Research Article

Image Interpolation via Gradient Correlation-Based Edge Direction Estimation

Sajid Khan,¹ Dong-Ho Lee²,³ Muhammad Asif Khan,³ Muhammad Faisal Siddiqui,⁴ Raja Fawad Zafar,⁵ Kashif Hussain Memon,² and Ghulam Mujtaba¹

¹Center of Excellence for Robotics, Artificial Intelligence, and Blockchain, Sukkur IBA University, Sukkur, Pakistan

²School of Electrical Engineering, Hanyang University ERICA Campus, Ansan, Republic of Korea

³Department of Electrical Engineering, Sukkur IBA University, Sukkur, Pakistan

⁴Faculty of Engineering, Department of Electrical Engineering, University of Malaya, Kuala Lumpur 50603, Malaysia

⁵Department of Mathematics and Social Sciences, Sukkur IBA University, Sukkur, Pakistan

Correspondence should be addressed to Dong-Ho Lee; dhlee77@hanyang.ac.kr

Received 3 January 2020; Accepted 17 February 2020; Published 21 April 2020

Guest Editor: Rahman Ali

Copyright © 2020 Sajid Khan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper introduces an image interpolation method that provides performance superior to that of the state-of-the-art algorithms. The simple linear method, if used for interpolation, provides interpolation at the cost of blurring, jagginess, and other artifacts; however, applying complex methods provides better interpolation results, but sometimes they fail to preserve some specific edge patterns or results in oversmoothing of the edges due to postprocessing of the initial interpolation process. The proposed method uses a new gradient-based approach that makes an intelligent decision based on the edge direction using the edge map and gradient map of an image and interpolates unknown pixels in the predicted direction using known intensity pixels. The input image is subjected to the efficient hysteresis thresholding-based edge map calculation, followed by interpolation of low-resolution edge map to obtain a high-resolution edge map. Edge map interpolation is followed by classification of unknown pixels into obvious edges, uniform regions, and transitional edges using the decision support system. Coefficient-based interpolation that involves gradient coefficient and distance coefficient is applied to obvious edge pixels in the high-resolution image, whereas transitional edges in the neighborhood of an obvious edge are interpolated in the same direction to provide uniform interpolation. Simple line averaging is applied to pixels that are not detected as an edge to decrease the complexity of the proposed method. Applying line averaging to smooth pixels helps to control the complexity of the algorithm, whereas applying gradient-based interpolation preserves edges and hence results in better performance at reasonable complexity.

1. Introduction

Image interpolation is used to produce a high-resolution image from a low-resolution one. Image interpolation is used in many applications and fields of image processing [1–3]. The increase in the resolution of modern-day display screens demands artifact-free upscaling of videos and images [4]. The TV industry has and is still evolving through the introduction of advanced and high-resolution technologies. The evolution of the TV industry from analog to digital and then from HDTV to SUHDTV increases the significance of image/video interpolation. Video interpolation is needed to

produce a pleasant experience of viewing a low-resolution video on high-resolution displays. A video essentially consists of a sequence of frames, and each frame is represented by an image. Upscaling a video sequence typically amounts to upscaling a series of frames. An image interpolation method should interpolate a low-resolution image in such a way that it produces no artifacts or few artifacts to provide a pleasant viewing experience.

Image interpolation is also used in medical image processing [5, 6] to enhance magnetic resonance images (MRI), picture archiving and communication systems (PACS), computer-assisted surgery (CAS), digital

subtraction angiography (DSA), and computer-aided diagnosis. After the acquisition of medical images, the zooming and rotation of images are required for the purpose of diagnosis and treatment, and interpolation methods are required to perform such operations.

An image contains two regions: uniform regions and edges. Interpolating an edge correctly is the most important requirement of an interpolating algorithm since human eyes are very sensitive to image edges. The production of artifacts along an edge results in an unpleasant viewing experience. Most of the interpolation algorithms that are prominent in the field [6–18] have focused on the interpolation of edges. These algorithms range from simple linear algorithms to complex edge-preserving methods. Linear algorithms fail to preserve most of the edges, whereas complex algorithms preserve edges very well compared to linear algorithms. However, these complex algorithms fail to detect certain edge patterns such as thin edges and sometimes produce artifacts along edges.

Nearest neighbor, bilinear, and bicubic [6] are the most prominent simple methods, providing time-efficient interpolations at the cost of blurring, jaggging, and smoothing along the edges. Paik et al. [10] used cross correlations using horizontal and vertical direction vectors that fail in restoring edges that are not correlated with predefined horizontal and vertical vectors. Allebach and Wong [11] interpolated smooth regions using bilinear interpolation and applied data correction along edges. The problem with EDI is its sensitivity to the texture that can spuriously produce artifacts along the edges. Dong et al. [12] combined a nonlocal aggressive model with a sparse representation model. The sparse representation-based method provides reasonable results, but they are always time-inefficient. Sparse-based methods are dictionary-based, and they change local properties such as image intensity and edge properties, as well as introduce artifacts along the edges. Shi and Shan [14] proposed a method that correlates neighbors of the unknown pixel with six modes and interpolates unknown pixels along the mode that has the closest correlation. The problem with this method is that it uses sixteen pixels for each mode, and the major variation of only one pixel along an appropriate mode might result in the rejection of that mode.

