Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 24758, 10 pages
doi:10.1155/2007/24758

Research Article

Utilizing Cross-Layer Information to Improve Performance in JPEG2000 Decoding

Hannes Persson,[1] Anna Brunstrom,[1] and Tony Ottosson[2]

[1] *Department of Computer Science, Karlstad University, 651 88 Karlstad, Sweden*
[2] *Department of Signals and Systems, Chalmers University of Technology, 412 96 Gothenburg, Sweden*

Received 1 May 2007; Accepted 28 July 2007

Recommended by Stavros Kotsopoulos

We focus on wireless multimedia communication and investigate how cross-layer information can be used to improve performance at the application layer, using JPEG2000 as an example. The cross-layer information is in the form of soft information from the physical layer. The soft information, which is supplied by a soft decision demodulator, yields reliability measures for the received bits and is fed into two soft input iterative JPEG2000 image decoders. When errors are detected with the error detecting mechanisms in JPEG2000, the decoders utilize the soft information to point out likely transmission errors. Hence, the decoders can correct errors and increase the image quality without making time-consuming retransmissions. We believe that the proposed decoding method utilizing soft information is suitable for a general IP-based network and that it keeps the principles of a layered structure of the protocol stack intact. Further, experimental results with images transmitted over a simulated wireless channel show that a simple decoding algorithm that utilizes soft information can give high gains in image quality compared to the standard hard-decision decoding.

1. INTRODUCTION

Multimedia is predicted to be one of the main applications in future wireless systems. When multimedia is transmitted over a noisy channel, the quality of the multimedia can unfortunately be affected by transmission errors. To mitigate the transmission errors a common technique at the link and the transport layer is to retransmit the erroneous or lost data. This is however not always the best solution, especially when considering delay sensitive applications, because retransmissions introduce delays and result in inefficient use of the channel.

Even though the multimedia user has high demands on the delivered data (e.g., constraints on delay and quality), limitations in human perception allow minor quality degraded multimedia to be forwarded to the user as long as the main information is conveyed in time. Hence, a tradeoff between the transmission delays and the quality of the received multimedia can be made to be able to reach the play out deadline. The tradeoff between delay and quality causes the sender and the receiver to have strategies apart from retransmissions for dealing with transmission errors. These strategies include, for example, error concealment or applying forward error correction (see review in [1]).

The multimedia applications possess knowledge about the structure of the data. In addition, many multimedia codecs that exist today, for example, JPEG2000 and MPEG-4, support features for error resilience [2] allowing a graceful multimedia quality degradation upon errors. Adding lower-layer knowledge about the channel conditions at the application layer can strengthen the error handling capabilities of the application even further. An example of this approach is joint source channel decoding (JSCD), which is an optimized technique to achieve high gains in multimedia quality. A comprehensive survey of JSCD is found in [3] and novel applications are presented, for example, in [4, 5]. However, the JSCD technique is impractical in a layered protocol stack due to the iterative communication between channel and source decoders. A general and simplified decoding technique would ease the deployment in an IP-based wireless network.

In this paper, the considered scenario concerns multimedia traffic transferred from a server in the fixed network to a mobile wireless host (see Figure 1). As an example

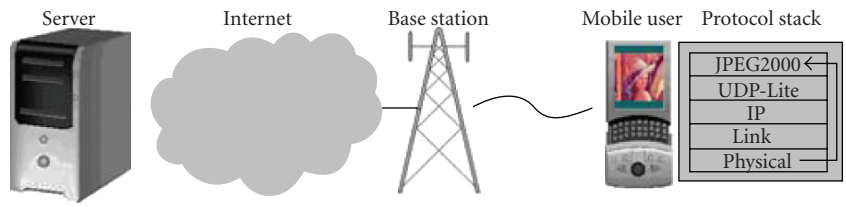


Figure 1: Considered scenario.

application the international standard for still image compression JPEG2000 [6] is used. The target environment concerns wireless channels where multiple bit errors can occur. Images that are sent over these channels should contain some error-resilient information to be able to detect the bit errors. The error resilience is provided by the JPEG2000 standard and is encoded at the sender side. Two soft input iterative JPEG2000 decoders that combine the error resilience and the soft information have been implemented. Experiments performed with these decoders and a simulated wireless channel illustrate that image quality can be improved significantly even with a simple decoding algorithm and without making retransmissions or applying channel coding. Under favorable conditions, high gains in image quality are observed.

We investigate a general and more simplified decoding technique compared to the JSCD technique, where the decoding at the source and channel are independent processes and hence are more suitable for a general IP-based network. More specifically, we investigate how lower-layer knowledge can be used as cross-layer information at the application layer to correct transmission errors. We propose that the knowledge about the channel conditions is in the form of soft information, similar to the information used in JSCD. The soft information yields reliability measures for the received bits and is generated from channel observations in the receiver's physical layer. The soft information gives a structured and channel independent representation of the channel condition. Once generated in the physical layer, the soft information should be transferred to the application through well-defined interfaces. This does not violate the principles of the layered structure of the protocol stack. Without any channel specific knowledge the application can be optimized on the receiver side. Hence, this optimization could be used for different channels. Further, through this approach a persistent and error-resilient source does not have to be transcoded each time a new channel is prevailing.

We consider deployment of soft information in an IP-based wireless network. Although this deployment is thoroughly discussed in [7], we will briefly put forward two important aspects: the necessity to modify the interfaces between the layers to facilitate the propagation of soft information from the physical layer and the use of bit error transparent protocols. Soft information should be forwarded to the application layer through a cross-layer framework. Intermediate layers could also take advantage of soft information and adapt accordingly to the channel. Further, the underlying link and transport protocols must allow erroneous payload data to be forwarded to the application layer. For example, UDP-Lite [8], TCP-L [9], and DCCP [10] are transport protocols that are transparent to bit errors (see overview in [11]). Since the modification proposed in this paper is limited to the receiver, we can assume control of the whole protocol stack in the mobile terminal.

The remainder of this paper is organized as follows. Section 2 defines soft information as used in this work and introduces JPEG2000 and its error-resilient mechanisms. A description of the soft input iterative JPEG2000 decoders is conducted in Section 3 followed by a description of the experimental set-up in Section 4. Section 5 presents the experiment results and Section 6 concludes the paper with some remarks.

