectral domain combined with P-law transformation of absolute values of image spectral coefficients with $P = 0.5$ as it is described in Chapter 5.

8.4.5. Numerical differentiation and integration

Signal numerical differentiation and integration are operations that require measuring infinitesimal increments of signals and their arguments. Therefore, numerical computing signal derivatives and integrals assume one of another method of building continuous models of signals specified by their samples through explicit or implicit interpolation between available signal samples. Because differentiation and integrations are shift invariant linear operations, methods of computing signal derivatives and integrals from their samples can be conveniently designed and compared in the Fourier transform domain.

Let Fourier transform spectrum of continuous signal $a(x)$ is $\alpha(f)$:

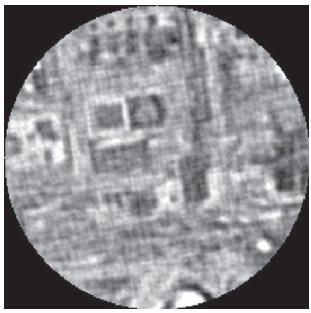
$$a(x) = \int_{-\infty}^{\infty} \alpha(f)\exp(-i2\pi f x)df. \tag{8.47}$$

Then, Fourier spectrum of its derivative

$$\dot{a}(x) = \frac{d}{dx}a(x) = \int_{-\infty}^{\infty}[(-i2\pi f)\alpha(f)]\exp(-i2\pi f x)df \tag{8.48}$$

(a)

(b)



(c)

FIGURE 8.15. Image re-scaling, rotation, denoising and enhancement using RotScDFT: (a) initial image; (b) 10°-rotated, 1.7 times magnified image; (c) 10°-rotated, 1.7 times magnified, denoised, and enhanced image.

will be $(-i2\pi f)\alpha(f)$ and Fourier spectrum of its integral

$$\overline{a}(x) = \int a(x)dx = \int_{-\infty}^{\infty} \left[\left(-\frac{1}{i2\pi f} \right) \alpha(f) \right] \exp(-i2\pi f x)df \tag{8.49}$$

will be $\alpha(f)/(-i2\pi f)$. From Fourier transform convolution theorem it follows that signal differentiation and integration can be regarded as signal linear filtering with filter frequency responses, correspondingly

$$H_{\text{diff}} = -i2\pi f, \qquad H_{\text{int}} = \frac{i}{2\pi f}. \tag{8.50}$$

Let now signal $a(x)$ be represented by its samples $\{a_k\}$, $k = 0, 1, \ldots, N-1$, and let $\{\alpha_r\}$ be a set of DFT coefficients of discrete signal $\{a_k\}$:

$$a_k = \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \alpha_r \exp\left(-i2\pi \frac{kr}{N} \right). \tag{8.51}$$

Then, following the argumentation of Section 8.2 for the optimal resampling filter, one can conclude that samples $\{\eta_{r,\text{opt}}^{(\text{diff})}\}$ and $\{\eta_{r,\text{opt}}^{(\text{int})}\}$ of continuous frequency

response of numerical differentiation and integration representation are defined as

$$\eta_r^{(\mathrm{diff})} = \begin{cases} -\dfrac{i2\pi r}{N}, & r = 0, 1, \ldots, \dfrac{N}{2} - 1, \\ -\dfrac{\pi}{2}, & r = \dfrac{N}{2}, \\ \dfrac{i2\pi(N-r)}{N}, & r = \dfrac{N}{2} + 1, \ldots, N-1, \end{cases} \tag{8.52a}$$

$$\eta_{r,\mathrm{opt}}^{(\mathrm{int})} = \begin{cases} 0, & r = 0, \\ \dfrac{iN}{2\pi r}, & r = 1, \ldots, \dfrac{N}{2} - 1, \\ -\dfrac{\pi}{2}, & r = \dfrac{N}{2}, \\ \dfrac{iN}{2\pi(N-r)}, & r = \dfrac{N}{2} + 1, \ldots, N-1, \end{cases} \tag{8.52b}$$

for even $N$and

$$\eta_r^{\mathrm{diff}} = \begin{cases} -\dfrac{i2\pi r}{N}, & r = 0, 1, \ldots, \dfrac{N-1}{2} - 1, \\ \dfrac{i2\pi(N-r)}{N}, & r = \dfrac{N+1}{2}, \ldots, N-1, \end{cases} \tag{8.52c}$$

$$\eta_r^{(\mathrm{int})} = \begin{cases} \dfrac{iN}{2\pi r}, & r = 0, 1, \ldots, \dfrac{N-1}{2} - 1, \\ \dfrac{iN}{2\pi(N-r)}, & r = \dfrac{N+1}{2}, \ldots, N-1, \end{cases} \tag{8.52d}$$

for odd N.

One can show that numerical differentiation and integration according to (8.52) imply discrete sinc-interpolation of signals. Note that the coefficients $\eta_{N/2}^{(\mathrm{diff})}$ and $\eta_{N/2}^{(\mathrm{int})}$ in (8.52a), (8.52b) are halved that correspond to the above-described Case 1 of discrete sinc-interpolation (8.23d).

Equations (8.52) imply the following algorithmic implementation for computing derivatives and integrals of signals specified by their samples

$$\{\dot{a}_k\} = \mathrm{IFFT}_N\left(\{\eta_{r,\mathrm{opt}}^{(\mathrm{diff})}\} \bullet \mathrm{FFT}\left(\{a_k\}\right)\right), \tag{8.53a}$$

$$\{\overline{a}_k\} = \mathrm{IFFT}_N\left(\{\eta_{r,\mathrm{opt}}^{(\mathrm{int})}\} \bullet \mathrm{FFT}\left(\{a_k\}\right)\right). \tag{8.53b}$$

Thanks to the use of fast Fourier transform, the computational complexity of the algorithms is $O(\log N)$ operations per signal sample. Digital filter described by (8.53a) is called *discrete ramp-filter*.

Likewise all DFT-based discrete sinc-interpolation algorithms, DFT-based differentiation and integration algorithms, being the most accurate in term of preserving signal spectral components within the baseband, suffer from boundary effects. Obviously, DFT-based differentiation is especially vulnerable in this respect.

This drawback can be efficiently overcome by means of even extension of the signals to double length through mirror reflection at their boundaries before applying above-described DFT-based algorithms. For such extended signals, DFT-based differentiation and integration are reduced to using fast DCT and IDcST algorithms instead of FFT:

$$\{\dot{a}_k\} = \mathrm{IDcST}\left(\{\eta_{r,\mathrm{opt}}^{(\mathrm{diff})}\} \bullet \mathrm{DCT}\left(\{a_k\}\right)\right) = \frac{2\pi}{N\sqrt{2N}} \sum_{r=1}^{N-1} r\alpha_r^{(\mathrm{DCT})} \sin\left(\pi \frac{k+1/2}{N} r\right); \tag{8.54a}$$

$$\{\overline{a}_k\} = \mathrm{IDcST}\left(\{\eta_{r,\mathrm{opt}}^{(\mathrm{diff})}\} \bullet \mathrm{DCT}\left(\{a_k\}\right)\right) = \frac{\sqrt{N}}{2\pi\sqrt{2}} \sum_{r=1}^{N-1} \frac{\alpha_r^{(\mathrm{DCT})}}{r} \sin\left(\pi \frac{k+1/2}{N} r\right), \tag{8.54b}$$

with the same computational complexity of $O(\log N)$ operations per signal sample.

In numerical mathematics, alternative methods of numerical computing signal derivatives and integrals are common that are implemented through signal discrete convolution in the signal domain

$$\dot{a}_k = \sum_{n=0}^{N_h-1} h_n^{\mathrm{diff}} a_{k-n}, \qquad \dot{a}_k = \sum_{n=0}^{N_h-1} h_n^{\mathrm{int}} a_{k-n}. \tag{8.55}$$

The following simplest differentiating kernels of two and five samples are recommended in manuals on numerical methods (see [12]):

$$h_n^{\mathrm{diff}(1)} = [-1, 1], \tag{8.56a}$$

$$h_n^{\mathrm{diff}(2)} = \left[-\frac{1}{12}, \frac{8}{12}, 0, -\frac{8}{12}, \frac{1}{12}\right]. \tag{8.56b}$$

Both are based on an assumption that in the vicinity of signal samples signals can be expanded into Taylor series.

Most known numerical integration methods are the Newton-Cotes quadrature rules (see [13]). The three first rules are the trapezoidal, the Simpson, and the 3/8-Simpson ones. In all the methods, the value of the integral in the first point is not defined because it affects to the result constant bias and should be arbitrarily chosen. When it is chosen to be equal to zero, the trapezoidal, Simpson, and 3/8-Simpson numerical integration methods are defined, for k as a running sample

index, by equations

$$\overline{a}_1^{(T)} = 0, \qquad \overline{a}_k^{(T)} = \overline{a}_{k-1}^{(T)} + \frac{1}{2}(a_{k-1} + a_k), \tag{8.57a}$$

$$\overline{a}_1^{(S)} = 0, \qquad \overline{a}_k^{(S)} = \overline{a}_{k-2}^{(S)} + \frac{1}{3}(a_{k-2} + 4a_{k-1} + a_k), \tag{8.57b}$$

$$\overline{a}_0^{(3/8S)} = 0, \qquad \overline{a}_k^{(3/8S)} = \overline{a}_{k-3}^{(3/8S)} + \frac{3}{8}(a_{k-3} + 3a_{k-2} + 3a_{k-1} + a_k). \tag{8.57c}$$

Another frequently used alternative is cubic spline integration (see [14]), for which

$$\overline{a}_{k+1}^{(\mathrm{int,CS})} - \overline{a}_k^{(\mathrm{int,CS})} = \frac{1}{2}(a_k + a_{k+1}) - \frac{1}{24}(m_k + m_{k+1}), \tag{8.57d}$$

where coefficients $\{m_k\}$ are determined by the system of linear equations

$$\{(m_{k-1} + 4m_k + m_{k+1}) = 6(a_{k+1} - 2a_k + a_{k-1})\}. \tag{8.57e}$$

In these integration methods, a linear, a quadratic, a cubic and a cubic spline interpolation, respectively, are assumed between the sampled slope data. In the cubic spline interpolation, a cubic polynomial is evaluated between every couple of points (see [14]), and then an analytical integration of these polynomials is performed.

As it was shown in Section 8.2.2, discrete frequency responses of digital filters completely determine their continuous and overall frequency responses. Compare discrete frequency responses of above-described numerical differentiation and integration methods. Applying N-point discrete Fourier transform to (8.57), obtain

$$\begin{aligned} \eta_r^{\mathrm{diff}(1)} &\propto \sin\left(\frac{\pi r}{N}\right), \\ \eta_r^{\mathrm{diff}(2)} &\propto \frac{8\sin(2\pi r/N) - \sin(4\pi r/N)}{12}, \end{aligned} \tag{8.58}$$

$$\begin{aligned} \eta_r^{(\mathrm{int},T)} &= \frac{\overline{\alpha}_r^{(Tr)}}{\alpha_r} = \begin{cases} 0, & r = 0, \\ -\dfrac{\cos(\pi r/N)}{2i\sin(\pi r/N)}, & r = 1,\dots,N-1, \end{cases} \\ \eta_r^{(\mathrm{int},S)} &= \frac{\overline{\alpha}_r^{(S)}}{\alpha_r} = \begin{cases} 0, & r = 0, \\ -\dfrac{\cos(2\pi r/N) + 2}{3i\sin(2\pi r/N)}, & r = 1,\dots,N-1, \end{cases} \\ \eta_r^{(\mathrm{int},3S)} &= \frac{\overline{\alpha}_r^{(3S)}}{\alpha_r} = \begin{cases} 0, & r = 0, \\ -\dfrac{\cos(3\pi r/N) + 3\cos(\pi r/N)}{i\sin(3\pi r/N)}, & r = 1,\dots,N-1, \end{cases} \\ \eta_r^{(\mathrm{int,CS})} &= \begin{cases} 0, & r = 0, \\ -\dfrac{1}{4i}\dfrac{\cos(\pi r/N)}{\sin(\pi r/N)}\left[1 + \dfrac{3}{\cos(2\pi(r/N)) + 2}\right], & r = 1,\dots,N-1. \end{cases} \end{aligned} \tag{8.59}$$

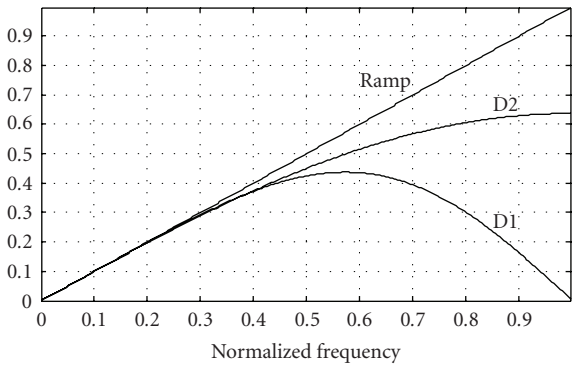


Figure 8.16. Absolute values of frequency responses of differentiation filters described by (8.56a) (curve D1), (8.56b) (curve D2), and (8.52c) ("Ramp"-filter).

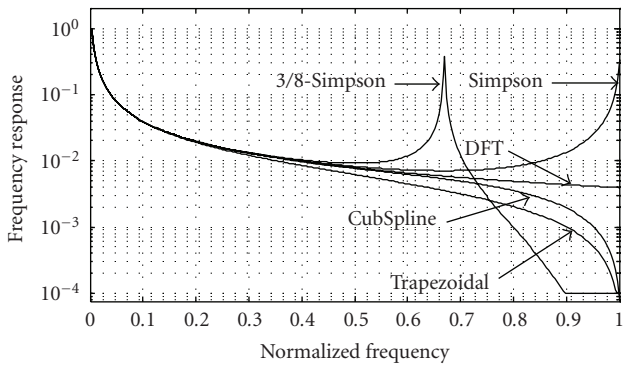


Figure 8.17. Absolute values of frequency responses of numerical integration filters described by (8.59) and of DFT-based method (8.52b) and (8.52d).

These frequency responses along with the frequency responses of DFT-based differentiation and integration filters (8.52a) and (8.52c) are shown, for comparison, in Figures 8.16 and 8.17.

One can see from the figure that common numerical differentiation and integration methods entail certain and sometimes very substantial distortions of signal spectral contents on high frequencies. All of them attenuate signal high frequencies and Simpson and 3/8-Simpson integration methods, being slightly more accurate than trapezoidal method in the middle of the signal baseband, tend even to generate substantial artifacts if signals contain higher frequencies. Frequency response of the 3/8-Simpson rule tends to infinity for the 2/3 of the maximum frequency, and the frequency response of the Simpson rule has almost the same tendency for the maximal frequency in the baseband. This means, in particular, that round-off computation errors and noise that might be present in input data will be over amplified by Simpson and 3/8-Simpson in these frequencies.