Some researchers have provided particularly promising contributions in the field of image interpolation. Li and Orchard [15] proposed the interpolation method new edge-directed interpolation (NEDI). NEDI calculates local covariance coefficients from a low-resolution input image and then uses these coefficients to produce high-resolution image on the basis of geometric duality between the coefficients of both images. However, NEDI fails to preserve thin edges or high-frequency regions.

Giachetti and Asuni [17] proposed the iterative curvature-based interpolation (ICBI) method that provides better results compared to other conventional algorithms and preserves edges well. ICBI is a two-step interpolation that uses a second-order derivative in the first step and minimization of energy functions in the second step. Minimization of the energy function refines edges, which can result in oversmoothing of

edges. Yang et al. [13] and Timofte et al. [18] worked on a dictionary learning-based interpolation algorithm. Timofte et al. proposed an anchored neighborhood regression (ANR) approach that provides a much faster dictionary learning-based interpolation compared to the method proposed by Yang et al. The problem with a dictionary-based learning approach is that information in an image is replaced using a limited size dictionary that sometimes changes image properties such as intensity and edge shapes.

In this paper, a new interpolation algorithm, named directional gradient-based edge interpolation (DGEI), is proposed. The proposed method categorizes the input image into obvious edges, transitional regions, and uniform regions. Uniform regions are interpolated using simple line averaging, whereas DGEI is applied to obvious edge pixels. DGEI predicts the direction of an edge and interpolates unknown edge pixels in the direction with a high gradient correlation. Transitional edges are interpolated in the direction of neighboring pixels. The proposed method preserves edges with no or minimum artifacts using all possible predefined edge patterns. Simulation results show that the proposed algorithm provides better results compared to bicubic, NEDI, ICBI, and ANR.

The rest of this paper is organized as follows. Section 2 describes some of the conventional methods. Section 3 describes the proposed algorithm. Section 4 describes the performance analysis. Section 5 is the conclusion.

2. Conventional Methods

Before describing the proposed algorithm, NEDI, ICBI, and ANR are briefly discussed since they are state-of-the-art methods and provide good results.

NEDI assumes that the input low-resolution image of size $H \times W$ directly comes from high-resolution output image of size c with pixel to pixel relation of $I_{2i,2j} = X_{i,j}$. Hence, unknown pixel can be calculated using the following equation:

$$I_{2i+1,2j+1} = \sum_{k=0}^1 \sum_{l=0}^1 \alpha_{2k+1,2(j+l)} I_{2(i+k),2(j+l)}. \quad (1)$$

Generalized equation for α is given as

$$\overrightarrow{\alpha} = R^{-1} \overrightarrow{r}, \quad (2)$$

where $R = [R_{kl}]$ and $\overrightarrow{r} = [r_k]$ are high-resolution covariance. Both R_{kl} and r_k can be obtained from Figure 1.

The problem with NEDI is that is assuming that low-resolution images and high-resolution images are symmetrical. This assumption works well for nonedges and smooth edges, but it fails to interpolate when there is an abrupt change in edge, thin edges, or region with texture.

ICBI is a two-step algorithm that first interpolates image in the main diagonal or auxiliary diagonal direction based on the minimum response of second-order derivative. It then refines the initially interpolated values by minimizing three energy terms U_c , U_e , and U_i which are edge sharpness, artifact removal, and preserving isolevel curves. The energy function that requires minimization is given as

FIGURE 1: Correspondence between low- and high-resolution covariance.

$$U(2i+1, 2j+1) = aU_c(2i+1, 2j+1) + bU_e(2i+1, 2j+1) + cU_i(2i+1, 2j+1). \quad (3)$$

U_c , U_e , and U_i are given in the following equations, respectively.

$$U_c(2i+1, 2j+1) = w_1 D_1 + w_2 D_2 + w_3 D_3 + w_4 D_4, \quad (4)$$

$$U_e(2i+2j) = |I_{11}(2i+1, 2j+1)| + |I_{22}(2i+1, 2j+1)|, \quad (5)$$

$$U_i = f(I)|_{2i+1, 2j+1} I(2i+1, 2j+1), \quad (6)$$

where

$$\begin{aligned} D_1 &= |(I_{11}(2i+1, 2j+1) - I_{11}(2i+2, 2j+2))| + |(I_{22}(2i+1, 2j+1) - I_{22}(2i+2, 2j+2))|, \\ D_2 &= |(I_{11}(2i+1, 2j+1) - I_{11}(2i+2, 2j))| + |(I_{22}(2i+1, 2j+1) - I_{22}(2i+2, 2j))|, \\ D_3 &= |I_{11}(2i+1, 2j+1) - I_{11}(2i, 2j+2)| + |I_{22}(2i+1, 2j+1) - I_{22}(2i, 2j+2)|, \\ D_4 &= |I_{11}(2i+1, 2j+1) - I_{11}(2i, 2j)| + |(I_{22}(2i+1, 2j+1) - I_{22}(2i, 2j))|. \end{aligned} \quad (7)$$