2. SOFT INFORMATION AND JPEG2000

This section defines soft information as used in this work. A brief introduction to the example application, JPEG2000, is given, followed by a description of its error-resilient mechanisms. The overview description of JPEG2000 relies heavily on [12, 13].

2.1. Definition of soft information

Soft information can be generated directly from channel observations or when channel decoding is used from a soft output channel decoder. As defined in this paper, the soft information is calculated from channel observations and is based on the Euclidean distance between the received symbol, r, and the modulation constellation symbols, s, in the demodulator. The received symbol, r, denotes a noisy observation at the demodulator, $r = s + n$, where n is the channel noise. Assuming M-ary QAM (quadrature amplitude modulation), the number of bits k for a modulation symbol equals $\log_2 M$. The soft information value for bit b_i ($i = 0, \ldots, k-1$) is described by the log-likelihood ratio (LLR) (as described, e.g., in [14–16]):

$$\mathrm{LLR}(b_i) = \log \frac{P(b_i = 0|r)}{P(b_i = 1|r)} = \log \frac{\sum_{\forall s \in b_i = 0} P(r|s)}{\sum_{\forall s \in b_i = 1} P(r|s)}. \quad (1)$$

The LLR is calculated for each received bit, b_i, and reveals both the binary value and the reliability of the bit. In (1), $P(b_i = 0|r)$ and $P(b_i = 1|r)$ express the conditional probability for receiving a binary value of 0 and 1 respectively, given the received symbol r. Source bits are assumed to be independent and identically distributed. Further in (1), $P(r|s)$ is

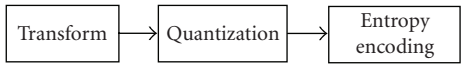


Figure 2: Block diagram of JPEG2000.

the conditional probability, the so-called a posteriori probability (APP), for receiving symbol r, given the transmitted symbol s. The LLR can be interpreted as a quality measure of the estimation that symbol r has been received correctly, given the transmitted symbol s. If an additive white Gaussian noise (AWGN) channel is used, $P(r|s)$ has the form of a Gaussian probability density function [15]. The LLRs for the received bits are assumed to be generated in the physical layer and delivered to the application. It is reasonable to state that the resolution of a soft information value should be x bits per L bit data block [7]. For example, if L equals 1 and x equals 3, the ratio would equal 3 bits of soft information for every data bit. Further, the LLR is transparent to modulation type, thus (1) is applicable when considering systems with arbitrary modulation type.[1] For example, when calculating the LLR for binary phase shift keying (BFSK) and for 64-quadrature amplitude modulation (QAM), (1) will consider the two symbols available for BFSK and the 64 symbols available for 64-QAM, respectively. Since the LLR is transparent to the modulation type in the physical layer, the transparency of the protocol structure is preserved.

2.2. JPEG2000 compression engine

The JPEG2000 compression engine consists of three main steps (see Figure 2). Firstly, the encoder applies a wavelet transform on the source image data. Secondly, the transformed data are quantized, and lastly, they are entropy encoded with an arithmetic encoder. The output forms the final code stream preceded by a header containing vital auxiliary image data. Image decoding is conducted by the inverse for each step.

The wavelet transform is a combination of applying a low and a high pass filter vertically and horizontally on the image to be able to explore local frequency characteristics. The wavelet transform produces four subbands for each decomposition level. The wavelet transform is repeatedly applied to the low pass filter output until a desired decomposition level is reached.

To be able to achieve distortion scalability in an image (i.e., an image represented with low quality and hence a low bit-per-pixel value) JPEG2000 involves bit-plane coding of subbands. The subband samples (coefficients) are coded one by one from the most significant bit to the least significant bit. Discarding the least significant bits for a sample will lead to a distorted image due to the information loss. Bits with the same significance form a bit plane. In a bit plane, subband samples contribute only with one bit each. All the bits in a bit plane are coded in only one of a total of three coding passes. Each one of the three coding passes collects contextual information about the bit-plane data.

The spatial restriction of the layered bit planes is the code block dimension. The code block dimension follows the form $2^n \times 2^n$ with a default dimension of 64×64 pixels. The data within a code block are arithmetically encoded and the arithmetic encoding can be performed on an entire code block. However, for error-resilient reasons, the arithmetic encoding is performed on smaller units, that is, on every single coding pass.

2.3. Error resilience in JPEG2000

A JPEG2000 code stream starts with a main header. The vital auxiliary data of an image (e.g., size of the image and the number of colors used) are stored in the main header. To be able to start the decoding of an image the main header must be correct. Smaller units called packets store the code block data and each packet has a packet header. The packet header stores auxiliary information about the code block (e.g., number of bytes in each code block and where the code-block is located in the image). To be able to decode the data inside the packet, the packet header must also be protected. This can be achieved by the so-called unequal error protection (UEP) techniques, by first moving the packet headers to the main header and then transferring the main header in a reliable manner over the network. UEP techniques for error robust JPEG2000 header and image data transfer are different kinds of retransmissions strategies, work that is identified in [18], or error correcting codes applied to the packet headers as suggested in [19, 20].

A JPEG2000 decoder, which takes advantage of the error-resilient mechanisms, will not utilize a coding pass that contains a bit error or coding passes in the remaining bit-planes that occur after the bit error. Hence the subband samples will be narrow values. This baseline resilient method of error handling is called error concealment. (The error concealment method is comparable to distortion scalability. The former of them will, however, be conducted in a more uncontrolled fashion.) This behavior can be compared to a baseline decoder with no error resilience, which will use all available data including the samples that are in error. For both cases, quality degradation will take place in the decoded images. However, when applying error concealment, the quality degradation is reduced.

To conceal bit errors, the bit errors must first be detected in the arithmetic decoder. In the JPEG2000 standard, the arithmetic codec has mechanisms to be able to detect bit errors.[2] We use a subset of these mechanisms: (1) restart the arithmetic coder for each coding pass (three coding passes in every bit plane), (2) when restarting the arithmetic coder, use a predictable error-resilient termination policy for every coding pass (consult [13] for an in-depth description) and, (3)

[1] Note that the LLR may not be an optimal representation of the channel. It depends on the modulation whether the LLR provides sufficient statistics or not [17].