Quantitative evaluation of the superiority of DFT/DCT-based differentiation methods can be gained from experimental data illustrated in Figures 8.17–8.19,

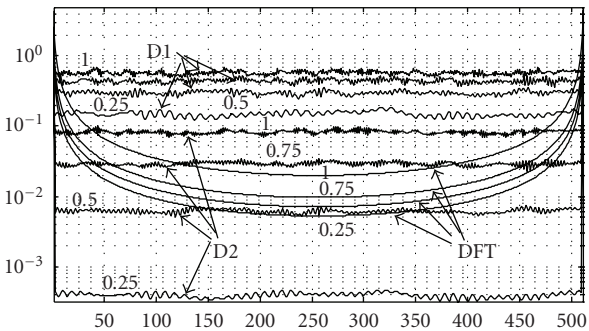


FIGURE 8.18. Experimental data on samplewise normalized standard deviation of the differentiation error for D1, D2, and DFT methods. Numbers at curves indicate fraction (from quarter to one) of test signal bandwidth with respect to the base bandwidth as defined by the sampling rate.

which present results obtained by computer statistical simulation of differentiation of realizations of pseudorandom signals with uniform spectrum in the range 1/16 of the baseband the entire baseband. In the simulation, 16 series of statistical experiments with 100 experiments in each run were carried out. In the runs, pseudorandom signals of 32 704 samples with uniform spectrum were generated in order to imitate continuous signals. In each Kth run, generated pseudorandom signal was lowpass filtered to $1/32K$ of its baseband using ideal lowpass filter implemented in DFT domain, and its derivative was computed using DFT domain ramp-filter. The filtered signal was used as a model of a continuous signal and its derivative was used as an estimate of its ideal derivative. Then, a central half of this signal that encompasses 16 352 samples taken 8196 samples apart from the signal borders was subsampled to generate 511 samples of a test discrete signal used in the differentiation by the three methods defined by (8.56a) (D1 method), (8.56b) (D2 method), (8.52c) (DFT method), and (8.54a) (DCT method). Corresponding central part of the ideal derivative signal was also correspondingly subsampled and was used as a reference to evaluate differentiation error for tested methods. Differential error was computed as a difference between the "ideal" derivative and results of applying tested differentiation methods divided by standard deviation of the "ideal" derivative over 511 samples. Finally, standard deviation of the normalized error over 100 realizations was found for each signal sample. Obtained results are plotted samplewise in Figure 8.18 for methods D1, D2, and DFT and in Figure 8.19 for methods D2, DFT, and DCT.

In Figure 8.18, one can see that, indeed, the simplest method D1 performs very poorly, while method D2 outperforms DFT method for signals with bandwidth less than 0.5 of the baseband. Also can be seen that accuracy of the DFT differentiation method improves with the distance from signal boundaries. However, even for samples that are far away from signal boundaries, boundary effects badly deteriorate the differentiation accuracy. Data presented in Figure 8.19 evidence that DCT method does successfully overcome the boundary effect problem

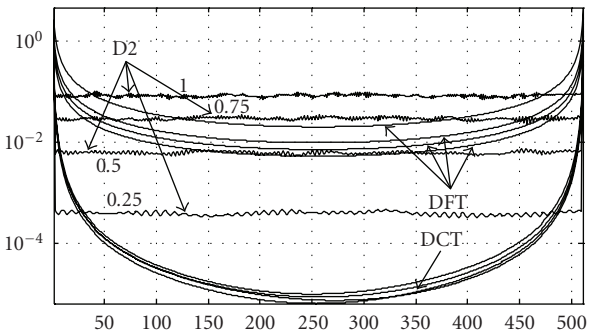


FIGURE 8.19. Experimental data on samplewise normalized standard deviation of the differentiation error for D2, DFT, and DCT methods. Numbers at curves indicate fraction (from quarter to one) of test signal bandwidth with respect to the base bandwidth as defined by the sampling rate.

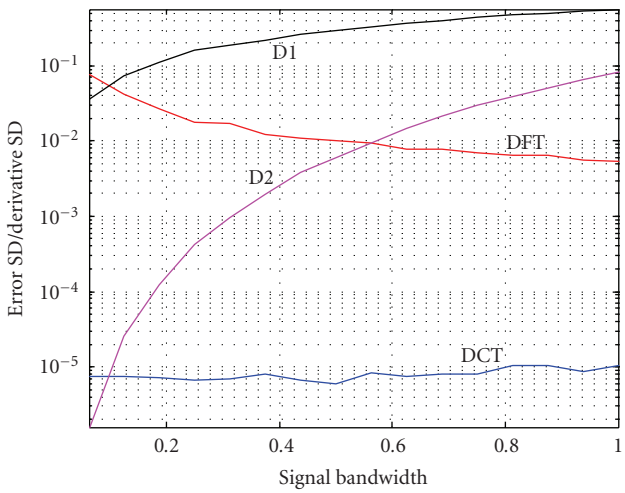


FIGURE 8.20. Normalized standard deviation of the differentiation error averaged over 100 samples in the middle of the signal (samples 200–300) for D1, D2, DFT, and DCT differentiation methods as a function of test signal bandwidth (in fractions of the baseband defined by the signal sampling rate).

and substantially outperforms both D1 and D2 methods even for narrowband signals.

In the described experiment, average mean square differentiation error computed over the central half of the test signals with different bandwidth was also measured for different methods. The results presented in Figure 8.20 convincingly evidence that method D2 provides better accuracy only for signals with bandwidth less than 0.05 of the baseband, and even for such signals normalized error standard deviation for DCT method is anyway less than 10^{-5}. For signals with broader bandwidth, the accuracy of the DCT differentiation method outperforms other methods at least by 2 order of magnitude. It was also found in this experiment that masking signal with a window function that gradually nulls signal samples

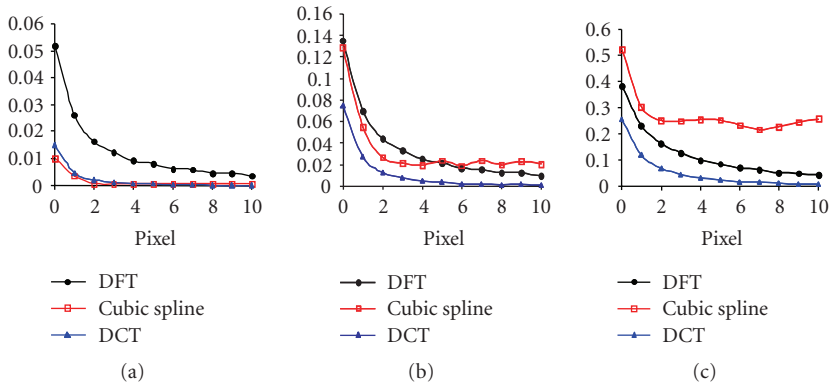


FIGURE 8.21. Integration error versus sample index for DFT-based, cubic spline, and DCT-based numerical integration methods for sinusoidal signals with normalized initial frequencies 0.27 (a), 0.54 (b), and 0.81 (c).

in the vicinity of its border substantially improves the differentiation accuracy of both DFT and DCT methods even further. To complete this comparison of differentiation methods, note also that, as it follows from above results, conventional numerical methods maintain good differentiation accuracy if signals are highly oversampled, which actually undermines their only advantage, that of low computational complexity.

In order to illustrate and compare, in quantitative terms, performance of different numerical integration methods, we present in Figures 8.21 and 8.22 experimental results reported in [15] on integration accuracy for test sinusoidal signals of different frequencies for cubic spline, DFT-based and DCT-based integration methods which are the most accurate among above-described methods. The number of signal samples N was 256, 512, and 1024 and, for each integer frequency index $\mathbf{r}$, results for 20 sinusoidal signals with frequencies in the range $[r - 1/2 : r + 1/2]$ taken with step 1/20 were averaged to obtain root mean square integration error for each signal sample. The error was evaluated with respect to the analytical integral value for the correspondent signal.

Figure 8.21(a) shows mean square integration error for first 10 samples of sinusoidal signals with $N = 256$ and frequency indices 35, 70, and 105 (0.27, 0.54, and 0.81 of the highest frequency in the baseband, correspondingly). As one can see, for low-signal frequency, DFT method underperforms cubic spline one due to boundary effects while, for higher frequencies, it provides higher accuracy and its advantage grows with signal frequency. One can also see that DCT-based method suffers from boundary effects not more than cubic spline one and substantially outperforms the latter for all signal samples beginning 4-5th ones.

Figure 8.22 shows similar data obtained for sinusoidal signals with number of samples $N = 256$, 512, and 1024 and normalized frequencies in the range $[0.54 \pm 1/N]$. They show that boundary effects do not practically depend on the number N of samples and that they affect only first 10 or so signal samples.

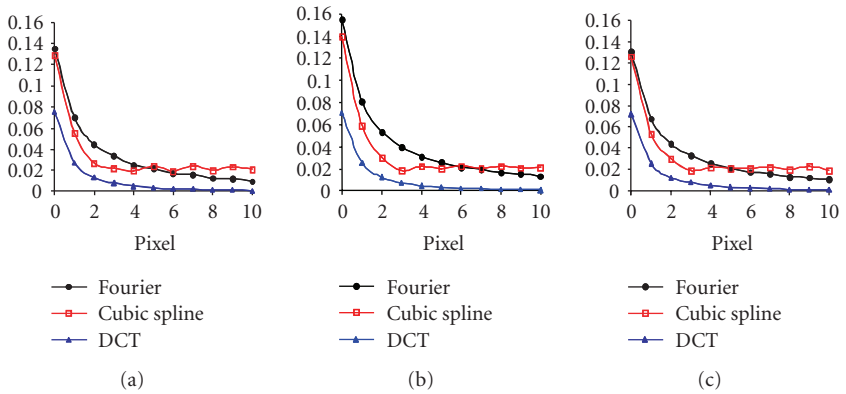


FIGURE 8.22. Integration error versus sample index for DFT-based, cubic spline, and DCT-based numerical integration methods for sinusoidal signal with normalized initial frequency 0.54 for sinusoidal signals with different number of samples N: (a) $N = 256$, (b) $N = 512$, (c) $N = 1024$.

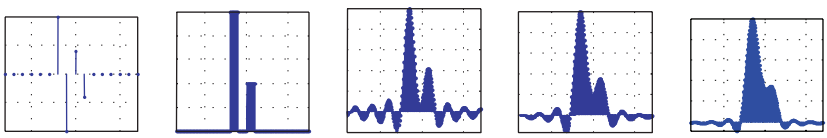


FIGURE 8.23. Comparison of the integrator resolving power: (a) integrator input; (b) output of the ideal integrator with unlimited bandwidth; (c) output of the DFT-based integrator; (d) output of the cubic spline integrator; (e) output of the trapezoidal integrator.

Further insight into metrological advantages of DFT/DCT-based numerical integration methods over other methods may be gained from comparison of the integration resolving power. Resolving power of integrators characterizes their capability to resolve between close sharp impulses in the input data. Though it is fully defined by the integrator frequency responses, it is much more straightforward to compare the resolving power for different integrators directly in signal domain.

Figure 8.23 illustrates results of a numerical evaluation of the capabilities of three types of integrators, trapezoidal, cubic spline, and DFT-based ones, to resolve two sharp impulses placed on the distance of one sampling interval one from another for the case when the second impulse is of half height of the first impulse (see [15]). The signals shown in the figure are 8 times subsampled and sincd-interpolated to imitate the corresponding continuous signals at the integrator output. The figure clearly shows how much the tested integrators differ in their resolving power. DFT-based integrator (Figure 8.23(c)) produces the sharpest peaks with the lowest valley between signal peaks compared to that for cubic spline integrator (Figure 8.23(d)) and trapezoidal integrator (Figure 8.23(e)). In particular, the latter seems to be virtually incapable of reliable resolving the half-height impulse.

DFT and especially DCT-based differentiation and integration methods being the most accurate numerical differentiation and integration methods can find numerous applications in optical metrology and experimental data processing. Signal differentiating by ramp-filtering has found its application in digital image reconstruction from its projections by the filtered back projection method. The DCT-based *discrete filtered back projection image reconstruction algorithm* of image reconstruction that uses the DCT-based ramp-filter and image rotation algorithm described in Section 8.4.2 is as follows:

$$\{\hat{a}_{k,l}\} = \sum_{s=0}^{N_\theta - 1} \mathrm{ROT}_\theta \left\{ \{\dot{p}_k^{(\theta_s)}\} \otimes \{\vec{1}_l\} \right\}; \quad k, l = 0, 1, \ldots, N-1, \tag{8.60}$$

where

$$\{\dot{p}_k^{(\theta_s)}\} = \mathrm{IDcST}\left(\{\eta_r^{\mathrm{diff}}\} \bullet \mathrm{DCT}\left(\{p_k^{(\theta_s)}\}\right)\right) \tag{8.61}$$

is a vector-row of samples of ramp-filtered image projection $\{p_k^{(\theta_s)}\}$ obtained for angle θ_s, $s = 0, 1, \ldots, N_\theta - 1$, $\{\vec{1}_l\}$ is a vector-column of N ones and $\otimes$ symbolizes matrix Kronecker product.

In addition to higher accuracy, DFT/DCT-based differentiation and integration methods have one more property that makes them ideally suited for differentiation/integration of noisy data. According to the theory of optimal Wiener filtering, when input data contain noise, frequency responses H_{diff} and H_{int} of ideal differentiation and integration devices defined by (8.50) should be modified with an account to signal-to-noise ratio in the data to become

$$\begin{aligned} H_{\mathrm{diff}}^{(\mathrm{noise})} &= H_{\mathrm{diff}} \frac{\mathrm{SNR}(f)}{1 + \mathrm{SNR}(f)}, \\ H_{\mathrm{int}}^{(\mathrm{noise})} &= H_{\mathrm{int}} \frac{\mathrm{SNR}(f)}{1 + \mathrm{SNR}(f)}, \end{aligned} \tag{8.62}$$

where $\mathrm{SNR}(f)$ is input data signal-to-noise ratio on frequency f. As DFT/DCT-based differentiation and integration methods are designed and implemented in signal spectral domain, corresponding modification of their discrete frequency responses (8.52) is straightforward and their implementation does not require any additional data processing.

