HIGHLY SCALABLE IMAGE COMPRESSION BASED ON SPIHT FOR NETWORK APPLICATIONS

Habibollah Danyali and Alfred Mertins

School of Electrical, Computer and Telecommunications Engineering
University of Wollongong, Wollongong, NSW 2522, Australia
Email: {hd04, mertins}@uow.edu.au

ABSTRACT

In this paper we propose a highly scalable image compression scheme based on the Set Partitioning in Hierarchical Trees (SPIHT) algorithm. Our algorithm, called Highly Scalable SPIHT (HS-SPIHT), supports spatial and SNR scalability and provides a bitstream that can be easily adapted (reordered) to given bandwidth and resolution requirements by a simple transcoder (parser). The HS-SPIHT algorithm adds the spatial scalability feature without sacrificing the SNR embeddedness property as found in the original SPIHT bitstream. HS-SPIHT finds applications in progressive web browsing, flexible image storage and retrieval and image transmission over heterogeneous networks.

1. INTRODUCTION

Traditional image coding systems have only focused on efficient compression of image data. The main objective of such systems is optimizing image quality at given bit rate. Due to the explosive growth of the Internet and networking technology, nowadays a huge number of users with different capabilities of processing and network access bandwidth can transfer and access data easily. For transmission of visual data on such a heterogenous network, efficient compression itself is not sufficient. There is an increasing demand for scalability to optimally service each user according to his bandwidth and computing capabilities. A scalable image coder generates a bitstream which consists of a set of embedded parts that offer increasingly better signal-to-noise ratio (SNR) or/and greater spatial resolution. Different parts of this bitstream can be selected and decoded by a scalable decoder to meet certain requirements. In the case of an entirely scalable bitstream, different types of decoders with different complexity and access bandwidth can coexist.

Over the past decade wavelet-based image compression schemes have become increasingly important and gained widespread acceptance. Because of their inherent multiresolution signal representation, wavelet-based coding schemes have the potential to support both SNR and spatial scalability. For efficient coding of wavelet coefficients, Shapiro [1] introduced the Embedded Zerotree Wavelet (EZW) coding scheme based on the idea of grouping spatially related coefficients at different scales to trees and efficiently predicting zero coefficients across scales. An improved scheme, called Set Partitioning in Hierarchical Trees (SPIHT), was developed by Said and Pearlman [2]. This coder uses the spatial orientation trees shown in Fig. 1 and partitions them as needed to sort wavelet coefficients according to magnitude. Further improvements of SPIHT have been published in [3–8]. Although the SPIHT coder is fully SNR scalable with excellent compression properties, it does not explicitly support spatial scalability and does not provide a bitstream that can be parsed easily according to the type of scalability desired by the decoder.

An improved version of the EZW algorithm that uses better context modeling for arithmetic coding, an improved symbol set for zerotree encoding, and proper syntax and markers for the compressed bitstream to allow extracting various qualities and resolutions was reported in [9]. However the decoder needs some additional side information to decode the bitstream. Tham et al. [10] introduced a new zerotree structure called tri-zerotree and used a layered coding strategy with the concept of embedded resolution block coding to achieve some degree of scalability for video coding. A spatially scalable video coding scheme based on SPIHT was reported by Kim et al. in [5]. Their coder produces a two-layer bitstream; the first layer is used for low resolution, and the second one adds the extra information required for high resolution. Although the first layer of this method is rate scalable, the bitstream is not fully embedded for high resolution. Moreover, it is not possible to easily transcode the encoded bitstream to arbitrary spatial resolutions and SNR's.

In this paper, a fully scalable image coding scheme based on the SPIHT algorithm is presented. We modify the SPIHT algorithm to support both spatial and SNR scalability features. The encoder creates a bitstream that can be easily parsed to achieve different levels of resolution or/and quality requested by the decoder. A distinct feature of the presented coder is that the reordered bitstreams for different spatial resolutions, which are obtained after parsing the main bitstream, are fully embedded (SNR scalable) and can be truncated at any point to obtain the best reconstructed image at the desired spatial resolution and bit rate. In other words, our modified SPIHT algorithm provides spatial scalability without sacrificing SNR scalability in any way.