[2] The arithmetic decoder has not been tested to see to what extent it lets bit errors pass through undetected.

reset the arithmetic coding context states after each coding pass to decouple the coding passes. Restarting the arithmetic coder and resetting the probability estimation for every coding pass helps to decouple the coding passes. By introducing the termination for each coding pass and explicitly signaling every coding pass length in the packet header, these mechanisms will give some extra data overhead (see [13, pages 509–511]). Resetting the context states also reduces the coding efficiency (see [13, page 503]).

3. UTILIZING SOFT INFORMATION IN JPEG2000

Two algorithms have been developed to process and utilize the soft information. Our goal is to investigate whether soft information is useful at the application layer and whether there could be image quality gains with different algorithms. The following subsections discuss these algorithms in more detail.

3.1. General idea

The arithmetically encoded coding passes are the smallest units of coded image data in JPEG2000. If a bit error has occurred in the bit-stream and the arithmetic decoder and the encoder become unsynchronized with each other in the current coding pass, the bit error can be detected if the error-resilient mechanisms discussed above are in use. After a bit error has been detected, the decoder discards (i.e., conceals) the image data in the coding pass that are currently being decoded and thus reduces the effect of bit errors. The baseline resilient decoder will then stop the decoding of the following coding passes and bit planes in the current code block.

By utilizing the soft information and the redundancy added by the arithmetic encoder, it is possible to correct errors. Sometimes the error correction will stop due to limitations in the algorithms, error detection, or in the soft information. However, by salvaging some data from being concealed, the resulting image quality will be improved. Two algorithms have been developed to process the soft information. The main purpose of the algorithms is to evaluate new likely bit-sequences transmitted by the sender.

As described in Section 2, a soft information value is based on the LLR, which reveals the reliability of the bit. From soft information values it is possible to find the most uncertain bits in the received bit-sequence. These bits are in a potential error state when an erroneous bit-sequence is detected. New bit-sequences are formed by swapping the binary values of these bits to their counterparts. When a new bit-sequence has been created, it is evaluated for correctness with the arithmetic decoder, hence making the process iterative. Algorithm 1 outlines the pseudocode for a general iterative decoding algorithm with soft information.

The main difference between the algorithms is how they process the soft information to find new bit-sequences. From a soft information point of view, it is less probable that the new bit-sequences that are generated are correct compared to the originally rejected bit-sequence. Thus, the degree of effectiveness of an algorithm should be based on how well it can find the next most probable bit-sequences. We consider two soft input iterative JPEG2000 decoders implementing an optimal and a heuristic algorithm, respectively.

```
foreach code block
    extract coding passes from code block
    i = 0 //code pass counter
    do
        result = arithmetic decode coding pass i
        if(result == error)
            if(FIRST_ITERATION)
                j = 0 //iteration counter
            else if(MAX_ITERATION)
                conceal remaining coding passes
                break while-loop
            else
                j = j + 1
            find new bit-sequence j for coding pass i
        else
            i = i + 1
    while(result == error || i < MAX_PASSES)
```

Algorithm 1: Pseudocode for a general iterative decoding algorithm.

3.2. Optimal algorithm

The soft information values calculated from (1) have the property of being additive. The sum of the adding of the values corresponds to the probability of a bit-sequence being correct. The mathematical background of using a logarithmic scale for the LLR is that it is more convenient to work with summations than multiplications for small values. A calculated sum from adding the soft values of the swapped bits in a bit-sequence represents the accumulated probability of the bit-sequence being correct. This implies that it is possible to develop an algorithm that evaluates the soft information and generates a list of optimally ordered bit-sequences. This optimal algorithm (previously presented in [21]) thus finds the most probable bit-sequences which contain one or several swapped bits.

The implementation of the optimal algorithm builds a binary tree structure considering the m most uncertain bits based on the corresponding soft information values in every n long bit-sequence (where $m \leq n$). When considering the m most uncertain bits, the height of the tree becomes m and is occupied by 2^m leaves. The tree supplies a structured way of finding the m most probable correct bit-sequences. The algorithm recursively traverses the tree, from the top root node down to every leaf node, and calculates the sum of the soft information values for each bit-sequence. Depending on whether a left or a right sub tree is traversed, the algorithm swaps the value of a bit and thus generates new bit-sequences. The implementation of this algorithm is not optimized in terms of efficiency and computational complexity.

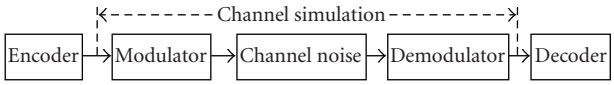


Figure 3: Experiment set-up.

3.3. Heuristic algorithm

We also consider a simple algorithm (previously presented in [22]) that follows a heuristic behavior to evaluate the soft information. On the basis of soft information it picks a small number of bits to swap and generates new probable correct bit-sequences. This algorithm considers a small number of the most uncertain bits and toggles each bit to find a probable correct bit-sequence.

This algorithm first swaps the value of the most uncertain bit and then makes a new decoding pass of the corresponding bit-sequence that previously failed. If the bit-sequence is still in error, the next most uncertain bit is swapped, and an additional decoding pass takes place. This process will continue until a correct bit-sequence is found or until a predefined number of bits have been swapped individually. When a predefined number of bits has been swapped individually the process is extended in such a manner that the decoder permanently swaps the most uncertain bit and then starts to swap the next most uncertain bit.

4. EXPERIMENTAL SET-UP

A number of experiments has been run to evaluate the performance of the algorithms. An overview of the experimental set-up is given in Figure 3. Besides the encoder and the decoder software components, Figure 3 depicts three middle components that are incorporated into the channel simulator software. We assume in our experiments that the underlying layers are able to forward the soft information and accept bit errors in the payload. A description of the experimental set-up of the JPEG2000 codec and the wireless channel follows in this section.