8.4.6. Computational complexity and other practical issues

One of the fallacies regarding discrete sinc-interpolation is a widely shared opinion of its high computational complexity. Indeed, the most known DFT spectrum zero-padding algorithms have a relatively high computational complexity and are very inflexible. Being implemented using traditional FFT algorithms, they require, for L-fold equidistant subsampling (zooming) of signals of N samples, $O(\log LN)$

of operations (additions and multiplications of complex numbers) per output signal sample and require subsampling factor L to be power of 2. Such an implementation involves a vast amount of operations with zero data. However, these operations can be removed using, for inverse FFT of zero padded data, pruned FFT algorithms (see [6]). Similar pruned algorithms for fast DCT are also available (see [16]). For large L, this can substantially reduce the computational complexity of the DFT/DCT spectra zero-padding algorithms. However, it does not improve the algorithm flexibility. Above described DFT- and DCT-based signal fractional shift algorithms completely eliminate those drawbacks of zero-padding algorithms.

Computational complexity of subsampling (zooming), with arbitrary subsampling factor, of 1D signals of N samples using DFT- and DCT-based signal fractional shift algorithms is $O(\log N)$ per output sample. One can compare them with the computational complexity of spline-based resampling algorithms whose computational complexity is, by the order of magnitude, $O(n)$, where n is a value proportional to the spline order. For instance, for mentioned Mems$(5, 3, 1)$, $n = 9$, which means that, by the order of magnitude, its complexity is comparable with that of DFT(DCT)-based fractional shift algorithm for $N = 512$.

Computational complexity of image zooming depends on whether zooming is carried out in a separable (rows-column) or in an inseparable way. In the separable implementation, $L_x \times L_y$-zooming images of $N_x \times N_y$ pixels, requires $O(L_x N_x N_y \log N_x)$ operations (L_x times N_x-point FFTs for each of N_y image rows) for zooming along coordinate x plus $O(L_y L_x N_x N_y \log N_y)$ operations (L_y times N_y-point FFTs for each of $L_x N_x$ 1D zoomed image rows) for zooming along coordinate y. Therefore, the per-output sample computational complexity of separable image $L_x \times L_y$-zooming can be estimated as being $O(\log N_y + \log N_x / L_y N_x)$, which means that the computational complexity of the separable image zooming using DFT/DCT-based signal fractional shift algorithms is determined mainly by the lowest dimensionality of the image. Inseparable implementation requires performing $L_x L_y$ times $N_x N_y$-point FFTs. Therefore, the per-output pixel computational complexity will be higher: $O(\log N_x N_y)$ operations. Note, however, that inseparable zooming allows zooming with simultaneous arbitrary inseparable 2D spectrum shaping and filtering.

Three-step image rotation implemented through the use of DFT/DCT-based signal fractional shift algorithm requires, for the image of $N_x \times N_y$ pixels, $O(2N_x N_y \log N_x)$ operations for each of two-image shearing in x direction plus $O(2N_x N_y \log N_y)$ operations for image shearing in y direction which amounts $O(2 \log N_x + \log N_y)$ operations. The same, by the order of magnitude, is estimation of computational complexity of fast Radon transform and fast filtered back projection inverse Radon transform as well as of DFT/DCT-based differentiation and integration methods.

Scaled DFT-based image zooming algorithm is computationally more expensive than image zooming by means of DFT/DCT-based signal fractional shift algorithm, especially for large zoom factors σ as it involves two $[N\sigma]$-point FFTs instead of one N-point FFT. Yet its capability to implement zooming with arbitrary

noninteger factors makes it an attractive option when accurate image zooming with zoom-factor in the range 0.5–2 is required.

Using, for image rotation, of **RotScDFT**-based rotation algorithm has no advantage with respect to the three-step rotation algorithm that uses DFT/DCT-based signal fractional shift. It might be useful only when both image rotation and rescaling are needed or when image data are already given in Fourier spectral domain, such as in NMR imaging.

Above-presented estimations of computational complexity of DFT/DCT-based discrete sinc-interpolation resampling methods are estimations by the order of magnitude. Exact figures of the computational complexity depend on particular software implementation, compiler efficiency and of the type of the computer CPU. Certain insight into comparison of real processing speed for different interpolation methods can be gained from the following data obtained in course of above-described experiments with image rotation using different resampling methods. Matlab m-file that implements the three-step rotation algorithm using a DCT-based signal fractional shift Matlab dll-file subroutine, and a Windows exe-file that implements Mems(5, 3, 1)-based rotation algorithm both run, on the same computer, about 7 seconds for 100 rotations of 256×256 pixel images. The proper Matlab program **imrotate.m** of Matlab image processing tool box runs, for bicubic interpolation, slower (17.67 seconds for 60 rotations) than a Matlab m-file that implements, using only standard Matlab means, three-step rotation algorithm with DFT-based signal fractional shift (14.16 seconds for 60 rotations).

All described sinc-interpolation algorithms implement processing in DFT/DCT domain. This provides to all of them an additional capability, not available in other convolution-based interpolation methods, the capability of simultaneous signal/image filtering (denoising, deblurring, enhancement, etc.) as it was demonstrated in Figure 8.15 on the example of image rotation and rescaling using RotScDFT. There are also some specific application areas, such as magnetic resonance imaging and digital holography where data are already given in Fourier domain and where using above described algorithms is only natural and straightforward. A good example is numerical reconstruction of digitally recorded holograms in which practically all reconstruction algorithms are using FFT and it is very frequently necessary to maintain special image scaling conditions (see Chapter 3).

8.5. Local (elastic) image resampling: sliding window discrete sinc-interpolation algorithms

It frequently happens in signal and image resampling tasks that require arbitrary different signal sample shifts for different samples. A typical example is image resampling for stabilization of images observed through turbulent atmosphere, which causes chaotic positioning of image samples with respect to their regular equidistant positions [17]. In such cases, above-described discrete sinc-interpolation algorithms have no efficient computational implementation. However, in these applications they can be implemented in sliding window (see [6–8]).

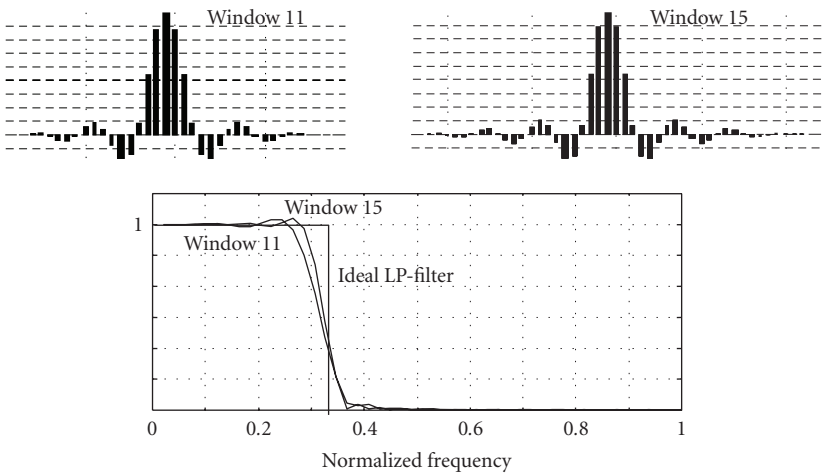


FIGURE 8.24. Windowed discrete sinc-functions for windows of 11 and 15 samples (top) and the corresponding interpolator frequency responses along with that of the ideal lowpass filter (bottom) for signal 3x-zooming.

In signal interpolation in sliding window, only those shifted and interpolated signal samples that correspond to the window central sample have to be computed in each window position from signal samples within the window. Interpolation function in this case is a discrete sinc-function whose extent is equal to the window size rather that to the whole signal size required for the perfect discrete sinc-interpolation. Therefore, sliding window discrete sinc-interpolation cannot provide the perfect interpolation, which above-described global discrete sinc-interpolation does. Figure 8.24 illustrates frequency responses and PSF of the corresponding interpolating lowpass filters of two different window sizes for signal 3x-zooming. As one can see from the figure, interpolation filter frequency responses deviate from a rectangular function, a frequency response of the ideal lowpass filter.

Such an implementation of the discrete sinc-interpolation can be regarded as a special case of direct signal domain convolution interpolation methods. As it follows from the above theory, it, in principle, has the highest interpolation accuracy among all convolution interpolation methods with the same window size. Additionally, it offers valuable options that are not available with other methods, specifically:

(i) signal resampling with arbitrary shifts and simultaneous signal restoration and enhancement and

(ii) local adaptive interpolation with "superresolution."

For signal resampling with simultaneous restoration/enhancement, the sliding window discrete sinc-interpolation should be combined with local adaptive filtering. Local adaptive filters, in each position k of the window of W samples (usually an odd number), compute transform coefficients $\{\beta_r = T\{a_n\}\}$ of the signal $\{a_n\}$ in the window $(n, r = 1, 2, \ldots, W)$ and nonlinearly modify them to obtain

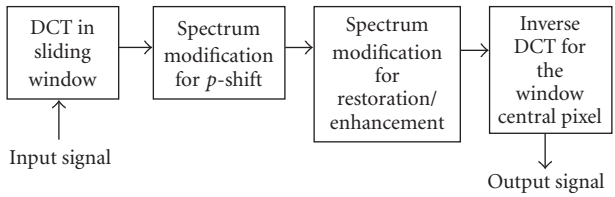


FIGURE 8.25. Flow diagram of signal resampling combined with denoising/enhancement.

(a) (b)

FIGURE 8.26. An example of image denoising and resampling from irregular to regular sampling grid: (a) noisy image sampled in an irregular sampling grid; (b) denoised and resampled (rectified) image.

coefficients $\{\hat{\alpha}_r(\alpha_r)\}$. These coefficients are then used to generate an estimate $\hat{a}_k$ of the window central pixel by inverse transform $T_k^{-1}\{\cdot\}$ computed for the window central pixel as

$$\hat{a}_k = T_k^{-1}\{\hat{\alpha}_r(\beta_r)\}, \tag{8.63}$$

such a filtering can be implemented in the domain of any transform though the DCT has proved to be one of the most efficient (see Chapter 5). Therefore, one can, in a straightforward way, combine the sliding window DCT domain discrete sinc-interpolation signal resampling (8.20) and filtering for signal restoration and enhancement:

$$\{a_k^p\} = \mathrm{ISDFT}_{1/2,0}\{\hat{\alpha}_r^{\mathrm{DCT}}(\alpha_r^{\mathrm{DCT}}) \cdot \eta_r(p)\}. \tag{8.64}$$

Figure 8.25 shows flow diagram of such a combined algorithm for signal restoration/enhancement and fractional p-shift.

Figure 8.26 illustrates application of the combined filtering/interpolation for image irregular to regular resampling combined with denoising. In this example, left image is distorted by known displacements of pixels with respect to regular equidistant positions and by an additive noise. In the right image, these displacements are compensated and noise is substantially reduced with the above-described sliding window resampling/denoising algorithm.

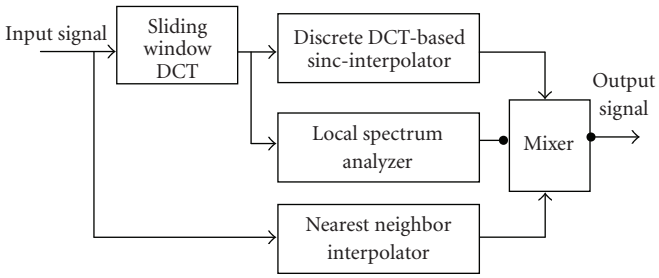


FIGURE 8.27. Flow diagram of local adaptive resampling.

One can further extend the applicability of this method to make interpolation kernel transform coefficients $\{\eta_r(p)\}$ in (8.64) to be adaptive to signal local features that exhibit themselves in signal local DCT spectra:

$$\{a_k^p\} = \mathrm{ISDFT}_{1/2,0}\{\hat{\alpha}_r^{\mathrm{DCT}}(\beta_r^{\mathrm{DCT}}) \cdot \eta_r(p, \{\beta_r^{\mathrm{DCT}}\})\}. \tag{8.65}$$

The adaptivity may be desired in such applications as, for instance, resampling images that contain gray tone images in a mixture with graphical data. While the discrete sinc-interpolation is completely perfect for gray tone images, it may produce undesirable oscillating artifacts in graphics.

The principle of local adaptive interpolation is schematically presented on Figure 8.27. It assumes that modification of signal local DCT spectra for signal resampling and restoration in the above-described algorithm is supplemented with the spectrum analysis for generating a control signal. This signal is used to select, in each sliding window position and depending on the window content, the discrete sinc-interpolation or another interpolation method such as, for instance, nearest neighbor one.

Figure 8.28 illustrates nonadaptive and adaptive sliding window sinc-interpolation on an example of a shift, by an interval equal 17.54 of discretization intervals, of a test signal composed of a sinusoidal wave and rectangular impulses. As one can see from the figure, nonadaptive sinc-interpolated resampling of such a signal results in oscillations at the edges of rectangular impulses. Adaptive resampling implemented in this example switches between the sinc-interpolation and the nearest neighbor interpolation whenever energy of high-frequency components of signal local spectrum exceeds a certain threshold level. As a result, rectangular impulses are resampled with "superresolution." Figure 8.29 illustrates, for comparison, zooming a test signal by means of the nearest neighbor, linear, and bicubic spline interpolations and the above-described adaptive sliding window DCT sinc-interpolation. One can see from these figures that interpolation artifacts seen in other interpolation methods are absent when the adaptive sliding widow interpolation is used.

Nonadaptive and adaptive sliding window sinc-interpolation are also illustrated and compared in Figure 8.30 for rotation of an image that contains gray tone and graphic components. One can clearly see oscillations at sharp boundaries

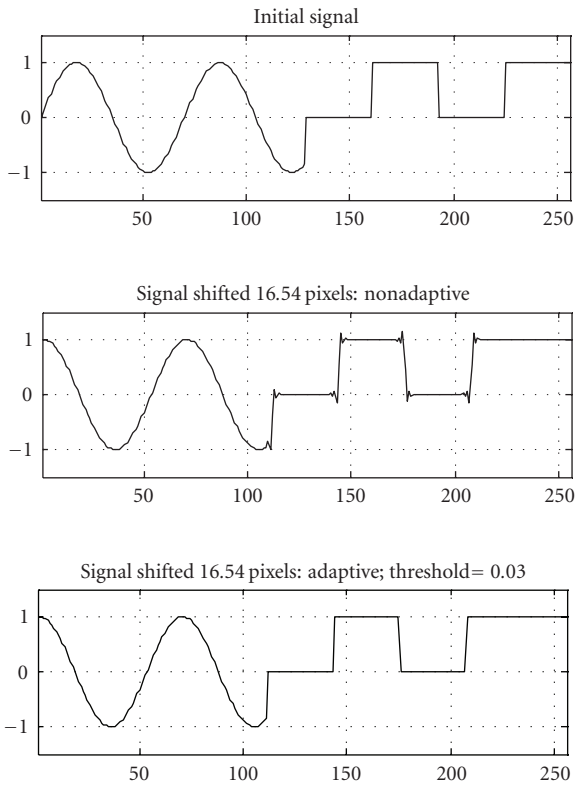


FIGURE 8.28. Fractional shift of a signal (upper plot) by nonadaptive (middle plot) and adaptive (bottom plot) sliding window DCT sinc-interpolations. One can notice disappearance of oscillations at the edges of rectangular impulses when interpolation is adaptive.

of black and white squares in the result of rotation without adaptation. Adaptive sinc-interpolation does not show these artifacts because of switching between the sinc-interpolation and the nearest neighbor interpolation at the sharp boundaries.