The problem with ICBI is that the minimization of energy terms does not consider the properties of the known pixel and that is why it sometimes produces artifacts for low-level upscaling (2x) and most of the time for high-level upscaling (4x or 8x). Production of false isolevel curves because of the energy function U_i is an issue with ICBI.

ANR [18] is a patch-based dictionary learning approach based on minimization of the following function:

$$\min_{\beta} \|y_F - N_i \beta\|_2^2 + \gamma \|\beta\|_2, \quad (8)$$

where N_i is set that consists of K nearest neighbors of the input feature image and γ allows to alleviate the singularity of ill-posed problems and stabilize the coefficient β .

The problem with ANR is that it completely changes some properties of an image such as intensity, production of artifacts in the transitional region, and unnecessary sharpness of edges.

3. The Proposed Method

The objective of this work is to propose a method that provides artifact-free upscaling and preserves the natural properties of the image. Figure 2 shows the flowchart of the algorithm. The proposed algorithm initially calculates the edge map of the low-resolution image by first calculating gradient G_{low} of the input image of size $M \times N$ and then applies hysteresis thresholding to G_{low} to obtain a low-resolution edge map E_{low} . E_{low} contains both obvious and

FIGURE 2: Flowchart of the proposed method.

transition regions since transition regions also have strong gradient values. Transition regions and obvious edge are isolated by applying nonmaximal suppression to produce E_{low}^{trans} and E_{low}^{edge} , respectively. High-resolution edge maps E_{high}^{edge} and E_{high}^{trans} of sizes $2M \times 2N$ are obtained by processing E_{low}^{edge} and E_{low}^{trans} , respectively. Smooth pixels are subjected to the application of simple line averaging, whereas obvious edges in E_{high}^{edge} are processed by directional gradient-based interpolation. Transition regions are interpolated in the direction estimated for their neighboring obvious edge pixel to provide uniformity along edge and transition regions.

3.1. Edge Detection. Hysteresis thresholding-based edge detection is used for edge detection. Hysteresis thresholding [19–24] involves the calculation of two thresholds T_{low} and T_{high} and uses edge-map linking for better true detection and noise reduction compared to single thresholding [25, 26]. The method proposed by Medina-Carnicer et al. [21] was used for the calculation of T_{low} and T_{high} as it provides reasonable pair of low and high thresholds.

The input of the Medina-Carnicer method is the gradient image G_{low} obtained by applying horizontal and vertical Sobel operators [27].

After calculation of T_L and T_H , low- and high-thresholded edge maps E_L and E_H are obtained. E_L contains all the obvious edges along with noisy edges, whereas E_H contains incomplete true edges. For each incomplete edge in E_H , the corresponding connected edge pixels from E_L were obtained and added to the final low-resolution edge map E_{low} . In that way, all the true edges were taken from E_L while ignoring noisy edges, as E_H does not have the representation of noisy edges.

3.2. Isolating Transitions and Obvious Edges. Edges of the edge map E_{low} are thick, as transition regions also have some reasonable gradient values and are detected in the low threshold edge map. To separate transition edges from obvious edges, nonmaximal suppression is applied to gradients at detected edge locations. Nonmaximal suppression is a technique used by Canny [22]. This method is used to thin edges by selecting the edge that corresponds to a maximum gradient value. Canny applied nonmaximal

suppression to the gradient image before thresholding. The proposed method, however, applies nonmaximal suppression only after the application of edge detection. Nonmaximal suppression is applied to the image $G_{low} \times E_{low}$ to obtain an edge map E_{low}^{edge} that has edges with a thickness of only one pixel. These edges are considered as obvious edges because they have a maximum gradient value in a thick detected edge. The transition map that corresponds to the transition regions can be calculated using

$$E_{low}^{trans} = E_{low} - E_{low}^{edge}. \quad (9)$$

Figure 3 shows the results of the isolation of the edge map and transition map. It can be seen from Figure 3 that the edge map obtained using nonmaximal suppression has edges of a thickness of one pixel. The direction obtained using E_{low}^{edge} contains a small chance of error because the direction of an edge can be predicted more precisely if the thickness of the edge is one pixel. E_{low}^{trans} , on the other hand, contains thick transition regions, and the probability of producing an error is high if the direction is obtained separately for transition regions. However, it is clear from Figure 3 that transitional edges in E_{low}^{trans} have the same direction as those of E_{low}^{edge} . Therefore, if an accurate direction for an edge pixel of E_{low}^{edge} is predicted, an interpolation in the same direction of its neighboring transition pixels in E_{low}^{trans} can be performed to produce artifact-free image upscaling.