2. HIGHLY SCALABLE SPIHT (HS-SPIHT)

In general, an N level wavelet decomposition allows at most $N+1$ levels of spatial resolution. To distinguish between different resolution levels, we denote the lowest spatial resolution level as level $N+1$. The full image then becomes resolution level 1. The three subbands that need to be added to increase the resolution from level k to level $k-1$ are referred to as level $k-1$ resolution subbands (see Fig. 1). An algorithm that provides full spatial scalability would encode the different resolution levels separately, allowing a transcoder or the decoder to directly access the data needed to reconstruct with a desired spatial resolution. The original SPIHT algorithm, however, encodes the entire wavelet tree in a bitplane by bitplane manner and produces a bitstream that contains the information about the different spatial resolutions in no particular order.

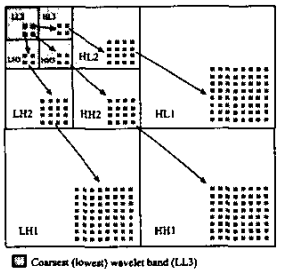

Fig. 1. Orientation of trees across wavelet bands.

The HS-SPIHT algorithm proposed in this paper solves the spatial scalability problem through the introduction of a resolution-dependent sorting pass that uses (compared to SPIHT) one additional list, called the list of delayed insignificant sets (LDIS). The HS-SPIHT coder first encodes all bitplanes for a given (low) resolution level and then moves to the next higher resolution level. Sets encountered during the sorting pass that lie outside the actually considered spatial resolution are temporarily stored in the LDIS. They are moved back from the LDIS into the LIS when they are required for encoding the next higher resolution. According to the magnitude of the coefficients in the wavelet pyramid, coding of higher resolution bands usually starts from lower bitplanes. Therefore, during the encoding process of resolution level k, the encoder keeps the number of coefficients that went to the LDIS for each quantization level. After finishing the encoding process for all bitplanes of resolution level k, the encoder knows which entries in the LDIS belong to which bitplane. To encode the additional three subbands for resolution level $k-1$, it moves the related entries of the LDIS that belong to the actual bitplane to the LIS and carries out the sorting of LIS with the same procedure as before. Altogether, the total number of bits belonging to a particular bitplane is the same for SPIHT and HS-SPIHT, but HS-SPIHT distributes them differently among the different spatial resolution levels.

The definitions for terms required by HS-SPIHT are the same as for SPIHT and are therefore not listed here. The reader is referred to [2]. In the following we list the entire HS-SPIHT coding algorithm:

1. **Initialization**
   Set $n = \lfloor \max_{(i,j)}\{|c_{i,j}|\} \rfloor$ and output it. Set the LSP and LDIS as empty lists. Put the coordinates of all roots in H into the LIP. Put the roots in H which have descendants also into the LIS as type A entries. Set $k = k_{max}$ where k_{max} is the maximum level of spatial scalability supported by the encoder ($1 \leq k \leq N+1$).
2. **Resolution-Dependent Sorting Pass**
   2.1 for each entry (i,j) in the LIP do:
   2.1.1. output $S_n(i,j)$;
   2.1.2. if $S_n(i,j) = 1$ then move (i,j) to the LSP and output the sign of $c_{i,j}$;
   2.2. for each entry (i,j) in the LIS do:
   2.2.1 if the entry is of type A then
   - if all coordinates in the $D(i,j)$ are located outside of the spatial resolution level k then move (i,j) to the LDIS as type A, else:
     * output $S_n(D(i,j))$;
     * if $S_n(D(i,j)) = 1$ then for each $(p,q) \in O(i,j)$ do:
       - output $S_n(p,q)$;
       - if $S_n(p,q) = 1$ then add (p,q) to the LSP and output the sign of $c_{p,q}$;
         else add (p,q) to the end of the LIP;
     * if $L(i,j) \neq \emptyset$ then move (i,j) to the end of the LIS as an entry of type B and go to step 2.2.2; else, remove entry (i,j) from the LIS;