Algorithmic implementation of sliding window DFT and DCT discrete sinc-interpolation methods is based on recursive algorithms for computing the DFT and DCT in sliding window described in Appendix E. Consider first computational complexity of sliding window DFT 1D discrete sinc-interpolation. Let N_w be the size of the sliding window. From the symmetry consideration, N_w should be an odd number. As it is shown in Appendix E, (E.3), signal spectrum $\alpha_r^{\{k\}}$, $r = 0, 1, \ldots, N_w - 1$, in kth position of the sliding window is computed from spectrum $\alpha_r^{\{k-1\}}$ in the previous window position as

$$\begin{aligned} \alpha_0^{(k)} &= \alpha_0^{(k-1)} + a_{k-1+(N_w+1)/2} - a_{k-1-(N_w-1)/2}, \\ \alpha_r^{(k)} &= \left[\alpha_r^{(k-1)} + a_{k-1+(N_w+1)/2} - a_{k-1-(N_w-1)/2}\right] \exp\left(-i2\pi\frac{r}{N_w}\right). \end{aligned} \tag{8.66}$$

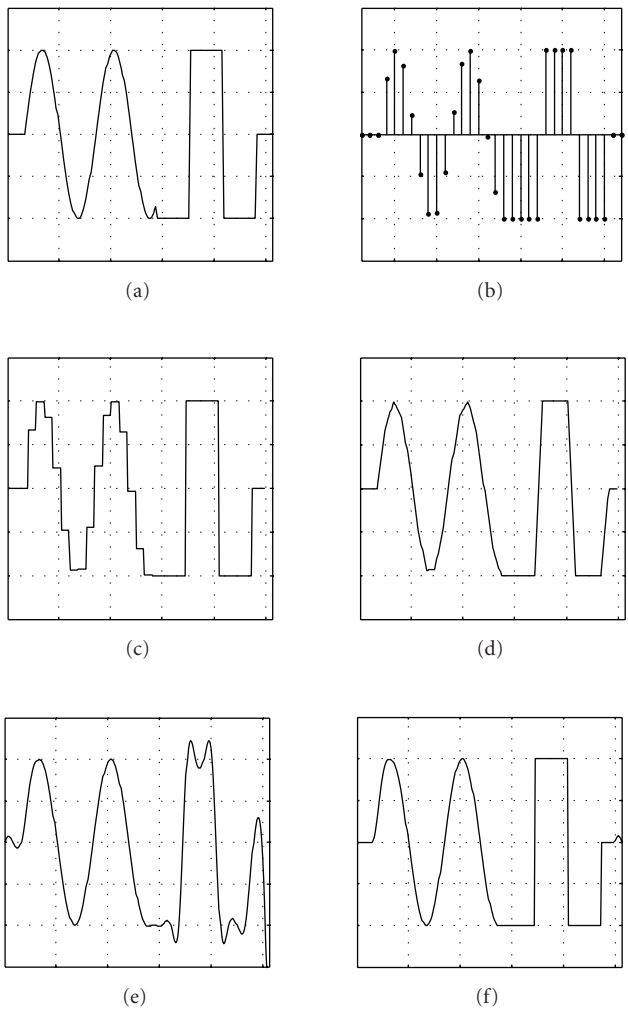
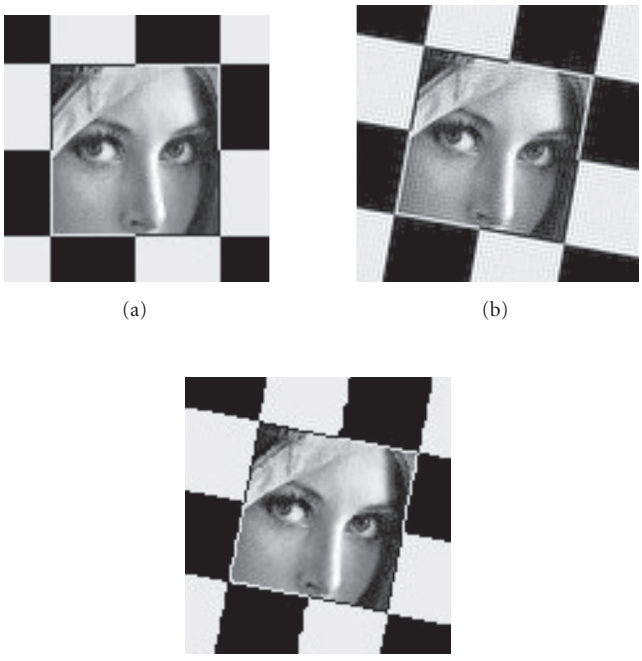


FIGURE 8.29. Signal zooming with nonadaptive and adaptive discrete sinc-interpolation. First row: a model of a continuous signal (a) and its sampled version (b). Second row: initial signal reconstructed by nearest neighbor (c) and linear (d) interpolation. Third row: signal reconstruction using nonadaptive (e) and adaptive (f) discrete sinc-interpolation. One can see that smooth sinusoidal signal and rectangular impulses are reconstructed with much less artifacts when adaptive interpolation is used.

It is convenient to compute signal local DC component $\alpha_0^{\{k\}}$ separately from other spectral coefficients as it is shown in (8.66). As for the other coefficients, only first $(N_w - 1)/2$ have to be computed, as, for real valued signals, the rest $(N_w - 1)/2$ of the coefficients are complex conjugate to them. These computations require $(N_w - 1)/2$ multiplications of complex numbers and $(N_w - 1)/2$ additions plus 2 additions for computing $\alpha_0^{\{k\}}$ which amounts to total $2(N_w - 1)$ of real number multiplications and $3(N_w + 1)/2$ additions.



(a) (b)

(c)

FIGURE 8.30. Rotation of an image (a) with sliding window nonadaptive (b) and adaptive DCT sinc-interpolations (c).

For signal p-shift, these coefficients have to be, according to (8.20), multiplied by $\exp(i2\pi pr/N)$ which yields modified coefficients $\widetilde{\alpha}_r^{(k)}$,

$$\widetilde{\alpha}_r^{(k)} = [\alpha_r^{(k-1)} + a_{k-1+(N_w+1)/2} - a_{k-1-(N_w-1)/2}] \exp\left(- i2\pi \frac{1-p}{N_w} r \right), \qquad (8.67)$$

that are to be used for reconstruction of the window central sample with index $(N_w + 1)/2$ by the inverse DFT as

$$\widetilde{a}_k = \frac{1}{N} \sum_{r-1}^{N_w-1} \widetilde{\alpha}_r^{(k)} \exp\left(- i2\pi \frac{N_w+1}{2N_w} r \right)$$

$$= \frac{1}{N} \left[\sum_{r-1}^{(N_w-1)/2} \widetilde{\alpha}_r^{(k)} \exp\left(- i\pi \frac{N_w+1}{N_w} r \right) + \sum_{r-(N_w+1)/2}^{N_w-1} \widetilde{\alpha}_r^{(k)} \exp\left(- i\pi \frac{N_w+1}{N_w} r \right) \right]$$

$$
\begin{aligned}
&= \frac{1}{N}\left[\sum_{r-1}^{(N_w-1)/2} \widetilde{\alpha}_r^{(k)}(-1)^r \exp\left(-i\pi\frac{r}{N_w}\right)\right.\\
&\qquad \left.+\sum_{r-1}^{(N_w-1)/2} \widetilde{\alpha}_{N_w-1}^{(k)}(-1)^{N-r} \exp\left(-i\pi\frac{N_w-r}{N_w}\right)\right]\\
&= \frac{1}{N}\left[\sum_{r-1}^{(N_w-1)/2} \widetilde{\alpha}_r^{(k)}(-1)^r \exp\left(-i\pi\frac{r}{N_w}\right) + \sum_{r-1}^{(N_w-1)/2} \left(\widetilde{\alpha}_r^{(k)}\right)^*(-1)^r \exp\left(i\pi\frac{r}{N_w}\right)\right]\\
&= \frac{2}{N}\sum_{r-1}^{(N_w-1)/2}(-1)^r \operatorname{Re}\left[\widetilde{\alpha}_r^{(k)} \exp\left(-i\pi\frac{r}{N_w}\right)\right].
\end{aligned}
\tag{8.68}
$$

The reconstruction, therefore, requires additional $2\times(N_w-1)/2=(N_w-1)$ real multiplications and $(N_w-1)/2$ additions plus one addition for adding local dc component plus one multiplication by $1/N$. Table 8.1 provides summary of computational complexity of sliding window DFT signal resampling.

Computational complexity of the sliding window DCT sinc-interpolation can be evaluated using data provided in Appendix E and taking into account that, for sliding window processing, only window central sample is evaluated and signal dc component is not modified. Therefore, reconstruction formula given by (8.29) is transformed into

$$
\begin{aligned}
a_{(N_w-1)/2}^{(p)} &= \sum_{n=0}^{N_w-1} a_n \operatorname{sincd}\left[K;N;\frac{N_w-1}{2}-n-p\right]\\
&= \mathrm{ISDFT}_{1/2,0}\left\{\mathrm{DCT}\left\{a_k\right\}\cdot\left\{\eta_r(p)\right\}\right\}\\
&= \frac{1}{\sqrt{2N}}\left[\alpha_0^{\mathrm{DCT}} + 2\sum_{r=1}^{N_w-1}\alpha_r^{\mathrm{DCT}}\eta_r^{\mathrm{re}}\cos\left(\frac{\pi}{2}r\right) - 2\sum_{r=1}^{N_w-1}\alpha_r^{\mathrm{DCT}}\eta_r^{\mathrm{im}}\sin\left(\frac{\pi}{2}r\right)\right]\\
&= \frac{1}{\sqrt{2N}}\left[\alpha_0^{\mathrm{DCT}} + 2\sum_{r=1}^{(N_w-1)/2}(-1)^r\alpha_{2r}^{\mathrm{DCT}}\eta_{2r}^{\mathrm{re}} - 2\sum_{r=1}^{(N_w-1)/2}(-1)^{r-1}\alpha_{2r-1}^{\mathrm{DCT}}\eta_{2r-1}^{\mathrm{im}}\right].
\end{aligned}
\tag{8.69}
$$

According to this formula, only terms $\{\eta_{2r}^{\mathrm{re}}\}$ and $\{\eta_{2r-1}^{\mathrm{im}}\}$ of spectrum modification coefficients have to be computed, and the reconstruction process requires $2(N_w-1)/2=N_w-1$ multiplications and $2[(N_w-1)/2-1]+1=N_w-1$ additions. Total computational complexity of DCT sliding window signal resampling can be evaluated from data presented in Table 8.2.

Table 8.1. Number of operations per output sample for sliding window DFT signal discrete sinc interpolation resampling in the window of N_w samples.

	Additions	Multiplications	Additional operations when look-up-table is not available
Computation of local spectra	$3(N_w + 1)/2$	$2(N_w - 1)$	$(N_w - 1)$
Spectrum modification	$(N_w - 1)/2$	$(N_w - 1)$	$(N_w - 1)$
Reconstruction	$(N_w + 1)/2$	$(N_w - 1)$	$(N_w - 1)$
Total	$5(N_w + 1)/2$	$2(N_w - 1)$	$3(N_w - 1)$

Table 8.2. Computational complexity per output sample for sliding window DCT signal resampling.

	Multiplications	Additions	Additional operations when look-up-table is not available
Computation of local spectra	$2(N_w - 1)$	$5N_w - 3$	$2(N_w - 1)$
Spectrum modification	$N_w - 1$	—	—
Reconstruction	—	$N_w - 1$	—
Total	$3(N_w - 1)$	$6N_w - 4$	$2(N_w - 1)$

8.6. Conclusion

In this chapter, we described a family of fast transform-based methods and corresponding efficient computational algorithms for discrete sinc-interpolation in digital signal and image processing and showed that they provide, for a finite number of signal samples, virtually error-free interpolation and, therefore, they outperform, in this respect, other known convolutional interpolation methods such as spline ones and alike that implement digital convolution in signal domain rather than in Fourier/DCT transform domain. We demonstrated how methods of discrete sinc-interpolation can be used in different applications and provided numerical estimations and experimental data that evidence that described discrete sinc-inerpolation methods are quite competitive, in terms of their computational complexity, with other methods. In addition, we described sliding window discrete sinc-interpolation methods for signal and image resampling in arbitrary nonuniform sampling grid. Being comparable, in their interpolation accuracy and computational complexity, with other convolutional methods for the same window size, they offer options of adaptive interpolation and simultaneous signal/image space-variant restoration not available with other methods.

Computational efficiency of the interpolation error-free discrete sinc-interpolation algorithms is rooted in the use of fast Fourier and fast DCT transforms. Perhaps, the best concluding remark of this discussion of the discrete sinc-interpolation methods and their applications would be mentioning that fast Fourier

transform algorithm was invented two hundred years ago by Carl Friedrich Gauss exactly for the purpose of facilitating numerical interpolation of sampled data of astronomical observation (see [19, 20]).