3.3. Interpolating Edge/Transition Maps. Both E_{low}^{trans} and E_{low}^{edge} are subjected to simple edge map interpolation. The proposed algorithm upscales an edge/transition map by first interpolating along the row direction and then along the column direction. Both row and column interpolation follow the same procedure. The edge/transition map of size $M \times N$ is used as the input edge/transition map for performing edge/transition interpolation along the row direction in order to obtain the row-interpolated edge/transition map E^{Row} of size $2M \times N$. The transpose of E^{Row} is then used as the input image for column interpolation to obtain an interpolated image of size $2N \times 2M$. The transpose of interpolated $2N \times 2M$ image is needed to obtain a final interpolated image E_{high}^{edge} (or E_{high}^{trans} in the case of transition map interpolation) of size $2M \times 2N$.

FIGURE 3: Result of edge and transition region isolation; (a) E_{low} ; (b) E_{low}^{edge} ; and (c) E_{low}^{trans} .

A 3×3 window is moved across every unknown row. The pixel that has edge/transition pixels in any of the three directions (main diagonal, vertical, and auxiliary diagonal) shown in Figure 4 is selected as an edge/transition pixel. Figure 5 shows the results of the application of edge interpolation on E_{low}^{edge} .

3.4. Gradient-Based Direction Estimation. All the unknown pixels that are not detected as edge pixels or transitional regions in E_{high}^{edge} and E_{high}^{trans} are interpolated using simple line averaging. Information for known edge pixels in E_{high}^{edge} is used to estimate the direction of unknown edge pixels. Unknown edge pixels are then interpolated along the estimated direction. Unknown transitional pixels in E_{high}^{trans} are interpolated in the estimated direction of their neighboring edge pixel in E_{high}^{edge} . This section provides the direction calculation methodology for the unknown edges pixels in E_{high}^{edge} .

Figure 6 shows all possible directions that an edge might have. These directions include horizontal, vertical, main diagonal, auxiliary diagonal, left concave, right concave, upward concave, and downward concave. The direction for every unknown edge pixel is determined by using eight available directions.

Every unknown pixel has some cases that show the positions of unknown and known pixels around it. For evaluating the direction, only the properties of the known pixels can be used. Given that known pixels belong to an odd row and odd column, three cases show the positions of known and unknown pixels.

- (1) Unknown pixel is at an even row and even column
- (2) Unknown pixel is at an odd row and even column
- (3) Unknown pixel is at an even row odd column

Figure 7 shows the three cases for the positions for known and unknown pixels. The red circle in each case shows the center unknown pixel, and blue circles show the positions of the known pixels, whereas black pixels show the positions of the surrounding unknown pixels. The case of even rows and even columns does not have known pixels along with adjacent vertical and horizontal directions, and the case of the odd row and even columns does not have

known pixels along the vertical direction, whereas the case of even rows and odd columns do not have known pixels along the horizontal direction. Therefore, the number of possible directions for the first case is six, whereas in the second and third cases, the number of possible directions is seven.

The proposed direction estimation-based method uses edge map E_{high}^{edge} to see if a certain direction is a candidate direction or not and then uses gradient correlation to determine the pixels that can be used for interpolation. A certain direction cannot be selected as a candidate direction if it does not follow the crossing condition, for example, if one wants to see whether or not the main diagonal is a candidate for an unknown edge pixel that is on an even row and even column. The pixels C31 and C51 should be edge pixels in order to satisfy the crossing condition. If any of these two pixels are not edge pixels, then the edge is certainly not moving along the main diagonal. Also, for gradient correlation, a reference gradient value is needed because gradient correlation takes the absolute difference between the reference gradient with respective gradients of possible pixels in that direction. These differences are finally compared with a threshold value to see whether the correlation is in the wanted range or not. Every direction has its own possible reference gradient position, its own “must be edges” condition to satisfy the crossing condition, and its own positions of possible directional members.

Table 1 shows all the information needed to estimate the direction for an unknown pixel that has a position of an even row and even column. Steps for estimating the direction of the unknown edge pixel are

- (1) Take a 9×9 patch around the unknown edge pixel.
- (2) For all directions, if possible references contain an edge pixel, move to step 3 and select the gradient value of that possible reference as a reference gradient. If none of the possible references for a direction has an edge, ignore that direction as the candidate direction.
- (3) If the direction follows the “must be edges” condition given in Table 1, select this direction as the candidate direction and move to step 4. If the “must be edges”

FIGURE 4: Directions selected for edge interpolation. (a) Main diagonal; (b) vertical; and (c) auxiliary diagonal.