   2.2.2. if the entry is of type B then
   - if all coordinates in the $L(i,j)$ are located outside of the spatial resolution level k then move (i,j) to the LDIS as type B; else:
     * output $S_n(L(i,j))$;
     * if $S_n(L(i,j)) = 1$ then
       - add each $(p,q) \in O(i,j)$ to the end of the LIS as an entry of type A;
       - remove (i,j) from the LIS.
3. **Refinement Pass**
   for each entry (i,j) in the LSP, except those included in the last sorting pass (i.e. the ones with the same n), output the n^{th} most significant bit of $|c_{i,j}|$;
4. **Quantization-Step Update**
   - decrement n by 1
   - if n is greater or equal to the minimum bitplane then
     * if $k = k_{max}$ then go to step 2 else go to step 5.1.
5. **Resolution Scale and Lists Update**
   - if $k > 1$ then:
     * decrement k by 1
     * set the LIS, LIP and LSP as empty lists
     * set n with the maximum quantization level related to the first entry that was moved from the LIS to the LDIS during resolution-dependent sorting pass for resolution level $k+1$.

   5.1. move back all entries in the LDIS which were moved to the LDIS during quantization level n of the sorting pass for resolution level $k+1$, to the LIS and go to step 2.
   - else: end of coding

To support bitstream parsing by an image server/transcoder, some markers are required to be put into the bitstream to identify the parts of the bitstream that belong to the different spatial resolution levels and bitplanes. This additional information does not need to be sent to the decoder.

3. BITSTREAM FORMATION AND PARSING

The bitstream generated by the encoder can be sorted in different ways, however it can exist in one specific order at a time only. Fig. 2 shows the bitstream structure generated by the encoder. The bitstream is divided into different parts according to the different spatial resolution levels. Inside each resolution level the bits that belong to different bitplanes are separable. A header at the beginning of the bitstream identifies the number of spatial resolution levels supported by the encoder, as well as information such as the

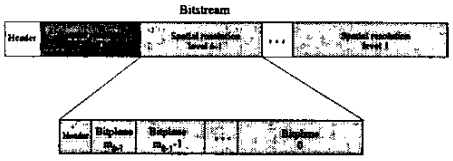

Fig. 2. Scalable bitstream which is made up of different parts according to spatial resolution and quality.

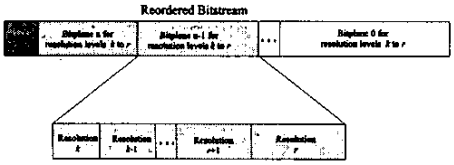

Fig. 3. Reordered bitstream for spatial resolution level r

image dimension, number of wavelet decomposition levels, and the maximum quantization level. At the beginning of each resolution level there is an additional header that provides the information required to identify each bitplane. In Fig. 2, m_{k-1} is the first (highest) bitplane used for coding of the level $k-1$ spatial resolution subbands.

After encoding the original image at high bit rate, the bitstream is stored on an image server. Different users with different requirements send their request to the server and the server or a transcoder within the network provides them with a properly tailored bitstream that is easily generated by selecting the related parts of the original bitstream and ordering them in such a way that the user request is fulfilled. To carry out the parsing process, the image server or transcoder does not need to decode any parts of the bitstream.

There are two principal ways of manipulating the encoded bitstream by the image server or transcoder to meet the user requests:

1. The bitstream gets truncated to the required resolution level, and in addition, some of the lower bitplanes may be removed. This yields a resolution embedded, but not a fully SNR embedded bitstream. To enable decoding, the headers introduced by the encoder need to be kept.
2. The bitstream is reordered bitplane by bitplane for the requested resolution level. This form of reordering results in a fully SNR embedded bitstream for the desired resolution. Fig. 3 shows an example of a reordered bitstream for spatial resolution level r. All header information for identifying the individual bitplanes is only used by the image parser and does not need to be sent to the decoder.