Appendices

A. Complex conjugate symmetry for SDFT of real sequences

Let $\{\eta_r\}$ be SDFT$(u, 0)$ of a real sequence $\{h_n = h_n^*\}$:

$$h_n = \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \eta_r \exp\left[- i2\pi \frac{(n+u)r}{N} \right]. \tag{A.1}$$

Then,

$$\begin{aligned} h_n^* &= \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \eta_r^* \exp\left[i2\pi \frac{(n+u)r}{N} \right] = \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \eta_{N-r}^* \exp\left[i2\pi \frac{(n+u)(N-r)}{N} \right] \\ &= \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} \eta_{N-r}^* \exp(i2\pi u) \exp\left[- i2\pi \frac{(n+u)r}{N} \right]. \end{aligned} \tag{A.2}$$

Therefore, for real sequences $\{h_n = h_n^*\}$, the following SDFT$(u, 0)$ complex conjugate symmetry rule holds:

$$\eta_r = \eta_{N-r}^* \exp(i2\pi u). \tag{A.3}$$

In particular, when $u = -(N-1)/2$,

$$\eta_r = \eta_{N-r}^* \exp\left(i\pi(N-1)\right) = \begin{cases} \eta_{N-r}^*, & N\text{-odd number}, \\ -\eta_{N-r}^*, & N\text{-even number}. \end{cases} \tag{A.4}$$

B. Point spread functions of optimal resampling filters

By definition (8.15) and with an account for (8.17), point spread function of the optimal resampling filter $h_n^{(\mathrm{intp})}(\delta\widetilde{x})$ is SDFT$(-(N-1)/2, 0)$ of samples $\{\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}(\delta\widetilde{x})\}$

of its continuous frequency response. From (8.20), for odd number of signal samples N, we have

$$
\begin{aligned}
h_n^{(\mathrm{intp})}(\delta\widetilde{x}) &= \frac{1}{\sqrt{N}}\sum_{r=0}^{N-1}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}(\delta\widetilde{x})\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right] \\
&= \frac{1}{\sqrt{N}}\left\{\sum_{r=0}^{(N-1)/2}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right]\right. \\
&\qquad \left.+\sum_{r=(N+1)/2}^{N-1}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right]\right\} \\
&= \frac{1}{\sqrt{N}}\left\{\sum_{r=0}^{(N-1)/2}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right]\right. \\
&\qquad \left.+\sum_{r=1}^{(N-1)/2}\eta_{N-r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}(N-r)\right]\right\} \\
&= \frac{1}{\sqrt{N}}\left\{\sum_{r=0}^{(N-1)/2}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right]\right. \\
&\qquad \left.+\sum_{r=1}^{(N-1)/2}\eta_{N-r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[i\pi(N-1)\right]\exp\left[i2\pi\frac{n-(N-1)/2}{N}r\right]\right\} \\
&= \frac{1}{\sqrt{N}}\left\{\sum_{r=0}^{(N-1)/2}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right]\right. \\
&\qquad \left.+\sum_{r=1}^{(N-1)/2}\eta_{r,\mathrm{opt}}^{*(\mathrm{intp})}\exp\left[i\pi(N-1)\right]\exp\left[i2\pi\frac{n-(N-1)/2}{N}r\right]\right\} \\
&= \frac{1}{N}\left\{\sum_{r=0}^{(N-1)/2}\exp\left(i2\pi\frac{\delta\widetilde{x}}{N\Delta x}\right)\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right]\right. \\
&\qquad \left.+\sum_{r=1}^{(N-1)/2}\exp\left(-i2\pi\frac{\delta\widetilde{x}}{N\Delta x}\right)\exp\left[i2\pi\frac{n-(N-1)/2}{N}r\right]\right\} \\
&= \frac{1}{N}\left\{\sum_{r=0}^{(N-1)/2}\exp\left[-i2\pi\frac{n-(N-1)/2-\delta\widetilde{x}\Delta x}{N}r\right]\right. \\
&\qquad \left.+\sum_{r=1}^{(N-1)/2}\exp\left[i2\pi\frac{n-(N-1)/2-\delta\widetilde{x}/\Delta x}{N}r\right]\right\}. \qquad (A.1)
\end{aligned}
$$

Denote temporarily

$$\tilde{n} = n - \frac{N-1}{2} - \frac{\delta\tilde{x}}{\Delta x}. \tag{A.2}$$

Then,

$$\begin{aligned} h_n^{(\mathrm{intp})}(\delta\tilde{x}) &= \frac{1}{N}\left\{\frac{\exp\left[-i\pi(\tilde{n}/N)(N+1)\right]-1}{\exp\left[-i2\pi(\tilde{n}/N)\right]-1}\right. \\ &\qquad \left. + \frac{\exp\left[i\pi(\tilde{n}/N)(N+1)\right]-\exp\left[i2\pi(\tilde{n}/N)\right]}{\exp\left[i2\pi(\tilde{n}/N)\right]-1}\right\} \\ &= \frac{1}{N}\left\{\frac{\exp(-i\pi\tilde{n})-\exp\left(i\pi(\tilde{n}/N)\right)}{\exp\left(-i\pi(\tilde{n}/N)\right)-\exp\left(i\pi(\tilde{n}/N)\right)}\right. \\ &\qquad \left. + \frac{\exp(i\pi\tilde{n})-\exp\left(i\pi(\tilde{n}/N)\right)}{\exp\left(i\pi(\tilde{n}/N)\right)-\exp\left(-i\pi(\tilde{n}/N)\right)}\right\} \\ &= \frac{1}{N}\frac{-\exp(-i\pi\tilde{n})+\exp\left(i\pi(\tilde{n}/N)\right)+\exp(i\pi\tilde{n})-\exp\left(i\pi(\tilde{n}/N)\right)}{\exp\left(i\pi(\tilde{n}/N)\right)-\exp\left(-i\pi(\tilde{n}/N)\right)} \\ &= \frac{1}{N}\frac{\sin(\pi\tilde{n})}{\sin\left(\pi(\tilde{n}/N)\right)} = \mathrm{sincd}\left[N, \pi\left(n - \frac{N-1}{2} - \frac{\delta\tilde{x}}{\Delta x}\right)\right]. \end{aligned} \tag{A.3}$$

From (8.21), for even number of signal samples N, we have

$$\begin{aligned} h_n^{(\mathrm{intp})}(\delta\tilde{x}) &= \frac{1}{\sqrt{N}}\sum_{r=0}^{N-1}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}(\delta\tilde{x})\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right] \\ &= \frac{1}{\sqrt{N}}\left\{\sum_{r=0}^{N/2-1}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right]\right. \\ &\qquad \left. + \sum_{r=N/2}^{N-1}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right]\right\} \\ &= \sum_{r=N/2}^{N-1}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right] \\ &= \frac{1}{\sqrt{N}}\left\{\sum_{r=0}^{N/2-1}\eta_{r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}r\right]\right. \\ &\qquad \left. + \sum_{r=1}^{N/2}\eta_{N-r,\mathrm{opt}}^{(\mathrm{intp})}\exp\left[-i2\pi\frac{n-(N-1)/2}{N}(N-r)\right]\right\} \end{aligned}$$

$$= \frac{1}{\sqrt{N}}\left\{ \sum_{r=0}^{N/2-1} \eta_{r,\mathrm{opt}}^{(\mathrm{intp})} \exp\left[-i2\pi \frac{n-(N-1)/2}{N} r \right] \right.$$

$$\left. + \sum_{r=1}^{N/2} \eta_{N-r,\mathrm{opt}}^{(\mathrm{intp})} \exp\left[i\pi(N-1) \right] \exp\left[i2\pi \frac{n-(N-1)/2}{N} r \right] \right\}$$

$$= \frac{1}{\sqrt{N}}\left\{ \sum_{r=0}^{(N-1)/2} \eta_{r,\mathrm{opt}}^{(\mathrm{intp})} \exp\left[-i2\pi \frac{n-(N-1)/2}{N} r \right] \right.$$

$$\left. + \sum_{r=1}^{N/2} \left(-\eta_{r,\mathrm{opt}}^{*(\mathrm{intp})} \right) \exp\left[i\pi(N-1) \right] \exp\left[i2\pi \frac{n-(N-1)/2}{N} r \right] \right\}$$

$$= \frac{1}{N}\left\{ \sum_{r=0}^{N/2-1} \exp\left(i2\pi \frac{\delta\widetilde{x}}{N\Delta x} \right) \exp\left[-i2\pi \frac{n-(N-1)/2}{N} r \right] \right.$$

$$+ \sum_{r=1}^{N/2-1} \exp\left(-i2\pi \frac{\delta\widetilde{x}}{N\Delta x} \right) \exp\left[i2\pi \frac{n-(N-1)/2}{N} r \right]$$

$$\left. + \eta_{N/2,\mathrm{opt}}^{(\mathrm{intp})} \exp\left[i\pi N \frac{n-(N-1)/2}{N} \right] \right\}$$

$$= \frac{1}{N}\left\{ \sum_{r=0}^{N/2-1} \exp\left[-i2\pi \frac{n-(N-1)/2-\delta\widetilde{x}/\Delta x}{N} r \right] \right.$$

$$+ \sum_{r=1}^{N/2-1} \exp\left[i2\pi \frac{n-(N-1)/2-\delta\widetilde{x}/\Delta x}{N} r \right]$$

$$\left. + \eta_{N/2,\mathrm{opt}}^{(\mathrm{intp})} \exp\left[i\pi N \frac{n-(N-1)/2}{N} \right] \right\}$$

$$= \frac{1}{N}\left\{ \sum_{r=0}^{N/2-1} \exp\left(-i2\pi \frac{\widetilde{n}}{N} r \right) + \sum_{r=1}^{(N-1)/2} \exp\left(i2\pi \frac{\widetilde{n}}{N} r \right) \right.$$

$$\left. + \eta_{N/2,\mathrm{opt}}^{(\mathrm{intp})} \exp\left(i\pi \frac{\delta\widetilde{x}}{\Delta x} \right) \exp\left[i\pi N \frac{n-(N-1)/2-\delta\widetilde{x}/\Delta x}{N} \right] \right\}$$

$$= \frac{1}{N}\left\{ \frac{\exp(-i\pi\widetilde{n})-1}{\exp\left(-i2\pi(\widetilde{n}/N)\right)-1} + \frac{\exp(-i\pi\widetilde{n})-\exp\left(i2\pi(\widetilde{n}/N)r\right)}{\exp\left(i2\pi(\widetilde{n}/N)\right)-1} \right.$$

$$\left. + \eta_{N/2,\mathrm{opt}}^{(\mathrm{intp})} \exp\left(i\pi \frac{\delta\widetilde{x}}{\Delta x} \right) \exp(i\pi\widetilde{n}) \right\}$$

$$
\begin{aligned}
&= \frac{1}{N}\left\{ \frac{\exp\left(-i\pi\left((N-1)/N\right)\tilde{n}\right) - \exp\left(i\pi(\tilde{n}/N)\right)}{\exp\left(-i\pi(\tilde{n}/N)\right) - \exp\left(i\pi(\tilde{n}/N)\right)} \right. \\
&\qquad + \frac{\exp\left(i\pi\left((N-1)/N\right)\tilde{n}\right) - \exp\left(i\pi(\tilde{n}/N)r\right)}{\exp\left(i\pi(\tilde{n}/N)\right) - \exp\left(-i\pi(\tilde{n}/N)\right)} \\
&\qquad \left. + \eta^{(\mathrm{intp})}_{N/2,\mathrm{opt}} \exp\left(i\pi\frac{\delta\tilde{x}}{\Delta x}\right)\exp(i\pi\tilde{n}) \right\} \\
&= \frac{1}{N}\left\{ \frac{\exp\left(i\pi\left((N-1)/N\right)\tilde{n}\right) - \exp\left(i\pi(\tilde{n}/N)r\right)}{\exp\left(i\pi(\tilde{n}/N)\right) - \exp\left(-i\pi(\tilde{n}/N)\right)} \right. \\
&\qquad - \frac{\exp\left(-i\pi\left((N-1)/N\right)\tilde{n}\right) + \exp\left(i\pi(\tilde{n}/N)\right)}{\exp\left(i\pi(\tilde{n}/N)\right) - \exp\left(-i\pi(\tilde{n}/N)\right)} \\
&\qquad \left. + \eta^{(\mathrm{intp})}_{N/2,\mathrm{opt}} \exp\left(i\pi\frac{\delta\tilde{x}}{\Delta x}\right)\exp(i\pi\tilde{n}) \right\} \\
&= \frac{1}{N}\left\{ \frac{\exp\left(i\pi\left((N-1)/N\right)\tilde{n}\right) - \exp\left(-i\pi\left((N-1)/N\right)\tilde{n}\right)}{\exp\left(i\pi(\tilde{n}/N)\right) - \exp\left(-i\pi(\tilde{n}/N)\right)} \right. \\
&\qquad \left. + \eta_{N/2} \exp\left(i\pi\frac{\delta\tilde{x}}{\Delta x}\right)\exp(i\pi\tilde{n}) \right\} \\
&= \frac{1}{N}\left\{ \frac{\sin\left(\pi\left((N-1)/N\right)\tilde{n}\right)}{\sin\left(\pi(\tilde{n}/N)\right)} + \eta^{(\mathrm{intp})}_{N/2,\mathrm{opt}} \exp\left(i\pi\frac{\delta\tilde{x}}{\Delta x}\right)\exp(i\pi\tilde{n}) \right\}.
\end{aligned}
\tag{A.4}
$$

Case 0:

$$
\eta^{(\mathrm{intp})}_{N/2,\mathrm{opt}} = 0, \qquad h^{(\mathrm{intp}\,0)}_n(\delta\tilde{x}) = \overline{\mathrm{sincd}}\left\{N; N-1; \pi\left[n - \frac{N-1}{2} - \frac{\delta\tilde{x}}{\Delta x}\right]\right\}. \tag{A.5}
$$

Case 2:

$$
\begin{aligned}
\eta^{(\mathrm{intp})}_{N/2,\mathrm{opt}} &= -i2\sin\left(\pi\frac{\delta\tilde{x}}{\Delta x}\right), \\
h^{(\mathrm{intp}\,0)}_n(\delta\tilde{x}) &= \overline{\mathrm{sincd}}\left\{N; N+1; \pi\left[n - \frac{N+1}{2} - \frac{\delta\tilde{x}}{\Delta x}\right]\right\},
\end{aligned}
\tag{A.6}
$$

where

$$
\overline{\mathrm{sincd}}\{N; M; x\} = \frac{1}{N}\frac{\sin(Mx/N)}{\sin(x/N)}. \tag{A.7}
$$

Proof of Case 2. Find $\eta_{N/2,\mathrm{opt}}^{(\mathrm{intp}\,2)}$ that satisfy the condition

$$\frac{\sin\left(\pi((N-1)N)\widetilde{n}\right)}{\sin\left(\pi(\widetilde{n}/N)\right)}+\eta_{N/2,\mathrm{opt}}^{(\mathrm{intp}\,2)}\exp\left(i\pi\frac{\delta\widetilde{x}}{\Delta x}\right)\exp(i\pi\widetilde{n})=\frac{\sin\left(\pi((N+1)/N)\widetilde{n}\right)}{\sin\left(\pi(\widetilde{n}/N)\right)}. \tag{A.8}$$