FIGURE 5: Results of application of edge map interpolation. (a, c) Patches of E_{low}^{edge} ; (b, d) patches of E_{high}^{edge} .

FIGURE 6: All possible primary configurations of an edge.

FIGURE 7: Three cases for positions of known and unknown pixels: (a) even row even column; (b) odd row even column; (c) even row odd column.

TABLE 1: Information for evaluating edge directions for the case of even rows and even columns.

Direction	Possible directional members	Possible references	Pixels that must be edges
Main diagonal	C11, C29, C13, C31, C51, C69, C53, C71	C31, C51	(C31 & C51)
Left concave	C11, C29, C31, C49, C51, C33, C35, C17	C31, C33	(C49 C51) & (C33 & C31)
Auxiliary diagonal	C65, C47, C67, C49, C33, C15, C35, C17	C49, C33	(C49 & C33)
Right concave	C65, C47, C49, C31, C33, C51, C53, C71	C49, C51	(C31 C33) & (C49 & C51)
Upward concave	C11, C13, C31, C33, C49, C51, C65, C67	C31, C49	(C33 C51) & (C31 & C49)
Downward concave	C15, C17, C31, C33, C49, C51, C69, C71	C33, C51	(C31 C49) (C33 & C51)
Horizontal	No horizontal direction available		
Vertical	No vertical direction available		

TABLE 2: Information for evaluating edge directions for the case of even rows and even columns.

Direction	Possible directional members	Possible references	Pixels that must be edges
Main diagonal	C10, C12, C30, C32, C50, C52, C70, C72	C32, C50, C30, C52	(C30 C32) & (C50 C52)
Left concave	C64, C66, C48, C50, C32, C34, C16, C18	C32, C50, C48	(C48 C50) & (C32 C34)
Auxiliary diagonal	C10, C12, C30, C32, C48, C50, C16, C34, C52, C18	C32, C30, C34	(C48 C50 C52) & (any two of C30, C32, & C34)
Right concave	C30, C32, C34, C48, C50, C52, C64, C66, C70, C72	C50, C48, C52	(C30 C32 C34) & (any two of C48, C50, & C52)
Upward concave	C10, C12, C30, C32, C48, C50, C64, C66	C32, C50	(C30 C32) & (C50 C48)
Downward concave	C16, C18, C32, C34, C50, C52, C70, C72	C32, C50	(C34 C32) & (C50 C52)
Horizontal	C14, C32, C50, C68	C32	(C32 & C50)
Vertical	No vertical direction available		

TABLE 3: Information for evaluating edge directions for the case of even rows and even columns.

Direction	Possible directional members	Possible references	Pixels that must be edges
Main diagonal	C2, C4, C20, C22, C40, C42, C60, C62, C78, C80	C22, C60	(C22 C60) & (C40 C42)
Left concave	C2, C8, C20, C22, C24, C26, C40, C42	C40, C42	(C20 C22) & (C40 C42) & (C24 C26)
Auxiliary diagonal	C6, C8, C24, C26, C40, C42, C56, C58, C74, C76	C24, C58	(C24 C58) & (C40 C42)
Right concave	C40, C42, C56, C58, C60, C62, C74, C80	C40, C42	(C56 C58) & (C40 C42) & (C60 C62)
Upward concave	C2, C20, C22, C40, C42, C56, C58, C74	C22, C40, C58	(C24 C42 C60) & (any two of C22, C40 & C58)
Downward concave	C8, C24, C26, C40, C42, C60, C62, C80	C24, C42, C60	(C22 C40 C58) & (any two of C24, C42 & C60)
Horizontal	No horizontal direction available		
Vertical	C38, C40, C42, C44	C40	(C40 & C42)

condition is not justified, ignore that direction as the candidate direction.

- (4) Take the absolute difference of the gradient of “possible directional members” with the reference gradient to calculate the gradient correlation. Select all those pixels as interpolating pixels that have a gradient correlation less than $T = 21$.
- (5) Among all candidate directions, select the direction that has more pixels with a gradient correlation value of less than T . Store their gradient correlation in an array D_G . D_G will then be used in the coefficient calculation for interpolation.

Directions estimated for unknown edge pixels that have positions of odd rows and even columns or even rows and

odd columns can be calculated in a similar way using Tables 2 and 3, respectively.

3.5. Interpolation Using the Estimated Direction. An edge pixel can be interpolated using

$$I_{\text{high}}^{\text{edge}} = \sum_{r=0}^{N_G-1} I_r^{\text{GC}} \times C_{G(r)}^{\text{GC}} * C_{D(r)}^{\text{GC}}, \quad (10)$$

where I_r^{GC} , $C_{G(r)}^{\text{GC}}$, and $C_{D(r)}^{\text{GC}}$ are the image intensity, the gradient coefficient, and the distance coefficient of the r th pixel with a gradient correlation less than T . C_G and C_D can be calculated using

FIGURE 8: Cropped regions after upscaling ($\times 2$); (a, g, m) original; (b, h, n) bicubic; (c, i, o) NEDI; (d, j, p) ICBI; (e, k, q) ANR; (f, l, q) proposed.