4.1. Image encoder set-up

To evaluate the performance of the algorithms, four image motifs are used in the experiments (Lena, Goldhill, Boat, and Peppers). All the images are JPEG2000 encoded[3] with 1 bit-per-pixel (bpp) including all side information. This setting gives subjectively small image quality degradation compared with the original image. In the experiments, the code block size varies between the values 4×4, 16×16, 32×32 and 64×64. A prerequisite for finding bit errors is that the encoder moves the packet headers to the main header and encodes the error-resilient mechanisms mentioned in Section 2.

[3] Encoder software provided by JJ2000 v4.1 (jj2000.epfl.ch).

4.2. Channel set-up

Simulations of the wireless channel are made by modulating the bits using 16-QAM and transmitting the resulting symbols over an AWGN channel.[4] The signal-to-noise ratio per bit (SNR = E_b/N_0) for the channel ranges from 5 to 16 dB. This ratio measures the relative power of the signal and the noise. An SNR of 5 dB implies a high noise level and 16 dB implies an almost error free channel.

The software for the channel simulates the middle three components shown in Figure 3. The channel simulation is only applied to the image data and no bit errors will thus occur in the vital main header where the packet headers are stored. No retransmissions, channel coding, or interleaving are applied to the image data. The QAM symbols are Gray coded, however. Finally, the hard decided bits and the corresponding soft information[5] are generated in the demodulator module and then later forwarded to the soft input iterative JPEG2000 decoder.

4.3. Image decoder set-up

Two soft input iterative JPEG2000 decoders[6] integrate the optimal and the heuristic algorithm, respectively. As stated in Section 3.2, the implementation of the optimal algorithm is based on a tree structure; hence it is named the tree decoder. The modified decoders assume that the auxiliary image data, that is, the image header, are transmitted in a reliable manner.

The number of iterations, m, in the tree decoder is varied between 5, 10, and 15. These settings will enable the tree decoder to correct up to 2, 3, and 4 bit errors, respectively. The number of attempts before the heuristic decoder stops to generate new bit-sequences from the soft information is set to 10. The number of attempts, before the heuristic decoder permanently swaps the first bit that is most likely to be in error, is set to 7, thereby enabling correction of 2 bit errors. These numbers have an impact on the chances of finding the bit in error. Depending on which code block size is chosen and which channel conditions are present, the parameters are more or less suitable. Large code blocks in combination with a bad channel imply more uncertain bits in the bit-sequence.

To be able to observe possible gains in image quality with the modified decoders, reference images with two baseline decoders are also decoded. The first baseline decoder is configured to utilize the error-resilient mechanisms in the code-stream and conceal bit errors while the second one is configured not to utilize the error-resilient mechanisms in the code-stream. To compare the quality between the original image and the images decoded by the modified and the baseline decoders, the peak-signal-to-noise-ratio (PSNR) is calculated. To achieve valid statistical results, every image

[4] Modulation software provided by the IT++ library v3.6.6 (itpp.sourceforge.net).

[5] In IT++ soft information values are represented by the data type double, hence 64 bits per data bit.

[6] Based on the Kakadu Software v2.2 (www.kakadusoftware.com).

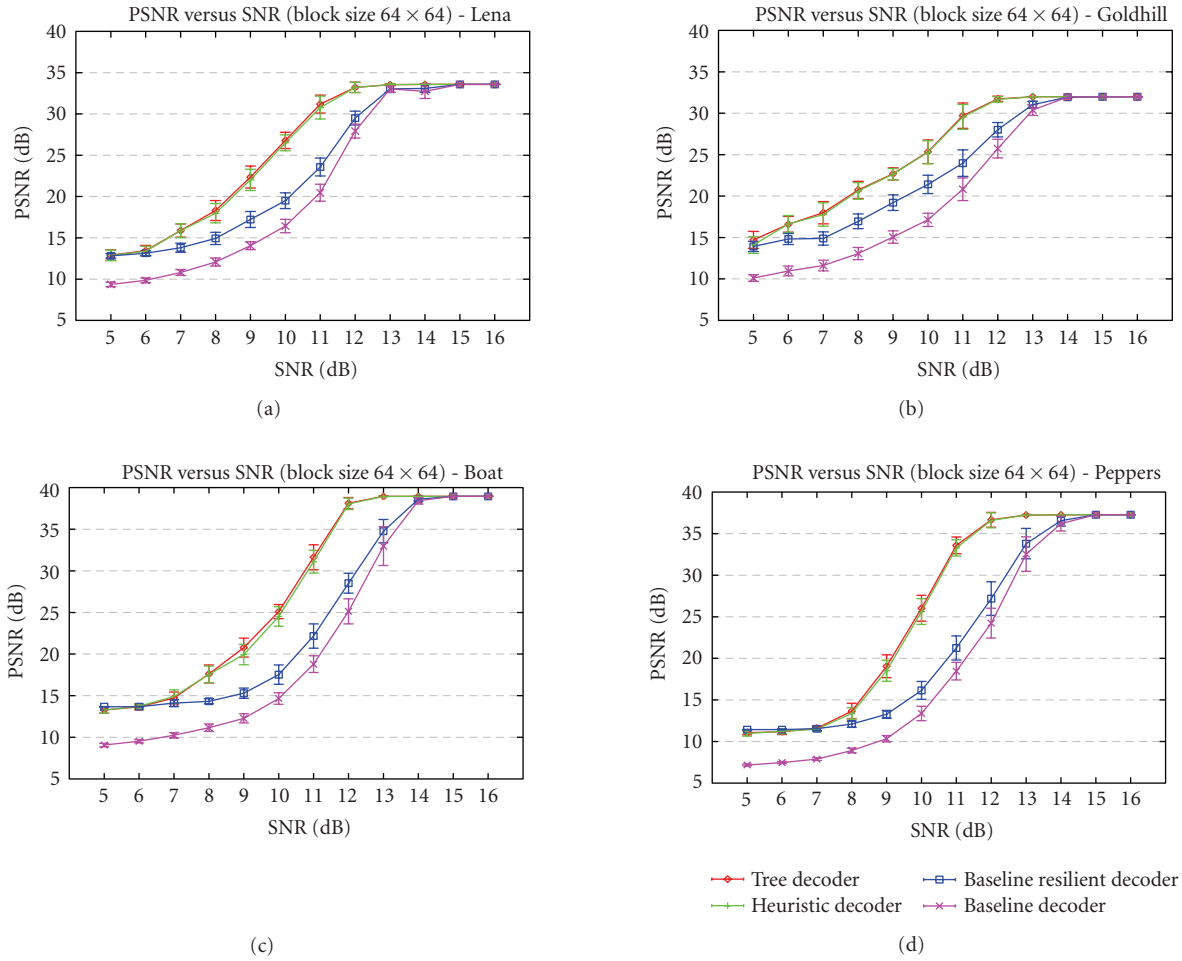


Figure 4: Comparison between decoders.

is transmitted over 30 different channels for every channel SNR.