From (A.8),

$$\begin{aligned}
\eta_{N/2,\mathrm{opt}}^{(\mathrm{intp})} &= \frac{\sin\left(\pi((N+1)/N)\widetilde{n}\right)-\sin\left(\pi((N-1)/N)\widetilde{n}\right)}{\sin\left(\pi(\widetilde{n}/N)\right)}\exp(-i\pi\widetilde{n})\exp\left(-i\pi\frac{\delta\widetilde{x}}{\Delta x}\right)\\
&= 2\cos(\pi\widetilde{n})\exp(-i\pi\widetilde{n})\exp\left(-i\pi\frac{\delta\widetilde{x}}{\Delta x}\right)\\
&= 2\cos(\pi\widetilde{n})\exp\left[-i\pi\left(n-\frac{N-1}{2}\right)\right]\\
&= 2\cos\left[\pi\left(n-\frac{N-1}{2}-\frac{\delta\widetilde{x}}{\Delta x}\right)\right]\exp\left[-i\pi\left(n-\frac{N-1}{2}\right)\right]\\
&= 2\cos\left[\pi\left(n-\frac{N}{2}+\frac{1}{2}-\frac{\delta\widetilde{x}}{\Delta x}\right)\right]\exp\left[-i\pi\left(n-\frac{N}{2}+\frac{1}{2}\right)\right]\\
&= -i2\cos\left[\pi\left(n-\frac{N}{2}\right)+\pi\left(\frac{1}{2}-\frac{\delta\widetilde{x}}{\Delta x}\right)\right]\cos\left[\pi\left(n-\frac{N}{2}\right)\right]\\
&= -i2\cos\left[\pi\overline{n}+\pi\left(\frac{1}{2}-\frac{\delta\widetilde{x}}{\Delta x}\right)\right]\cos(\pi\overline{n}),
\end{aligned} \tag{A.9}$$

where $\overline{n}=n-N/2$ is an integer number. Therefore,

$$\begin{aligned}
\eta_{N/2,\mathrm{opt}}^{(\mathrm{intp}\,2)} &= -i2\cos\left[\pi\overline{n}+\pi\left(\frac{1}{2}-\frac{\delta\widetilde{x}}{\Delta x}\right)\right](-1)^{\overline{n}}\\
&= -i2\cos(\pi\overline{n})\cos\left[\pi\left(\frac{1}{2}-\frac{\delta\widetilde{x}}{\Delta x}\right)\right](-1)^{\overline{n}}\\
&= -i2\cos\left[\pi\left(\frac{1}{2}-\frac{\delta\widetilde{x}}{\Delta x}\right)\right](-1)^{2\overline{n}}=-i2\sin\left(\pi\frac{\delta\widetilde{x}}{\Delta x}\right).
\end{aligned} \tag{A.10}$$

C. PSF of signal zooming by means of zero-padding of its DCT spectrum

Consider analytical expressions that describe signal zooming by means of zero-padding its DCT spectrum. Let α_r^{DCT} be DCT spectrum of signal $\{a_k\}$,

$$\alpha_r^{\mathrm{DCT}}=\frac{2}{\sqrt{N}}\sum_{k=0}^{N-1}a_k\cos\left[\pi\frac{(k+1/2)r}{N}\right]=\frac{1}{\sqrt{2N}}\sum_{k=0}^{2N-1}\widetilde{a}_k\exp\left[i2\pi\frac{(k+1/2)r}{2N}\right], \tag{C.1}$$

where $k = 0, \ldots, N-1$ and

$$\widetilde{a}_k = \begin{cases} a_k, & k = 0, \ldots, N-1, \\ a_{2N-1-k}, & k = N, \ldots, LN-1. \end{cases} \tag{C.2}$$

Form a zero pad spectrum,

$$\widetilde{\alpha}_r^{\mathrm{DCT},L} = \begin{cases} \alpha_r^{\mathrm{DCT}}, & r = 0, \ldots, N-1, \\ 0, & r = N, \ldots, LN-1. \end{cases} \tag{C.3}$$

Being a spectrum of DCT, this spectrum has an odd-symmetry property

$$\widetilde{\alpha}_r^{\mathrm{DCT},L} = -\widetilde{\alpha}_{2LN-r}^{\mathrm{DCT},L}; \qquad \widetilde{\alpha}_{LN}^{\mathrm{DCT},L} = 0. \tag{C.4}$$

Compute inverse DCT of the zero pad spectrum, using representation of DCT through SDFT$(1/2, 0)$ and its symmetry property (C.4):

$$\begin{aligned}
\widetilde{\widetilde{\alpha}}_k &= \frac{1}{\sqrt{2LN}} \sum_{r=0}^{2LN-1} \widetilde{\alpha}_r^{\mathrm{DCT},L} \exp\left[-i2\pi \frac{(k+1/2)}{2LN} r \right] \\
&= \frac{1}{\sqrt{2LN}} \left\{ \sum_{r=0}^{N-1} \alpha_r^{\mathrm{DCT}} \exp\left[-i2\pi \frac{(k+1/2)}{2LN} r \right] \right. \\
&\qquad \left. + \sum_{r=2LN-N+1}^{2LN-1} \widetilde{\alpha}_r^{\mathrm{DCT}} \exp\left[-i2\pi \frac{(k+1/2)}{2LN} r \right] \right\} \\
&= \frac{1}{\sqrt{2LN}} \left\{ \sum_{r=0}^{N-1} \alpha_r^{\mathrm{DCT}} \exp\left[-i2\pi \frac{(k+1/2)}{2LN} r \right] \right. \\
&\qquad \left. + \sum_{r=1}^{N-1} \widetilde{\alpha}_{2LN-r}^{\mathrm{DCT}} \exp\left[-i2\pi \frac{(k+1/2)}{2LN} (2LN-r) \right] \right\} \\
&= \frac{1}{\sqrt{2LN}} \left\{ \alpha_0^{\mathrm{DCT}} + \sum_{r=1}^{N-1} \alpha_r^{\mathrm{DCT}} \exp\left[-i2\pi \frac{(k+1/2)}{2LN} r \right] \right. \\
&\qquad \left. - \sum_{r=1}^{N-1} \alpha_r^{\mathrm{DCT}} \exp\left[-i2\pi \frac{(k+1/2)}{2LN} (2LN-r) \right] \right\} \\
&= \frac{1}{\sqrt{2LN}} \left\{ \alpha_0^{\mathrm{DCT}} + \sum_{r=0}^{N-1} \alpha_r^{\mathrm{DCT}} \exp\left[-i2\pi \frac{(k+1/2)}{2LN} r \right] \right. \\
&\qquad \left. + \sum_{r=0}^{N-1} \alpha_r^{\mathrm{DCT}} \exp\left[i2\pi \frac{(k+1/2)}{2LN} r \right] \right\}.
\end{aligned} \tag{C.5}$$

Replace in this formula α_r^{DCT} with its expressions through SDFT(1/2, 0) in the right part of (C.1) and obtain

$$\begin{aligned}\tilde{\tilde{a}}_k=&\frac{1}{\sqrt{2LN}}\left\{\alpha_0^{\mathrm{DCT}}+\sum_{r=1}^{N-1}\frac{1}{\sqrt{2N}}\sum_{n=0}^{2N-1}\tilde{a}_n\exp\left[i2\pi\frac{(n+1/2)r}{2N}\right]\times\exp\left[-i2\pi\frac{(k+1/2)}{2LN}r\right]\right.\\&\left.+\sum_{r=1}^{N-1}\frac{1}{\sqrt{2N}}\sum_{n=0}^{2N-1}\tilde{a}_n\exp\left[i2\pi\frac{(n+1/2)r}{2N}\right]\exp\left[i2\pi\frac{(k+1/2)}{2LN}r\right]\right\}\\=&\frac{\alpha_r^{\mathrm{DCT}}}{\sqrt{2LN}}+\frac{1}{\sqrt{2LN}}\sum_{r=1}^{N-1}\frac{1}{\sqrt{2N}}\sum_{n=0}^{2N-1}\tilde{a}_n\left\{\exp\left[\frac{i\pi(n+1/2-(k+1/2)/L)r}{N}\right]\right.\\&\left.+\exp\left[i\pi\frac{(n+1/2+(k+1/2)/L)r}{N}\right]\right\}.\end{aligned}\tag{C.6}$$

Denote, for brevity, $\tilde{n}=n+1/2$ and $\tilde{k}=k+1/2$. Then, obtain

$$\begin{aligned}\tilde{\tilde{a}}_k=&\frac{\alpha_r^{\mathrm{DCT}}}{\sqrt{2LN}}+\frac{1}{\sqrt{2LN}}\sum_{r=1}^{N-1}\frac{1}{\sqrt{2N}}\\&\times\sum_{n=0}^{2N-1}\tilde{a}_n\left\{\exp\left[\frac{i\pi(\tilde{n}-\tilde{k}/L)r}{N}\right]+\exp\left[\frac{i\pi(\tilde{n}+\tilde{k}/L)r}{N}\right]\right\}\\=&\frac{\alpha_r^{\mathrm{DCT}}}{\sqrt{2LN}}+\frac{1}{\sqrt{2LN}}\\&\times\sum_{r=1}^{N-1}\frac{1}{\sqrt{2N}}\left[\sum_{n=0}^{N-1}\tilde{a}_n\left\{\exp\left[\frac{i\pi(\tilde{n}-\tilde{k}/L)r}{N}\right]+\exp\left[\frac{i\pi(\tilde{n}+\tilde{k}/L)r}{N}\right]\right\}\right.\\&\left.+\sum_{n=N}^{2N-1}\tilde{a}_n\left\{\exp\left[\frac{i\pi(\tilde{n}-\tilde{k}/L)r}{N}\right]+\exp\left[\frac{i\pi(\tilde{n}+\tilde{k}/L)r}{N}\right]\right\}\right]\\=&\frac{\alpha_r^{\mathrm{DCT}}}{\sqrt{2LN}}+\frac{1}{\sqrt{2LN}}\\&\times\sum_{r=1}^{N-1}\frac{1}{\sqrt{2N}}\left[\sum_{n=0}^{N-1}\tilde{a}_n\left\{\exp\left[\frac{i\pi(\tilde{n}-\tilde{k}/L)r}{N}\right]+\exp\left[\frac{i\pi(\tilde{n}+\tilde{k}/L)r}{N}\right]\right\}\right.\\&+\sum_{n=0}^{N-1}\tilde{a}_{2N-1-n}\left\{\exp\left[\frac{i\pi(2N-\tilde{n}-\tilde{k}/L)r}{N}\right]\right.\\&\left.\left.\left.+\exp\left[\frac{i\pi(2N-\tilde{n}+\tilde{k}/L)r}{N}\right]\right\}\right]\end{aligned}$$

$$= \frac{\alpha_r^{\mathrm{DCT}}}{\sqrt{2LN}} + \frac{1}{\sqrt{2LN}}$$

$$\times \sum_{r=1}^{N-1} \frac{1}{\sqrt{2N}} \left[\sum_{n=0}^{N-1} a_n \left\{ \exp\left[\frac{i\pi(\tilde{n} - \tilde{k}/L)r}{N} \right] + \exp\left[\frac{i\pi(\tilde{n} + \tilde{k}/L)r}{N} \right] \right\} \right.$$

$$+ \sum_{n=0}^{N-1} \tilde{a}_{2N-1-n} \left\{ \exp\left[-\frac{i\pi(\tilde{n} + \tilde{k}/L)r}{N} \right] \right.$$

$$\left. \left. + \exp\left[-\frac{i\pi(\tilde{n} - \tilde{k}/L)r}{N} \right] \right\} \right]$$

$$= \frac{\alpha_r^{\mathrm{DCT}}}{\sqrt{2LN}} + \frac{1}{2N\sqrt{L}} \sum_{n=0}^{N-1} a_n \left\{ \sum_{r=1}^{N-1} \exp\left[\frac{i\pi(\tilde{n} - \tilde{k}/L)r}{N} \right] \right.$$

$$+ \sum_{r=1}^{N-1} \exp\left[\frac{i\pi(\tilde{n} + \tilde{k}/L)r}{N} \right]$$

$$\times \sum_{r=1}^{N-1} \exp\left[-\frac{i\pi(\tilde{n} + \tilde{k}/L)r}{N} \right]$$

$$\left. + \sum_{r=1}^{N-1} \exp\left[-\frac{i\pi(\tilde{n} - \tilde{k}/L)r}{N} \right] \right\}$$

$$= \sum_{r=1}^{N-1} \exp\left[-\frac{i\pi(\tilde{n} + \tilde{k}/L)r}{N} \right] + \sum_{r=1}^{N-1} \exp\left[-\frac{i\pi(\tilde{n} - \tilde{k}/L)r}{N} \right]$$

$$= \frac{\alpha_r^{\mathrm{DCT}}}{\sqrt{2LN}} + \frac{1}{2N\sqrt{L}}$$

$$\times \sum_{n=0}^{N-1} a_n \left\{ \frac{\exp\left[i\pi(\tilde{n}L - \tilde{k})/L \right] - \exp\left[i\pi(\tilde{n}L - \tilde{k})/LN \right]}{\exp\left[i\pi(\tilde{n}L - \tilde{k})/LN \right] - 1} \right.$$

$$+ \frac{\exp\left[(\tilde{n}L + \tilde{k})/L \right] - \exp\left[i\pi(\tilde{n}L + \tilde{k})/LN \right]}{\exp\left[\pi(\tilde{n}L + \tilde{k})/LN \right] - 1}$$

$$+ \frac{\exp\left[-\pi(\tilde{n}L + \tilde{k})/L \right] - \exp\left[-\pi(\tilde{n}L + \tilde{k})/LN \right]}{\exp\left[-\pi(\tilde{n}L + \tilde{k})/LN \right] - 1}$$

$$\left. + \frac{\exp\left[-i\pi(\tilde{n}L - \tilde{k})/L \right] - \exp\left[-i\pi(\tilde{n}L - \tilde{k})/LN \right]}{\exp\left[-i\pi(\tilde{n}L - \tilde{k})/LN \right] - 1} \right\}$$

$$= \frac{\alpha_0^{\mathrm{DCT}}}{\sqrt{2LN}} + \frac{1}{2N\sqrt{L}} \sum_{n=0}^{N-1} a_n \left\{ \frac{\sin\left[\pi((N+1)/2NL)(\tilde{n}L - \tilde{k})\right]}{\sin\left[\pi(\tilde{n}L - \tilde{k})/2NL\right]} \exp\left[\frac{i\pi(\tilde{n}L - \tilde{k})}{2L}\right] \right.$$
$$+ \frac{\sin\left[\pi((N+1)/2NL)(\tilde{n}L + \tilde{k})\right]}{\sin\left[\pi(\tilde{n}L + \tilde{k})/2NL\right]} \exp\left[\frac{i\pi(\tilde{n}L + \tilde{k})}{2L}\right]$$
$$+ \frac{\sin\left[\pi((N+1)/2NL)(\tilde{n}L + \tilde{k})\right]}{\sin\left[\pi(\tilde{n}L + \tilde{k})/2NL\right]} \exp\left[-\frac{i\pi(\tilde{n}L + \tilde{k})}{2L}\right]$$
$$\left. + \frac{\sin\left[\pi((N+1)/2NL)(\tilde{n}L - \tilde{k})\right]}{\sin\left[\pi(\tilde{n}L - \tilde{k})/2NL\right]} \exp\left[-\frac{i\pi(\tilde{n}L - \tilde{k})}{2L}\right] \right\}. \tag{C.7}$$