The decoders required for the two parsing methods are different. For the first method the decoder directly follows the encoder with the output command replaced by an input command, similar to the original SPIHT algorithm. The decoder for the second method additionally needs to keep track of the various lists (LIS, LIP, LSP) for all resolution levels greater or equal to the required one. It can recover all information for updating the lists during sorting pass of each quantization level (bitplane) at each spatial resolution level. The only additional information required by the decoder is the maximum number of spatial scalability levels (k_{max}) supported by the encoder. Note that it is also possible to modify the encoder to directly produce the bitstream for the second decoding method.

4. EXPERIMENTAL RESULTS

In this section we present some numerical results for the HS-SPIHT algorithm. All results were obtained with 8 bit per pixel (bpp) monochrome images of size 512 x 512 pixels. We first applied five levels of wavelet decomposition with the 9/7-tap filters of [11] and symmetric extension at the image boundaries. The HS-SPIHT encoder was set to produce a bitstream that supports the maximum number of spatial scalability levels.

After encoding, the HS-SPIHT bitstream was fed into a transcoder to produce progressive (by quality) bitstreams for different spatial resolutions. This is the second type of transcoding described in Section 3. The bitstreams were decoded with different rates and the fidelity was measured by the peak signal-to-noise ratio defined as

$$\text{PSNR} = 10 \log_{10} \frac{\text{PEAK}^2}{\text{MSE}} \, dB$$

where MSE is the mean squared error between the original and the reconstructed image, and PEAK is the maximum possible magnitude for a pixel inside the image. The PEAK value is 255 for an 8 bpp original image (level 1) and $255 \times 2^{k-1}$ for resolution level k. The bit rates for all levels were calculated according to the number of pixels in the original full size image.

Fig. 4 compares rate-distortion results for HS-SPIHT and original (level 1) test images against SPIHT. Prior to the decoding process, the HS-SPIHT compressed bitstreams were transcoded in order to produce the proper bitstreams for spatial resolution level 1 decoding. The results in Fig. 4 clearly show that the HS-SPIHT completely keeps the progressiveness (by SNR) property of the SPIHT algorithm. The very small deviation between HS-SPIHT and SPIHT is due to a different order of coefficients within the bitstreams. All the results are obtained without extra arithmetic coding of the output bits. As shown in [2], an improved coding performance (about 0.3-0.6 dB) for SPIHT and consequently for HS-SPIHT can be achieved by further compressing the binary bitstreams with an arithmetic coder.

Fig. 5 shows the results obtained for decoding the resolution levels 2 and 3 by the SPIHT and HS-SPIHT coders. The HS-SPIHT decoder obtained the proper bitstreams tailored by the transcoder for each resolution level while for the SPIHT case we first decoded the whole image at each bit rate and then compared the requested spatial resolutions of the reconstructed and original images. All bits in the transcoded HS-SPIHT bitstream for a particular resolution belong only to that resolution, while in the SPIHT bitstream, the bits that belong to the different resolution levels are interwoven. Therefore, as expected, the performance of HS-SPIHT is much better than for SPIHT. As the resolution level increases, the difference between HS-SPIHT and SPIHT becomes more and more significant.

5. CONCLUSIONS

We have presented a highly scalable SPIHT algorithm that produces a bitstream which supports spatial scalability and can be used for multiresolution transcoding. This bitstream not only has spatial scalability features but also keeps the full SNR embeddedness property for any required resolution level after a simple reordering which can be done in a transcoder without decoding the bitstream. The embeddedness is so fine granular that almost

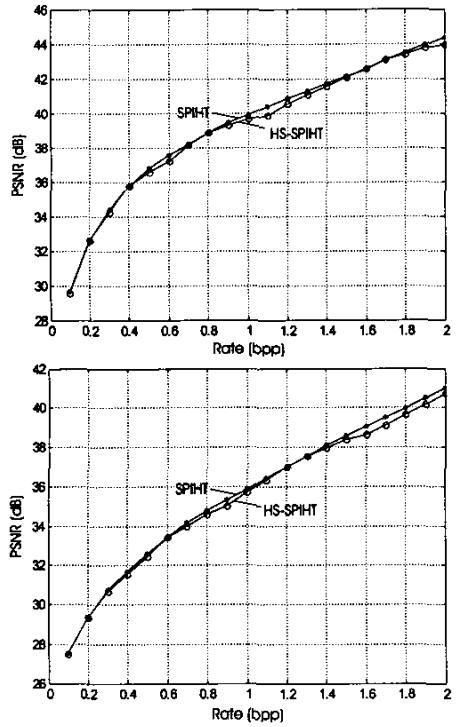

Fig. 4. Comparison of rate-distortion results for maximum spatial resolution (level 1) of 512 × 512 test images: top: Lena; bottom: Goldhill