FIGURE 9: Error patches for cropped regions after upscaling ($\times 2$); (a, f, k) bicubic; (b, g, l) NEDI; (c, h, m) ICBI; (d, i, n) ANR; (e, j, o) proposed.

$$C_G = e^{-\sqrt[N]{D_G}}, \quad (11)$$

$$\hat{C}_D = e^{-(x^2+y^2)/2\sigma^2}, \quad (12)$$

$$C_D = \frac{\hat{C}_D}{\max(\hat{C}_D)}. \quad (13)$$

Here, $N = 4$ and $\sigma = 1.5$ are parameters chosen as a result of simulations. The variables x and y in equation (12) are the x -directional and y -directional distances of a pixel from the center pixel. There is no need to dynamically calculate C_D for every pixel. This is because each pixel in a 9×9 patch is located at a fixed distance from the center pixel. Therefore, a 9×9 matrix of coefficients C_D can be calculated only once and used for all pixels. C_G , however, requires

dynamic calculation for every pixel because the gradient correlation is different for all pixels. Before applying C_D and C_G for interpolation, both coefficients should be normalized in such a way that it follows the given condition:

$$\sum_{r=0}^{N_G-1} C_{G(r)}^{GC} * C_{D(r)}^{GC} = 1. \quad (14)$$

After interpolating an edge pixel, its neighboring region needs to be scanned for the interpolation of transitional pixels. If the edge is moving in the vertical direction with zero slope or some slope, all of the transitional pixels that are connected to the left and right are scanned and interpolated in the same direction as the edge pixel. Unknown transitional pixels on the left or right direction of the interpolated edge pixel are checked for whether they are on an even row and even column, even row and odd column, or odd row and even column and are interpolated in that direction using information available in the corresponding Tables 1–3. A transitional pixel is interpolated by selecting a reference gradient and then following steps 4 and 5 of the previous subsection by calculating the gradient correlation D_G , using the gradient of the selected transitional pixel as the reference. After calculating D_G and indices for members of the array D_G , interpolation can be applied to the transitional pixel using equations (10)–(14). However, if an edge is moving in the horizontal direction, transitional pixels connected to its upward and downward directions are interpolated similarly. A few exceptional cases might occur when the direction estimated for an edge pixel is vertical or horizontal; however, some neighboring transitional pixels are located, where they do not have known vertical or horizontal pixels. In that case, the transitional pixel is interpolated by using the mean of the two closest pixels in that direction using line averaging.

4. Results

4.1. Subjective Evaluation. The proposed method is compared with the bicubic, NEDI, ICBI, and ANR approaches by applying algorithms on grayscale images provided in [28]. Both subjective and objective evaluations of all algorithms are performed in order to quantitatively compare the quality of the images created by different methods and the related computational costs. The tic toc function in Matlab 2018b was used to calculate the execution time. Bicubic interpolation was applied using the MATLAB built-in function named `interp2`, whereas the source codes for NEDI, ICBI, and ANR were kindly provided by Prof. Xin Li, Prof. Andrea Giacheti, and Dr. Yang, respectively.

Figure 8 shows some cropped regions of some grayscale test images that are upscaled by a factor of two. The input image was first downsampled by a factor of two, and then all algorithms were applied to the downsampled images to see whether or not they preserve distorted edges. It can be seen that the bicubic and NEDI algorithms failed to preserve edges. ANR provided some sharp edges, but changed the edge structures and intensity of those patches. ICBI preserved edges well; however, it oversmoothed the edges with

TABLE 4: Error indices comparison of all algorithms on the patches in Figure 9.

Image	Bicubic	NEDI	ICBI	ANR	Proposed
Airplane	646.07	394.29	377.38	606.04	336.86
Beehive	573.86	446.54	402.01	683.11	362.58
Woman	388.10	334.40	332.03	468.28	255.62

TABLE 5: PSNR (dB) comparison of all algorithms.