From this, we finally obtain

$$\tilde{\tilde{a}}_k = \frac{\alpha_0^{\mathrm{DCT}}}{\sqrt{2LN}} + \frac{1}{N\sqrt{L}} \sum_{n=0}^{N-1} a_n \left\{ \frac{\sin\left[\pi((N+1)/2NL)(\tilde{n}L - \tilde{k})\right]}{\sin\left[\pi(\tilde{n}L - \tilde{k})/2NL\right]} \cos\left[\frac{\pi(\tilde{n}L - \tilde{k})}{2L}\right] \right.$$
$$\left. + \frac{\sin\left[\pi((N+1)/2NL)(\tilde{n}L + \tilde{k})\right]}{\sin\left[\pi(\tilde{n}L + \tilde{k})/2NL\right]} \cos\left[\frac{\pi(\tilde{n}L + \tilde{k})}{2L}\right] \right\}. \tag{C.8}$$

By analogy with DFT zero-padding, one can improve convergence of DCT zero-padding PSF by halving DCT spectral coefficients that correspond to the highest frequency. For signals of N samples, these are coefficients with indices $N-2$ and $N-1$. To find an analytical expression for this case, we first compute, using (C.7) and (C.8), L-zoomed signal for the case when those coefficients are zeroed:

$$\tilde{\tilde{a}}_k = \frac{\alpha_r^{\mathrm{DCT}}}{\sqrt{2LN}} + \frac{1}{2N\sqrt{L}}$$
$$\times \sum_{n=0}^{N-1} a_n \left\{ \sum_{r=1}^{N-3} \exp\left[\frac{i\pi(\tilde{n} - \tilde{k}/L)r}{N}\right] + \sum_{r=1}^{N-3} \exp\left[\frac{i\pi(\tilde{n} + \tilde{k}/L)r}{N}\right] \right.$$
$$\left. + \sum_{r=1}^{N-3} \exp\left[-\frac{i\pi(\tilde{n} + \tilde{k}/L)r}{N}\right] + \sum_{r=1}^{N-3} \exp\left[-\frac{i\pi(\tilde{n} - \tilde{k}/L)r}{N}\right] \right\}$$
$$= \frac{\alpha_0^{\mathrm{DCT}}}{\sqrt{2LN}} + \frac{1}{N\sqrt{L}}$$
$$\times \sum_{n=0}^{N-1} a_n \left\{ \frac{\sin\left[\pi((N-1)/2N)(\tilde{n} - \tilde{k}/L)\right]}{\sin\left[\pi(\tilde{n} - \tilde{k}/L)/2N\right]} \cos\left[\frac{\pi(N-2)/N(\tilde{n} - \tilde{k}/L)}{2}\right] \right.$$
$$\left. + \frac{\sin\left[\pi((N-1)/2N)(\tilde{n} + \tilde{k}/L)\right]}{\sin\left[\pi(\tilde{n} + \tilde{k}/L)/2N\right]} \cos\left[\frac{\pi(N-2)/N(\tilde{n} + \tilde{k}/L)}{2}\right] \right\}. \tag{C.9}$$

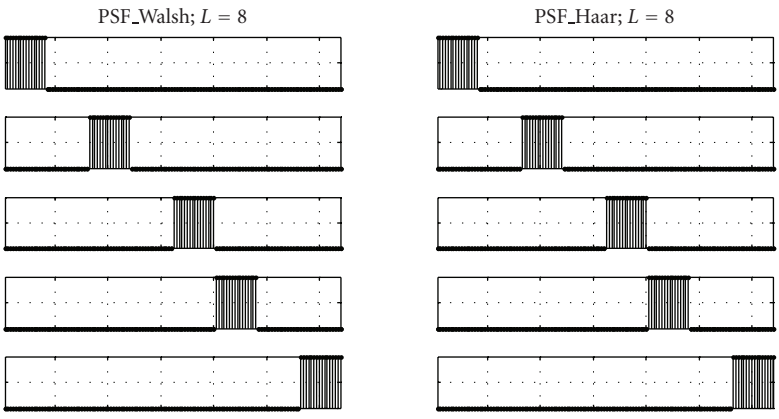


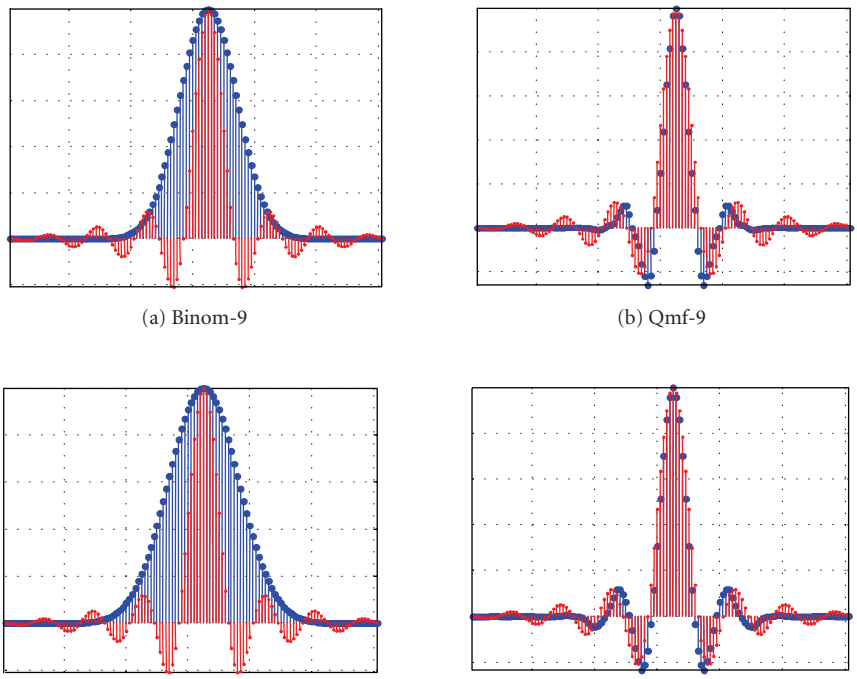
FIGURE D.1. Point spread functions of signal 8x-zooming by means of zero-padding its Walsh spectrum (a) and Haar spectrum (b) for five different positions of the signal impulse.

Then L-zoomed signal obtained by zero-padding signal DCT spectrum and halving its highest frequency components is defined by the equation

$$\begin{aligned}\widetilde{\widetilde{a}}_k = \frac{\alpha_0^{\mathrm{DCT}}}{\sqrt{2LN}} + \frac{1}{N\sqrt{L}} \sum_{n=0}^{N-1} a_n \left\{ \frac{\sin\left[\pi\left((N+1)/2NL\right)(\tilde{n}L-\widetilde{k})\right]}{\sin\left[\pi(\tilde{n}L-\widetilde{k})/2NL\right]} \cos\left[\frac{\pi(\tilde{n}L-\widetilde{k})}{2L}\right] \right.\\ + \frac{\sin\left[\pi\left((N+1)/2NL\right)(\tilde{n}L+\widetilde{k})\right]}{\sin\left[\pi(\tilde{n}L+\widetilde{k})/2NL\right]} \cos\left[\frac{\pi(\tilde{n}L+\widetilde{k})}{2L}\right]\\ + \frac{\sin\left[\pi\left((N-1)/2NL\right)(\tilde{n}L-\widetilde{k})\right]}{\sin\left[\pi(\tilde{n}L-\widetilde{k})/2NL\right]} \cos\left[\pi\frac{N-2}{2NL}(\tilde{n}L-\widetilde{k})\right]\\ \left. + \frac{\sin\left[\pi\left((N-1)/2NL\right)(\tilde{n}L+\widetilde{k})\right]}{\sin\left[\pi(\tilde{n}L+\widetilde{k})/2NL\right]} \cos\left[\pi\frac{N-2}{2NL}(\tilde{n}L+\widetilde{k})\right] \right\}. \end{aligned} \tag{C.10}$$

D. Signal zooming by means of zero-padding their Walsh and wavelet transforms

In this appendix, we consider point spread functions of signal zooming by means of zero-padding of their spectra in Walsh, Haar, and wavelet transform domain. Obviously, point spread functions of zero-padding in transform domain depend



(a) Binom-9 (b) Qmf-9

(c) Binom-13 (d) Qmf-13

FIGURE D.2. Point spread functions of signal zooming by means of zero-padding its wavelet spectra (bold circles) compared with those for DFT zero-padding (dots).

on transform basis functions. For transforms with binary or ternary basis functions, such as Walsh transform and Haar transform, one cannot expect anything but a rectangular point spread function which describes the nearest neighbor interpolation. Experiments confirm this guess (see Figure D.1). Curiously enough, zero-padding Walsh and Haar spectra turn out to be "quasi"-shift invariant interpolation processes, which is of course not the case for general signal filtering by modification of their Walsh and Haar spectra.

Wavelet transforms are good practical candidates for image zooming by zero-padding its transform when zooming is required for images presented in a compressed form such as in JPEG2000 image compression standard. Point spread functions of such a zooming are determined by the type of wavelets used. Figure D.2 illustrates point spread functions for 4 types of wavelets: Binom-9, Binom-13, Qmf-9, and Qmf-13, and compares them with discrete sinc-function $\mathrm{sincd}(\pm 1, N, x)$.

E. Sliding window DFT and DCT/DcST recursive algorithms

Let us consider DFT of two adjacent $(k-1)$th and kth fragments $\{a_{k-1-\mathrm{fix}((N_w-1)/2)}, \ldots, a_{k-1+\mathrm{fix}(N_w/2)}\}$ and $\{a_{k-\mathrm{fix}((N_w-1)/2)}, \ldots, a_{k+\mathrm{fix}(N_w/2)}\}$ of a signal $\{a_k\}$ in sliding

window of N_w samples

$$\begin{aligned}\alpha_r^{(k-1)} &= \sum_{n=0}^{N_w-1} a_{k-1-\mathrm{fix}((N_w-1)/2)+n} \exp\left(i2\pi\frac{nr}{N_w}\right),\\ \alpha_r^{(k)} &= \sum_{n=0}^{N_w-1} a_{k-\mathrm{fix}((N_w-1)/2)+n} \exp\left(i2\pi\frac{nr}{N_w}\right),\end{aligned} \tag{E.1}$$

where $\mathrm{fix}(x)$ is integer part of x and normalization factor $1/\sqrt{N}$ is neglected. Spectrum $\{\alpha_r^{(k)}\}$ in kth position of the sliding window can be computed from spectrum $\{\alpha_r^{(k-1)}\}$ in $(k-1)$th window position in the following way:

$$\begin{aligned}\alpha_r^{(k)} &= \sum_{n=0}^{N_w-1} a_{k-\mathrm{fix}((N_w-1)/2)+n} \exp\left(i2\pi\frac{nr}{N_w}\right)\\ &= \sum_{n=1}^{N_w} a_{k-1-\mathrm{fix}((N_w-1)/2)+n} \exp\left(i2\pi\frac{(n-1)r}{N_w}\right)\\ &= \left[\sum_{n=0}^{N_w-1} a_{k-1-\mathrm{fix}((N_w-1)/2)+n} \exp\left(i2\pi\frac{nr}{N_w}\right)\right.\\ &\qquad \left. + \left(a_{k-1-\mathrm{fix}((N_w-1)/2)+N_w} - a_{k-1-\mathrm{fix}((N_w-1)/2)}\right)\right] \exp\left(-i2\pi\frac{r}{N_w}\right)\\ &= \left(\alpha_r^{(k-1)} + a_{k-1-\mathrm{fix}((N_w-1)/2)+N_w} - a_{k-1-\mathrm{fix}((N_w-1)/2)}\right) \exp\left(-i2\pi\frac{r}{N_w}\right).\end{aligned} \tag{E.2}$$

Equation (E.2) describes the recursive algorithm for computing DFT signal spectrum in sliding window of N_w samples. For signals whose samples are real numbers, only first $(N_w+1)/2$ coefficients have to be computed. The rest $(N_w-1)/2$ can be found from the symmetry property $\alpha_r = \alpha_{N-r}^*$ of DFT of real numbers. Therefore, $(N_w-1)/2$ multiplications of complex numbers and $(N_w+1)/2$ additions plus 2 additions are required for computing $\{\alpha_r^{\{k\}}\}$ which amounts to total $3(N_w+1)/2$ additions and $2(N_w-1)$ of real number multiplications provided $\{\sin(2\pi(r/N_w))\}$ and $\{\cos(2\pi(r/N_w))\}$ coefficients are taken from a look-up table, which practically always can be done. Otherwise, additional (N_w-1) operations for computing sine-cosine coefficients are required.

When sliding window DFT algorithms are used for local adaptive filtering and resampling as described in Chapter 5, inverse DFT is required for computing window central pixel. Using the above-mentioned symmetry property of DFT of

real numbers, one can show that required inverse can be computed as

$$\begin{aligned}
\widetilde{a}_k &= \sum_{r-1}^{N_w-1} \widetilde{\alpha}_r^{(k)} \exp\left(-i2\pi\frac{N_w+1}{2N_w}r\right) \\
&= \left[\sum_{r-1}^{(N_w-1)/2} \widetilde{\alpha}_r^{(k)} \exp\left(-i\pi\frac{N_w+1}{N_w}r\right) + \sum_{r-(N_w+1)/2}^{N_w-1} \widetilde{\alpha}_r^{(k)} \exp\left(-i\pi\frac{N_w+1}{N_w}r\right)\right] \\
&= \left[\sum_{r-1}^{(N_w-1)/2} \widetilde{\alpha}_r^{(k)}(-1)^r \exp\left(-i\pi\frac{r}{N_w}\right) + \sum_{r-1}^{(N_w-1)/2} \widetilde{\alpha}_{N_w-1}^{(k)}(-1)^{N-r} \exp\left(-i\pi\frac{N_w-r}{N_w}\right)\right] \\
&= \left[\sum_{r-1}^{(N_w-1)/2} \widetilde{\alpha}_r^{(k)}(-1)^r \exp\left(-i\pi\frac{r}{N_w}\right) + \sum_{r-1}^{(N_w-1)/2} (\widetilde{\alpha}_r^{(k)})^*(-1)^r \exp\left(i\pi\frac{r}{N_w}\right)\right] \\
&= 2\sum_{r-1}^{(N_w-1)/2} (-1)^r \operatorname{Re}\left[\widetilde{\alpha}_r^{(k)} \exp\left(-i\pi\frac{r}{N_w}\right)\right].
\end{aligned} \tag{E.3a}$$

The reconstruction, therefore, requires additional $(N_w + 1)/2$ additions and $2 \times (N_w - 1)/2 = (N_w - 1)$ real multiplications provided $\{\sin(\pi(r/N_w))\}$ and $\{\cos(\pi(r/N_w))\}$ coefficients are taken from a look-up table.