5. EXPERIMENTAL RESULTS

Below follows a discussion of the experimental results. The graphs show image quality, measured with the PSNR metric, on the y-axis versus different choices of channel SNRs on the x-axis. The results are primarily presented for the code block size 64×64. The mean values and confidence intervals of 95 percent are used to present the results.

5.1. *Tree versus heuristic decoder*

The first experiment compares the tree and the heuristic decoder. The maximum number of iterations used is set to 10. The results are depicted in Figure 4 with the results for the two baseline decoders included as reference.

The most important observation made in Figure 4 is that the modified decoders outperform the baseline decoders for a range of channel SNRs. The highest gain observed (in PSNR) between the heuristic and the baseline error-resilient decoder is over 12 dB (see Figure 4 peppers image at SNR 11 dB). Another important observation is that the simple heuristic decoder performs well in comparison with the tree decoder. Only very small differences are observed regardless of the channel SNR. Even if the modified decoders give similar results, we still have to point out that the tree decoder performs somewhat better in general than the heuristic decoder. All the decoders behave in a similar manner, independently of image motif.

Taking a detailed look at the results in Figure 4, we find the following. All the decoders, especially the baseline decoder, perform poorly at SNRs 5-6 dB. The performance of the modified decoders and the baseline error-resilient decoder is very similar. In some cases, depending highly on image motif and image size, the baseline error-resilient decoder even performs better than the modified decoders. The modified decoders have a difficult time generating correct bit-sequences since they only consider a small amount of uncertain bits. Still, the advantage of utilizing soft information starts to show already at low SNRs. Depending on the image, this could occur as early as 6 dB. Observe, however, that for SNRs 5–10 dB the resulting subjective image quality is very poor for all the decoders involved, especially for the baseline decoder. Under these bad channel conditions, there are

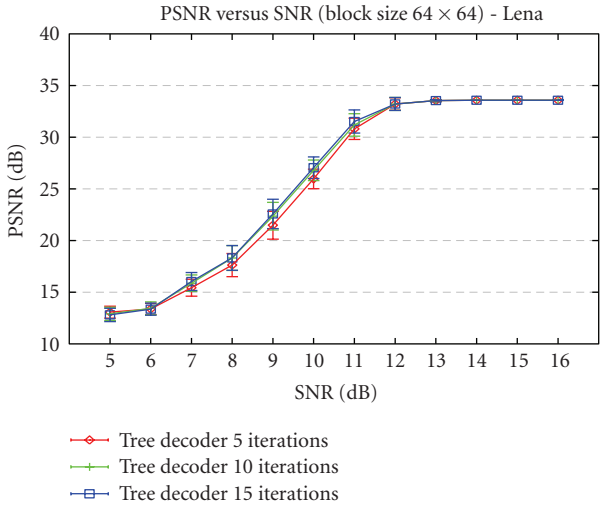


Figure 5: Different number of iterations (tree decoder).

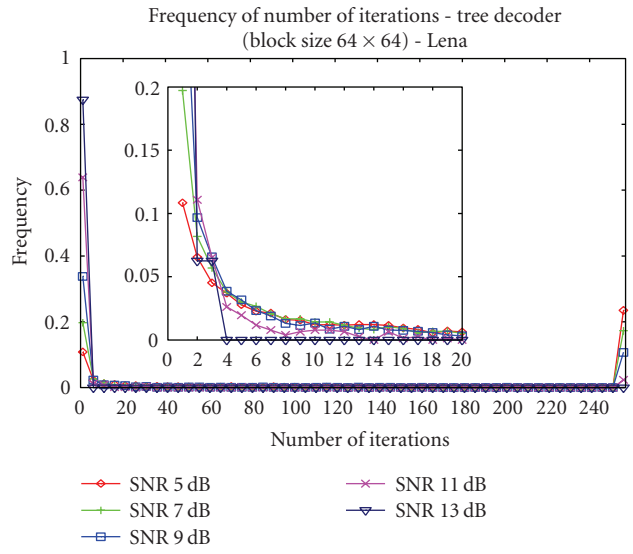


Figure 6: Frequency of number of iterations (tree decoder).

too many bit errors and the received images are often useless from a visual perspective. An image with a PSNR value of 30 dB usually gives a good visual experience (see [13]). For SNRs 8–12 dB, the positive effect of utilizing soft information is significant and high image quality gains can be achieved. Here the modified decoders outperform the others. The tree decoder performs somewhat better than the heuristic decoder for SNRs 8–11 dB. However, only gains less than 1 dB in mean values are observed with the tree decoder compared to the heuristic decoder. Furthermore, no statistically significant difference is established between the modified decoders. After SNR 12 dB, there is no obvious quality difference, based both on objective and subjective judgments, between the images decoded from the modified decoders. Hence it is sufficient to use the simpler and less computationally complex heuristic decoder independently of the channel SNR to be able to increase the image quality. For SNR 12 dB and below, it is evident that the baseline decoder performs very poorly compared to the other decoders. Hence taking no resort such as error concealment will lead to a very low image quality. At 12 dB, it is possible to decode an error free image by utilizing soft information. First at 14 dB it is possible to receive an almost error free image independently of decoder choice. No significant difference is present between the decoders at 15-16 dB due to an error free transmission.

Sample images with the Boat motif for SNR 11 dB are depicted in Figures 9(b), 9(d)–9(f). The image quality is very satisfying for the images decoded with the modified decoders while the baseline decoders perform poorly. High gains are also reached with the modified decoders when we investigate the resulting PSNR values.