Image	Bicubic	NEDI	ICBI	ANR	Proposed
Airplane	25.762	27.509	27.873	20.075	28.679
Banana	34.23	36.124	36.853	22.739	36.931
Barbara	24.126	25.526	24.96	21.552	25.594
Brush	29.818	31.065	31.226	21.309	31.641
Beehive	31.643	33.245	33.664	20.723	33.921
Bridge	23.88	25.12	25.246	20.326	25.857
Brief case	27.98	29.532	30.254	22.894	30.644
Coffee maker	29.365	31.546	32.353	24.267	32.912
Elephant	26.051	27.577	27.611	19.845	28.164
Feather	25.703	27.14	27.208	21.599	27.953
Golf cart	27.529	29.011	29.306	21.173	30.075
Grater	30.47	31.772	32.126	20.626	32.618
Lena	30.028	32.619	33.523	22.764	32.958
Mail box	26.383	27.986	28.504	22.143	29.134
Pond	23.203	24.453	24.107	18.731	25.172
Shopping cart	24.412	25.622	25.52	19.362	26.705
Tire	25.873	26.859	27.025	19.566	27.901
Traffic cone	29.102	30.683	31.319	23.5	32.379
Trash can	26.076	27.137	27.739	21.748	28.728
Woman	36.964	39.25	40.662	21.365	40.991
Average	27.923	29.488	29.851	21.315	30.4478

the postprocessing step of refinement, as seen in Figure 8(j). The proposed method preserved edges well compared to other algorithms and also effectively preserved the contrast and shapes of the edges.

The error index (EI) defines an algorithm's cumulative magnitude of error for a patch and is calculated for all patches of size $M \times N$, given as

$$EI = \sum_{r=1}^{M \times N} |\text{Patch}_{\text{orig}} - \text{Patch}_{\text{algo}}|. \quad (15)$$

Error indices calculated for patches in Figure 9 are given in Table 4. Error indices for the ICBI method and the proposed method are the lowest with the proposed method showing superior results. ANR and bicubic have the highest EI, whereas NEDI provides EI that is better than ANR and Bicubic.

4.2. Objective Evaluation. The proposed algorithm outperformed conventional algorithms based on not only subjective evaluations but also objective evaluations. Table 5 shows the peak signal to noise ratio (PSNR) comparison of all algorithms for 20 images. On average, the proposed algorithm provides a PSNR that is 2.5181 dB higher than bicubic, 0.959 dB higher than NEDI, 0.594 dB higher than ICBI, and 9.13 dB higher than ANR.

TABLE 6: Elapsed CPU time (sec) comparison of all algorithms.

Image	Bicubic	NEDI	ICBI	ANR	Proposed
Airplane	0.0449	23.654	7.1067	3.2999	2.063
Banana	0.0042	21.548	6.1403	2.8692	2.2415
Barbara	0.0052	37.441	10.76	4.5643	5.2492
Brush	0.0052	25.22	7.4608	3.373	2.4569
Beehive	0.0037	24.726	6.8956	3.2409	2.3253
Bridge	0.0038	23.011	6.7087	3.0958	3.5507
Brief case	0.0037	21.374	6.2375	2.8979	2.1482
Coffee maker	0.004	27.333	7.8577	3.5475	2.4451
Elephant	0.0031	20.428	5.9387	2.8511	2.3893
Feather	0.0037	23.063	6.6781	3.1025	2.9475
Golf cart	0.0038	24.581	7.1438	3.2469	2.6199
Grater	0.0034	19.554	5.6954	2.738	1.7863
Lena	0.0036	23.046	6.7263	3.0986	2.3076
Mail box	0.0041	22.974	6.6943	3.1225	3.1428
Pond	0.0035	22.799	6.6565	3.0779	3.0704
Shopping cart	0.0036	22.479	6.6835	3.1652	2.6862
Tire	0.0042	22.555	6.5826	3.0227	2.0995
Traffic cone	0.0036	22.765	6.6323	3.0833	2.8033
Trash can	0.003	20.275	5.9118	2.7633	1.9244
Woman	0.0037	22.635	6.6033	3.0823	2.4294
Average	0.00588	23.572	6.8557	3.1621	2.6343

Table 6 shows the CPU elapsed time for all algorithms. CPU elapsed time for bicubic is the lowest compared to other methods because it is a very basic interpolation method and the function interp2 interpolates an image using the mex compiler of MATLAB. On average, the proposed method is almost 9 times faster than NEDI, 2.61 times faster than ICBI, and 1.2 times faster than ANR.

5. Conclusions

The preservation of edges is considered to be the most important parameter in the field of image interpolation. Linear interpolation methods produce blurring effects while complex algorithms result in oversmoothed edges or add artifacts to the image. These artifacts are a direct consequence of post-processing conducted on the interpolated images to either enhance or refine edges. In this paper, a gradient-based edge-preserving interpolation method is proposed. The proposed method uses a combination of low-complexity and efficient hysteresis thresholding based edge detection, the isolation of edges and transitional regions, simple edge map interpolation, direction estimation using the edge map and gradient map, followed by gradient and distance coefficient based interpolation. Simulation results show that the proposed method provides better performance with regard to both subjective and objective evaluations, while accurately preserving edges and image details when compared to leading alternative methods.