In order to evaluate total computational complexity of the filtering in DFT domain in sliding window such as described in Chapter 5, additional $(N_w - 1)/2$ operations on complex numbers are required for modification of the window spectral coefficients according to the filtering. For signal p-shift resampling, discussed in Section 8.2.3, the spectrum modification is multiplying spectral coefficients by $\exp(-i2\pi pr/N_w)$ that can be included in the inverse transform

$$\frac{2}{N}\sum_{r-1}^{(N_w-1)/2} (-1)^r \operatorname{Re}\left[\widetilde{\alpha}_r^{(k)} \exp\left(-i\pi\frac{1+2p}{N_w}r\right)\right]. \tag{E.3b}$$

However, in this case, keeping coefficients $\{\sin(\pi((1+2p)/N_w)r)\}$ and $\{\cos(\pi((1+2p)/N_w)r)\}$ is not always possible if the required shift is different for different window position. The results of evaluation of the computational complexity are summarized in Table E.1 separately for general local adaptive filtering and or signal p-shift.

The DCT and DcST in a running window can be computed in a similar way. For adjacent $(k-1)$th and kth signal fragments, their DCT and DcST spectra $\{\alpha_r^{(k-1)}\}$ and $\{\alpha_r^{(k)}\}$ are, correspondingly, real and imaginary parts of auxiliary spectra

$$\begin{aligned}
\widetilde{\alpha}_r^{(k-1)} &= \frac{1}{\sqrt{N_w}}\sum_{n=0}^{N-1} a_{k-1-\operatorname{fix}((N_w-1)/2)+n} \exp\left[i\pi\frac{(n+1/2)r}{N_w}\right], \\
\widetilde{\alpha}_r^{(k)} &= \frac{1}{\sqrt{N_w}}\sum_{n=0}^{N-1} a_{k-\operatorname{fix}((N_w-1)/2)+n} \exp\left[i\pi\frac{(n+1/2)r}{N_w}\right].
\end{aligned} \tag{E.4}$$

Table E.1. Number of operations per output sample for sliding window DFT filtering in the window of N_w samples.

		Additions	Multiplications	Additional operations when look-up-table is not available
Computation of local spectra		$3(N_w+1)/2$	$2(N_w-1)$	(N_w-1)
Reconstruction		$(N_w+1)/2$	(N_w-1)	(N_w-1)
Spectrum modification	Filtering	$(N_w-1)/2$ complex number operations		
	p-shift	$(N_w-1)/2$	(N_w-1)	(N_w-1)
Total	Filtering	Additional $(N_w-1)/2$ complex number operations		
	p-shift	$5(N_w+1)/2$	$2(N_w-1)$	(N_w-1)

Spectrum $\{\widetilde{\alpha}_r^{(k)}\}$ can be represented through spectrum $\{\widetilde{\alpha}_r^{(k-1)}\}$ as follows:

$$\widetilde{\alpha}_r^{(k)} = \widetilde{\alpha}_r^{(k-1)} \exp\left(-i\pi\frac{r}{N_w}\right) + \left[(-1)^r a_{k-1-\mathrm{fix}((N_w-1)/2)+N_w} - a_{k-1-\mathrm{fix}((N_w-1)/2)}\right] \exp\left(-i\pi\frac{r}{2N_w}\right). \tag{E.5}$$

Recursive computations according to (E.5) require $4(N_w-1)$ multiplication and $6N_w-3$ addition operations for both DCT and DcST transform in sliding window of N samples.

Extension of these algorithms to the recursive computation of 2D DFT, DCT, and DcST is straightforward owing to the separability of the transforms.

As one can see, in the recursive algorithm for computing DCT and DcST in sliding window described by (E.5), DCT and DcST are inseparable:

$$\alpha_r^{(k)\,\mathrm{DCT}} = \mathrm{Real}\{\widetilde{\alpha}_r^{(k)}\} = \alpha_r^{(k-1)\,\mathrm{DCT}} C_1 + \alpha_r^{(k-1)\,\mathrm{DST}} S_1 + \Delta a_k C_2, \tag{E.6}$$

$$\alpha_r^{(k)\,\mathrm{DcST}} = \mathrm{Imag}\{\widetilde{\alpha}_r^{(k)}\} = \alpha_r^{(k-1)\,\mathrm{DcST}} C_1 - \alpha_r^{(k-1)\,\mathrm{DCT}} S_1 - \Delta a_k S_2, \tag{E.7}$$

where

$$C_1 = \cos\left(\pi\frac{r}{N_w}\right), \qquad C_2 = \cos\left(\pi\frac{r}{2N_w}\right),$$
$$S_1 = \sin\left(\pi\frac{r}{N_w}\right), \qquad S_2 = \sin\left(\pi\frac{r}{2N_w}\right), \tag{E.8}$$
$$\Delta a_r^{(k)} = (-1)^r a_{k-1-\mathrm{fix}((N_w-1)/2)+N_w} - a_{k-1-\mathrm{fix}((N_w-1)/2)}.$$

Many applications require computing only the DCT in sliding window. In these cases, computations of both DCT and DcST according to (E.5) are redundant. This redundancy associated with the need to compute the DcST for recursive

computing the DCT can be eliminated in an algorithm that uses DCT spectrum found on two previous window positions (see [18]). To derive this algorithm, compute DFTs A_s^{DCT} and A_s^{DcST} of $\{\alpha_r^{(k)\,\mathrm{DCT}}\}$ and $\{\alpha_r^{(k)\,\mathrm{DcST}}\}$, respectively, over index k assuming that the number of signal samples is N_k. According to the shift theorem of the DFT, obtain, for A_s^{DCT} and A_s^{DcST} from (E.6) and (E.7),

$$A_s^{\mathrm{DCT}} = A_s^{\mathrm{DCT}} C_1 Z + A_s^{\mathrm{DcST}} S_1 Z + \Delta A_s C_2, \tag{E.9}$$

$$A_s^{\mathrm{DcST}} = A_s^{\mathrm{DcST}} C_1 Z - A_s^{\mathrm{DCT}} S_1 Z - \Delta A_s S_2, \tag{E.10}$$

where s is index in the DFT domain, ΔA_s is DFT of $\Delta a_r^{(k)}$, and $Z = \exp(i2\pi(s/N_k))$—the spectral shift factor that corresponds to signal shift one sample back. Express, using (E.7), A_s^{DcST} through A_s^{DCT} and substitute it into (E.9) and (E.10). Then, obtain

$$A_s^{\mathrm{DCT}}[1 - C_1 Z]^2 = -A_s^{\mathrm{DCT}} S_1^2 Z^2 + \Delta A_s S_2 S_1 Z + \Delta A_s C_2 (1 - C_1 Z), \tag{E.11}$$

from which it follows that

$$A_s^{\mathrm{DCT}} = 2C_1 A_s^{\mathrm{DCT}} Z - (C_1^2 + S_1^2) A_s^{\mathrm{DCT}} Z^2 - (C_2 C_1 - S_2 S_1)\Delta A_s Z + C_2 \Delta A_s, \tag{E.12}$$

or, as $C_1^2 + S_1^2 = 1$ and $C_2 C_1 - S_2 S_1 = C_2$,

$$A_s^{\mathrm{DCT}} = 2C_1 A_s^{\mathrm{DCT}} Z - A_s^{\mathrm{DCT}} Z^2 + C_2(\Delta A_s - \Delta A_s Z). \tag{E.13}$$

Finally, taking inverse DFT of both parts of (E.13) and using the fact that spectral multiplicands Z and Z^2 correspond to signal shifts one and two samples back, respectively, obtain for the sliding window DCT that

$$\begin{aligned} \alpha_r^{(k)\,\mathrm{DCT}} = {} & 2\alpha_r^{(k-1)\,\mathrm{DCT}} \cos\left(\pi \frac{r}{N_w}\right) - \alpha_r^{(k-2)\,\mathrm{DCT}} \\ & + (\Delta a_r^{(k)} - \Delta a_r^{(k-1)}) \cos\left(\pi \frac{r}{2N_w}\right). \end{aligned} \tag{E.14}$$

Similar recursive relationship can be derived for the sliding window DcST:

$$\begin{aligned} \alpha_r^{(k)\,\mathrm{DcST}} = {} & 2\alpha_r^{(k-1)\,\mathrm{DcST}} \cos\left(\pi \frac{r}{N_w}\right) - \alpha_r^{(k-2)\,\mathrm{DcST}} \\ & - (\Delta a_r^{(k)} + \Delta a_r^{(k-1)}) \sin\left(\pi \frac{r}{2N_w}\right). \end{aligned} \tag{E.15}$$

Computations according to (E.14) and (E.15) require only $2(N_w - 1)$ multiplication and $5(N_w - 1) + 2$ addition operations per output signal sample provided local dc ($\alpha_0^{(k)\,\mathrm{DCT}}$, $\alpha_0^{(k)\,\mathrm{DcST}}$) coefficients are computed separately with recursive algorithm that requires 2 additions per sample and sine and cosine coefficients are

Table E.2. Computational complexity per output sample for sliding window DCT/DcST processing.

			Multiplications	Additions	Additional operations when look-up-table is not available
Computation of local spectra	DCT&DcST		$4(N_w - 1)$	$6N_w - 3$	$2(N_w - 1)$
	DCT	Filtering	$N_w - 1$	$5(N_w - 1)/2 + 2$	$N_w - 1$
		p-shift	$2(N_w - 1)$	$5N_w - 3$	$2(N_w - 1)$
Spectrum modification	Filtering		$N_w - 1$	—	—
	p-shift		$N_w - 1$	—	—
Reconstruction	Filtering		—	$(N_w - 1)/2 + 1$	
	p-shift		—	$N_w - 1$	—
Total	Filtering		—	—	—
	p-shift		$3(N_w - 1)$	$6N_w - 4$	—

taken from a look-up-table. Note that, when local DCT coefficients are used for local adaptive filtering, only coefficients with even indices are used. Therefore, in this case only $(N_w - 1)$ multiplications and $5(N_w - 1)/2 + 2$ additions are required. Table E.2 summarizes above estimates of the computational complexity of DCT/DcST domain processing in sliding window.

Bibliography

[1] A. Gotchev, "Spline and Wavelet Based Techniques for Signal and Image Processing," this volume, Chapter 8.

[2] Ph. Thévenaz, EPFL/STI/IOA/LIB, Bldg. BM-Ecublens 4.137, Station 17 , CH-1015 Lausanne VD, Switzerland, (private communication), http://bigwww.epfl.ch/thevenaz/.

[3] A. Gotchev, "Spline and wavelet based techniques for signal and image processing," Doctoral thesis, Tampere University of Technology, Tampere, Finland, 2003, publication 429.

[4] M. Unser, Ph. Thévenaz, and L. Yaroslavsky, "Convolution-based interpolation for fast, high-quality rotation of images," *IEEE Transactions on Image Processing*, vol. 4, no. 10, pp. 1371–1381, 1995.

[5] L. Yaroslavsky and Y. Chernobrodov, "DFT and DCT based discrete sinc-interpolation methods for direct Fourier tomographic reconstruction," in *Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis (ISPA '03)*, vol. 1, pp. 405–410, Rome, Italy, September 2003.

[6] L. Yaroslavsky, *Digital Holography and Digital Image Processing*, Kluwer Academic, Boston, Mass, USA, 2004.

[7] L. Yaroslavsky, "Fast signal sinc-interpolation methods for signal and image resampling," in *Image Processing: Algorithms and Systems*, vol. 4667 of *Proceedings of SPIE*, pp. 120–129, San Jose, Calif, USA, January 2002.

[8] L. Yaroslavsky, "Boundary effect free and adaptive discrete signal sinc-interpolation algorithms for signal and image resampling," *Applied Optics*, vol. 42, no. 20, pp. 4166–4175, 2003.

[9] L. Yaroslavsky, "Shifted discrete Fourier transforms," in *Digital Signal Processing*, V. Cappellini and A. G. Constantinides, Eds., pp. 69–74, Academic Press, London, UK, 1980.

[10] X. Deng, B. Bihari, J. Gan, F. Zhao, and R. T. Chen, "Fast algorithm for chirp transforms with zooming-in ability and its applications," *Journal of the Optical Society of America A*, vol. 17, no. 4, pp. 762–771, 2000.

[11] R. Tong and R. W. Cox, "Rotation of NMR images using the 2D chirp-z transform," *Magnetic Resonance in Medicine*, vol. 41, no. 2, pp. 253–256, 1999.
[12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes. The Art of Scientific Computing*, Cambridge University Press, Cambridge, Mass, USA, 1987.
[13] J. H. Mathew and K. D. Fink, *Numerical Methods Using MATLAB*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1999.
[14] C. Elster and I. Weingärtner, "High-accuracy reconstruction of a function f(x) when only $df(x)/dx$ or $d^2 f(x)/dx^2$ is known at discrete measurement points," in *X-Ray Mirrors, Crystals, and Multilayers II*, vol. 4782 of *Proceedings of SPIE*, pp. 12–20, Seattle, Wash, USA, July 2002.
[15] L. Yaroslavsky, A. Moreno, and J. Campos, "Frequency responses and resolving power of numerical integration of sampled data," *Optics Express*, vol. 13, no. 8, pp. 2892–2905, 2005.
[16] Y. Katiyi and L. Yaroslavsky, "Regular matrix methods for synthesis of fast transforms: general pruned and integer-to-integer transforms," in *Proceedings of International Workshop on Spectral Methods and Multirate Signal Processing (SMMSP '01)*, T. Saramäki, K. Egiazarian, and J. Astola, Eds., Pula, Croatia, June 2001, TICSP Series#13, ISSN 1456-2774, ISBN 952-15-0633-4L.
[17] S. Gepstein, A. Shtainman, B. Fishbain, and L. Yaroslavsky, "Restoration of atmospheric turbulent video containing real motion using rank filtering and elastic image registration," in *Proceedings of the 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, September 2004.
[18] J. Xi and J. F. Chicharo, "Computing running DCT's and DST's based on their second-order shift properties," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 779–783, 2000.
[19] C. F. Gauss, "Nachlass: theoria interpolationis methodo nova tractata," in *Werke*, vol. 3, pp. 265–330, Königliche Gesellschaft der Wissenschaften, Göttingen, Germany, 1866.
[20] M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the history of the fast Fourier transform," *IEEE ASSP Magazine*, vol. 1, no. 4, pp. 14–21, 1984.

L. Yaroslavsky: Department of Interdisciplinary Studies, Faculty of Engineering,
Tel Aviv University, Tel Aviv 69978, Israel

Email: yaro@eng.tau.ac.il

Index