5.2. *Impact of number of iterations*

Three additional experiments are conducted to investigate the impact of the number of iterations. First, we examine three different maximum numbers of iterations used by the tree decoder (5, 10, and 15 iterations). The results for the Lena image are depicted in Figure 5. The overall observation is that more iterations will only lead to minor image quality gains. The experimental results show that going from 5 to 10 iterations seems to give higher gains than going from 10 to 15 iterations. Hence a limited number of iterations appear to obtain most of the image quality gain. Similar to the results presented in the previous subsection all the different numbers of iterations perform poorly at SNRs 5-6 dB. For SNRs 7–11 dB the highest gain observed is with 15 iterations. However, 10 iterations are sufficient to obtain most of the gain. The image quality gain that 15 iterations introduce is only marginal compared to 10 iterations (less than 1 dB in mean values). The highest PSNR gain observed for 15 iterations over 5 iterations is approximately 2 dB in mean values. When the channel gets better (12–14 dB) no or only minor gains are observed with 15 iterations and for higher SNRs no difference between the different number of iterations is observed due to an error free channel. No statistically significant difference is established between the different numbers of iterations and from a computational overhead perspective it is not motivated to use too many iterations with the tree decoder. Figures 9(a)–9(c) depict sample images with the Boat motif at an SNR of 11 dB. The images show very good quality and it is difficult to detect visual differences between the different numbers of iterations.

To further investigate the impact of number of iterations the tree decoder is allowed to use a maximum of 255 iterations.[7] Each time the algorithm is invoked we record the number of iterations that is needed to correct the bit-sequence. Figure 6 depicts, for different channel SNRs, the

[7] The use of 255 iterations does not necessarily imply an optimal algorithm due to the current implementation of the tree decoder. Only a maximum of 16 bits is under consideration in this decoder.

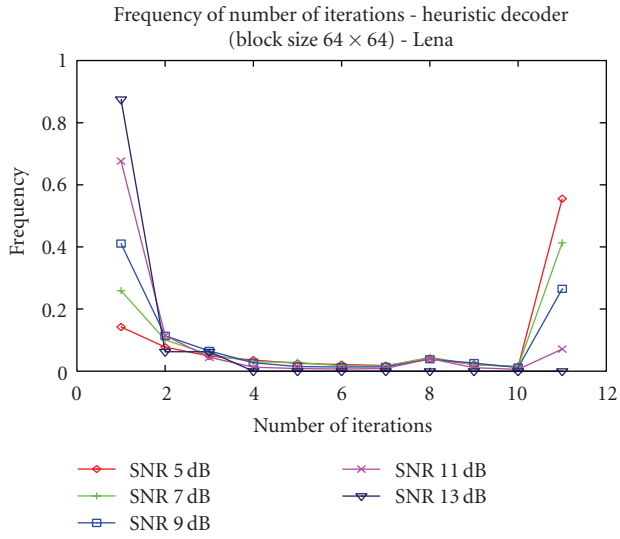


FIGURE 7: Frequency of number of iterations (heuristic decoder).

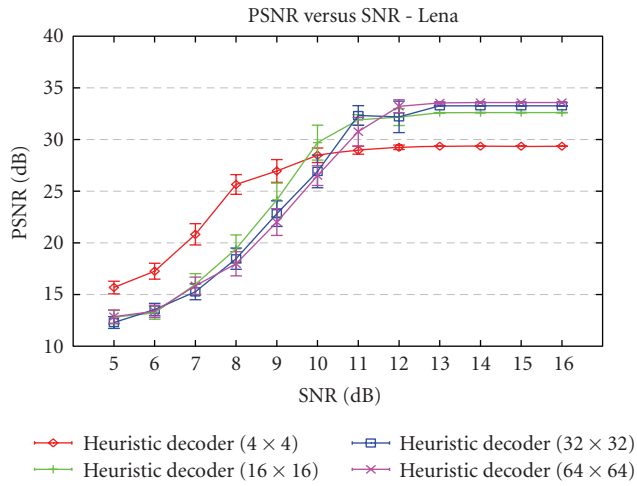


FIGURE 8: Implications of different code blocks (heuristic decoder).

frequency of the different number of iterations used to correct a bit-sequence relative to the total number of algorithm invocations.[8] It can be concluded that the decoder typically needs very few iterations to correct an erroneous bit-sequence, especially for high channel SNRs. Although not displayed by the graph, no or very low quality gains are observed when 255 iterations are allowed, compared to the results presented above. Allowing up to 10 iterations for SNR 11 dB will resolve 90 percent of the detected errors. This supports our observation above that a limited number of iterations is sufficient to obtain most of the image quality gain. Detailed analysis of the tree decoder's output also reveals that the error-resilient mechanisms in JPEG2000 are not able to detect all bit errors. This can unfortunately result in successive logical errors in the arithmetic decoder's internal states, errors that cannot be corrected with the two proposed algorithms. Thus, the conclusion made from this experiment is that a low number of iterations is sufficient to be able to increase image quality given the robustness the present error-resilient mechanisms can offer.

The last experiment investigates the number of iterations for the heuristic decoder. In this experiment, we again log the number of iterations needed to correct an error. Figure 7 depicts, for different channel SNRs, the frequency of the different number of iterations relative to the total number of algorithm invocations.[9] It can be concluded that the decoder needs only few iterations to resolve an error for SNRs 11 and 13 dB. Similar to the previous experiment for SNR 11 dB, the heuristic decoder resolves 90 percent of the detected errors. For low SNRs the number of iterations needed to correct an error increases and the heuristic decoder is not able to correct many errors.