Abbreviations

ICBI: Iterative curvature-based interpolation
 ANR: Anchored neighborhood regression
 NEDI: New edge-directed interpolation
 EI: Error index
 PSNR: Peak signal to noise ratio.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

- [1] S. Khan and D. Lee, "Efficient deinterlacing method using simple edge slope tracing," *Optical Engineering*, vol. 54, no. 10, Article ID 103108, 2015.
- [2] S. Ng, "A brief history of entertainment technologies," *Proceedings of the IEEE*, vol. 100, pp. 1386–1390, 2012.
- [3] A. Amanatiadis and I. Andreadis, "A survey on evaluation methods for image interpolation," *Measurement Science and Technology*, vol. 20, no. 10, pp. 316–324, 2009.
- [4] R. W. Kubey, "Television and aging: past, present, and future," *The Gerontologist*, vol. 20, no. 1, pp. 998–1004, 1980.
- [5] 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.
- [6] R. C. Gonzalez, *Digital Image Processing*, Prentice-Hall, Upper Saddle River, NJ, USA, 2009.
- [7] B. Jin, J. G. Kuk, and N. I. Cho, "A gradient guided deinterlacing algorithm," in *Proceedings of the 2012 19th IEEE International Conference on Image Processing*, pp. 853–856, Orlando, FL, USA, October 2012.
- [8] D. H. Lee, "A simple, high performance edge-adaptive deinterlacing algorithm with very low complexity," in *Proceedings of the 2012 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 636–637, Las Vegas, NV, USA, January 2012.
- [9] X. Chen, G. Jeon, and J. Jeong, "Filter switching interpolation method for deinterlacing," *Optical Engineering*, vol. 51, no. 10, 2012.
- [10] J. H. Paik, K. P. Hong, J. K. Paik, and J. H. Hwang, "Image sequence interpolation for improving the resolution of the magnified image," in *Proceedings of the APCCAS'96-Asia Pacific Conference on Circuits and Systems*, pp. 544–547, Seoul, Republic of Korea, November 1996.
- [11] J. Allebach and P. W. Wong, "Edge-directed interpolation," in *Proceedings of the 3rd IEEE International Conference on Image Processing*, pp. 707–710, Lausanne, Switzerland, September 1996.
- [12] W. Dong, L. Zhang, R. Lukac, and G. Shi, "Sparse representation based image interpolation with nonlocal autoregressive modeling," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1382–1394, 2013.
- [13] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.
- [14] J. Shi and Z. Shan, "Image interpolation using a variation-based approach," in *Proceedings of the 2011 8th International Conference on Information, Communications & Signal Processing*, pp. 1–4, Singapore, December 2011.
- [15] X. Li and M. T. Orchard, "New edge-directed interpolation," *IEEE Transactions on Image Processing: A Publication of the IEEE Signal Processing Society*, vol. 10, no. 10, pp. 1521–1527, 2001.

- [16] N. Asuni and A. Giachetti, "Accuracy improvements and artifacts removal in edge based image interpolation," *International Conference on Computer Vision Theory and Applications*, vol. 8, pp. 352–355, 2008.
- [17] A. Giachetti and N. Asuni, "Real-time artifact-free image upscaling," *IEEE Transactions on Image Processing*, vol. 20, no. 10, pp. 2760–2768, 2011.
- [18] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1920–1927, Sydney, Australia, December 2013.
- [19] E. R. Hancock and J. Kittler, "Adaptive estimation of hysteresis thresholds," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 196–201, Maui, HI, USA, June 1991.
- [20] R. Medina-Carnicer, A. Carmona-Poyato, R. Muñoz-Salinas, and F. J. Madrid-Cuevas, "Determining hysteresis thresholds for edge detection by combining the advantages and disadvantages of thresholding methods," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 165–173, 2009.
- [21] R. Medina-Carnicer, F. J. Madrid-Cuevas, A. Carmona-Poyato, and R. Muñoz-Salinas, "On candidates selection for hysteresis thresholds in edge detection," *Pattern Recognition*, vol. 42, no. 7, pp. 1284–1296, 2009.
- [22] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.
- [23] W. Rong, Z. Li, W. Zhang, and L. Sun, "An improved CANNY edge detection algorithm," in *Proceedings of the IEEE International Conference on Mechatronics and Automation*, pp. 577–582, Tianjin, China, August 2014.
- [24] Y. Yu, Z. Li, B. Liu, and X. Liu, *An Adaptive Unimodal and Hysteresis Thresholding Method, Bio-Inspired Computing-Theories and Applications*, Springer, Berlin, Germany, 2014.
- [25] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
- [26] P. L. Rosin, "Unimodal thresholding," *Pattern Recognition*, vol. 34, no. 11, pp. 2083–2096, 2001.
- [27] W. Burger and M. J. Burge, *Digital Image Processing: An Algorithmic Introduction Using Java*, Springer, Berlin, Germany, 2016.
- [28] Computer Vision and Pattern Recognition Group, http://marathon.csee.usf.edu/edge/edge_detection.html, 2019.