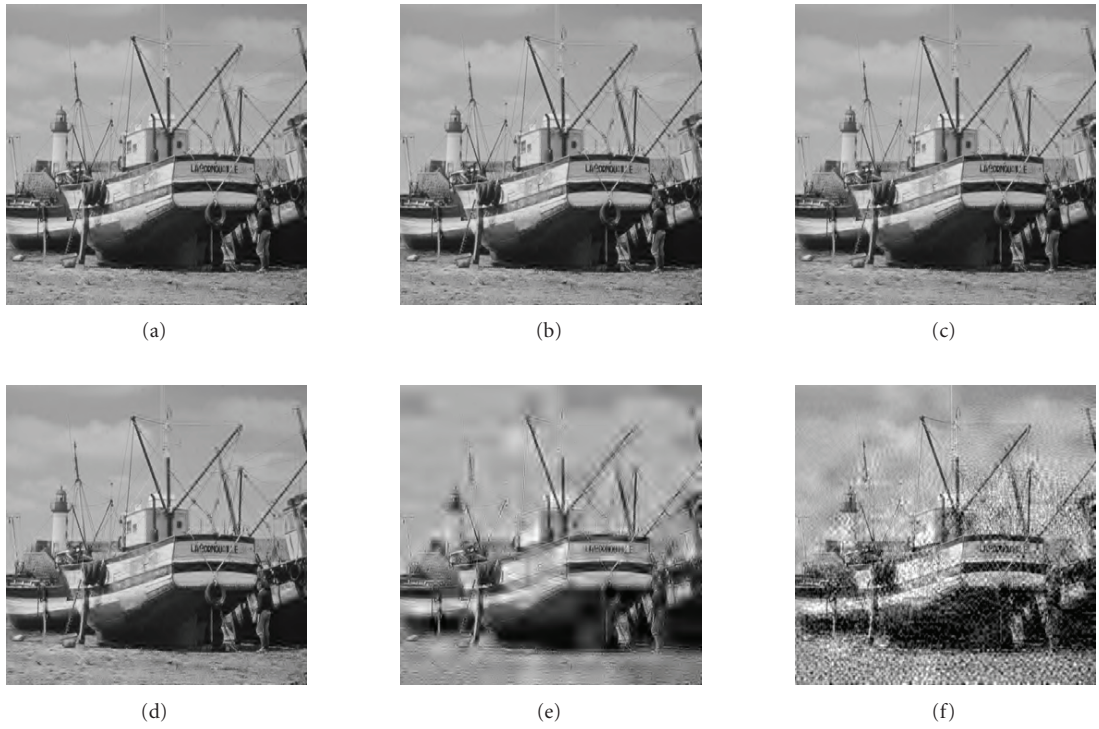
[8] A value of 256 for the number of iterations indicates that the error could not be corrected by the algorithm.

[9] The number 11 on the x-axis in Figure 7 corresponds to the case where the decoder is unable to correct the error.

5.3. *Impact of code block size*

The code block size plays an important role in the combat against errors and has a great impact on how efficiently the proposed algorithms can find new error free bit-sequences. The results in Sections 5.1 and 5.2 are only presented for 64×64 large code blocks; thus a discussion of how the code block size affects the outcome of the image quality is conducted below. A discussion of the impact of the code block size for the heuristic decoder is also given in [21]. Figure 8 depicts the influence that different code block sizes have on the image quality for the Lena motif when decoding is done by the heuristic decoder.

Generally when using small code block sizes at low SNRs, we are able to receive images with higher quality compared to larger code blocks because of the bit error enclosing feature of code blocks. The benefit of using small code blocks is clear for all the evaluated error-resilient decoders because they all have less data to process and concealing a bit error does not lead to a large data loss. The pursuit of an error free bit-sequence for the modified decoders is simplified because of the smaller data quantities. For the heuristic decoder, this conclusion is also clear from Figure 8 at an SNR of 5–9 dB. When increasing the code block size at low SNRs all the decoders perform poorly because bit errors in larger code blocks result in more data that are unusable. It will also be harder to pin point potential bit errors with the heuristic and the tree decoders. This is due to the fact that the modified decoders always consider a limited number of uncertain bits and use only a limited number of iterations. When the channel conditions are better (i.e., $\geq$ 11 dB) it is better to use large code blocks. Compared to small code blocks, large code blocks result in less auxiliary data and, for a fixed image file size, more pure image data are available as a whole. The great overhead of auxiliary information for small code blocks in the file structure reduces the image quality substantially. Thus in an almost error free environment we are better off



(a) (b) (c)

(d) (e) (f)

Figure 9: Sample images at SNR 11 dB with code block size 64×64 (Boat motif). (a) Tree decoder 15 iterations PSNR 31.4461 dB, (b) tree decoder 10 iterations PSNR 31.4318 dB, (c) tree decoder 5 iterations PSNR 30.3724 dB, (d) heuristic decoder PSNR 30.6885 dB, (e) baseline error-resilient decoder PSNR 23.301 dB, (f) baseline decoder PSNR 20.0612 dB.

using larger code blocks. This observation is again evident for the heuristic decoder from Figure 8, at SNRs between 12 and 16 dB. It is also possible to receive images with large code blocks that contain errors but have higher PSNR values than error free images with small code blocks (see Figure 8 for SNRs 11-12 dB).

6. CONCLUDING REMARKS

The utilization of cross-layer information in the form of soft information from the physical layer is considered in this paper. We have implemented two soft input iterative JPEG2000 image decoders that take advantage of the soft information to improve image quality when combating channel bit errors in a wireless environment. We can state that decoding with soft information performs better than with hard decided bits in terms of gains in image quality and a more versatile decoding process. The gain in channel SNR could be as high as 2 dB. Further, the results of our experiments presented here indicate that a decoder with simple heuristic rules and a limited number of iterations performs well. Both high PSNRs and visual image quality gains are present, especially when the channel SNR lies between 8 and 12 dB. Only marginal additional gains are made with a more complex decoder and when more iterations are allowed.

With our experimental set-up, an acceptable subjective image quality is achieved with soft information approximately at channel SNR 11 dB. When the channel SNR is between 5 and 10 dB, the resulting subjective image quality is not acceptable; thus soft information will not help in the decoding process. When the channel SNR is very high, no image quality gains are observed with soft information due to an error free transmission. Different JPEG2000 code block sizes also have an evident impact on image quality in the occurrence of bit errors and in the context of soft information. At channel SNR 11 dB and above the recommendation is to use larger code blocks.

Comparing a JSCD system to ours, our work differs in the following ways. Firstly, we do not have any demand for applying channel coding in the physical or link layer on the data that soft information is applied on, although it can be included at the sender. Secondly, our work does not involve an iterative process between the channel decoder and the source decoder. This iterative process demands that information can be exchanged between the decoders thus between network protocols. In our proposed decoder, there is an iterative process, but it is solely restricted to the internals of the source decoder. Thirdly, our technique is simplified and not an optimal technique for improving image quality. Work done in, for example, [5] is a novel example of a JSCD system which gives higher gains in image quality compared to our approach but at the expense of higher complexity.

Utilizing soft information from the physical layer in the application layer requires modifications to the intermediate communication layers. The modifications involve firstly the propagation of soft information from the physical layer to the application layer, and secondly, intermediate layers must allow erroneous payload data due to the application's

enhanced error resilience. Potential solutions for these demands can be found in the literature (e.g., [23–27] for propagation of soft information and [8–10, 28] for allowing erroneous payload data). In the future, we wish to investigate possible performance improvements with soft information in a system-wide perspective. Combining the results presented here, for example, with a bit error transparent protocol such as UDP-Lite [8] or TCP-L [9] will make it possible to achieve numerical results about the system performance.

ACKNOWLEDGMENTS

This work was supported in part by the Swedish Governmental Agency for Innovation Systems, VINNOVA, and the Swedish Foundation for Strategic Research.

REFERENCES

[1] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: a review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, 1998.

[2] I. Moccagatta, S. Soudagar, J. Liang, and H. Chen, "Error-resilient coding in JPEG-2000 and MPEG-4," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 899–914, 2000.

[3] C. Guillemot and P. Siohan, "Joint source-channel decoding of variable-length codes with soft information: a survey," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 6, pp. 906–927, 2005.

[4] C. Guillemot and P. Christ, "Joint source-channel coding as an element of a QoS framework for '4G' wireless multimedia," *Computer Communications*, vol. 27, no. 8, pp. 762–779, 2004.

[5] W. Xiang, S. S. Pietrobon, and S. A. Barbulescu, "Iterative source-channel decoding for robust image transmission," in *Proceedings 4th Australian Communications Theory Workshop (AusCTW '03)*, pp. 61–65, Melbourne, Australia, February 2003.

[6] "JPEG2000 Part 1—JPEG2000 Final Committee Draft Version 1.0, ISO/IEC 15444-1," March 2000.

[7] A. Brunstrom and T. Ottosson, "The introduction of soft information in IP-based wireless networks," in *Proceedings of Nordic Radio Symposium (NRS '01)*, Nynäshamn, Sweden, April 2001.

[8] L.-Å. Larzon, M. Degermark, S. Pink, L.-E. Jonsson, and G. Fairhurst, "The lightweight user datagram protocol (UDPLite)," Network Working Group, RFC 3828, July 2004.

[9] S. Alfredsson and A. Brunstrom, "TCP-L: allowing bit errors in wireless TCP," in *Proceedings of IST Mobile and Wireless Communications Summit*, Aveiro, Portugal, July 2003.

[10] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," Internet Engineering Task Force, Internet draft expires, March 2006.

[11] M. Welzl, "Passing corrupt data across network layers: an overview of recent developments and issues," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 2, pp. 242–247, 2005.

[12] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[13] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Boston, Mass, USA, 2002.

[14] C. Berrou, "The ten-year-old turbo codes are entering into service," *IEEE Communications Magazine*, vol. 41, no. 8, pp. 110–116, 2003.

[15] S. ten Brink, J. Speidel, and R.-H. Yan, "Iterative demapping and decoding for multilevel modulation," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '98)*, vol. 1, pp. 579–584, Sydney, Australia, November 1998.

[16] J. Hagenauer, "The turbo principle: tutorial introduction and state of the art," in *Proceedings of the Symposium on Turbo-Codes*, pp. 1–11, Brest, France, September 1997.

[17] M. Skoglund and T. Ottosson, "Soft multiuser decoding for vector quantization over a CDMA channel," *IEEE Transactions on Communications*, vol. 46, no. 3, pp. 327–337, 1998.

[18] M. Grangetto, E. Magli, and G. Olmo, "Error sensitivity data structures and retransmission strategies for robust JPEG 2000 wireless imaging," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 872–882, 2003.

[19] A. Natu and D. Taubman, "Unequal protection of JPEG2000 code-streams in wireless channels," in *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM '02)*, vol. 1, pp. 534–538, Taipei, Taiwan, November 2002.

[20] D. Nicholson, C. Lamy-Bergot, X. Naturel, and C. Poulliat, "JPEG 2000 backward compatible error protection with Reed-Solomon codes," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 855–860, 2003.

[21] H. Persson, A. Brunstrom, and T. Ottosson, "Utilizing soft information at the application layer: quality enhanced JPEG2000 decoding," in *Proceedings of Radio Vetenskap och Kommunikation (RVK '05)*, Linköping, Sweden, June 2005.

[22] H. Persson, A. Brunstrom, and T. Ottosson, "Utilizing soft information in image decoding," in *Proceedings of the 14th IEEE Personal, Indoor and Mobile Radio Communications (PIMRC '03)*, vol. 3, pp. 2678–2682, Beijing, China, September 2003.

[23] S. Mérigeault and C. Lamy, "Concepts for exchanging extra information between protocol layers transparently for the standard protocol stack," in *Proceedings of the 10th International Conference on Telecommunications (ICT '03)*, vol. 2, pp. 981–985, Tahiti, Papeete, French Polynesia, February-March 2003.

[24] L.-Å. Larzon, U. Bodin, and O. Schelén, "Hints and notifications," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC '02)*, vol. 2, pp. 635–641, Orlando, Fla, USA, March 2002.

[25] H. Zheng and J. Boyce, "An improved UDP protocol for video transmission over Internet-to-wireless networks," *IEEE Transactions on Multimedia*, vol. 3, no. 3, pp. 356–365, 2001.

[26] C. Lamy-Bergot and P. Vila, "Multiplex header compression for transparent cross-layer design," in *Proceedings of the IEEE International Conference on Networking (ICN '04)*, pp. 1084–1089, Guadeloupe, French Carribean, March 2004.

[27] Q. Wang and M. A. Abu-Rgheff, "Cross-layer signalling for next-generation wireless systems," in *Proceedings of IEEE Wireless Communications and Networking (WCNC '03)*, vol. 2, pp. 1084–1089, New Orleans, La, USA, March 2003.

[28] G. Fairhurst and L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)," Network Working Group, RFC 3366, August 2002.