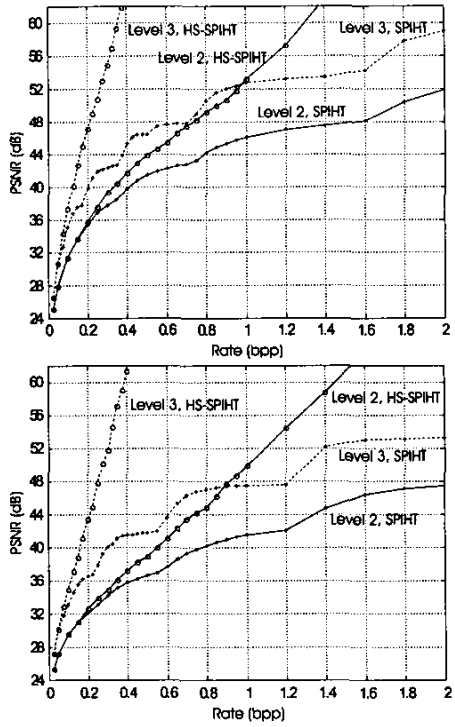

Fig. 5. Comparison of rate-distortion results for spatial resolution levels 2 (256 × 256) and level 3 (128 × 128) of 512 × 512 test images: top: Lena bottom: Goldhill

each additional bit improves the quality, and the bitstream can be stopped at any point to meet a bit budget during the coding or decoding process. The proposed multiresolution image codec is a good candidate for multimedia applications such as image storage and retrieval systems, progressive web browsing and multimedia information transmission, especially over heterogenous networks where a wide variety of users need to be differently serviced according to their network access and data processing capabilities.

6. REFERENCES

[1] J. M. Shapiro, "Embedded image coding using zerotree of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.

[2] A. Said and W. A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circ. and Syst. for Video Technology*, vol. 6, pp. 243–250, June 1996.

[3] B.-J.Kim and W. A. Pearlman, "An embedded video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)," in *proc. IEEE Data Compression Conf.*, Mar. 1997, pp. 251–260.

[4] J. Karlenkar and U. B. Desai, "SPIHT video coder," in *Proc. IEEE Region 10 International Conference on Glaobal Connectivity in Energy, Computer, Communication and Control, TENCON'98*, 1998, vol. 1, pp. 45–48.

[5] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Low bit-rate scalable video coding with 3-d set partitioning in hierarchical trees (3-D SPIHT)," *IEEE Trans. Circ. and Syst. for Video Technology*, vol. 10, no. 8, pp. 1374–1387, Dec. 2000.

[6] J. Zho and S. Lawson, "Improvements of the SPIHT for image coding by wavelet transform," in *Proc. IEE Seminar on Time-scale and Time-Frequency Analysis and Applications (Ref. No. 2000/019)*, 2000, pp. 24/1 –24/5.

[7] E. Khan and M. Ghanbari, "Very low bit rate video coding using virtual spiht," *IEE Electronics Letters*, vol. 37, no. 1, pp. 40–42, Jan. 2001.

[8] H. Cai and B. Zeng, "A new SPIHT algorithm based on variable sorting thresholds," in *Proc. IEEE Int. Symp. Circuits and Systems*, May 2001, vol. 5, pp. 231–234.

[9] J. Liang, "Highly scalable image coding for multimedia applicatioans," in *Proc. ACM Multimedia 97*, Nov. 1997, pp. 11–19.

[10] J. Y. Tham, S. Ranganath, and A. A. Kassim, "Highly scalable wavelet-based video codec for very low bit-rate environment," *IEEE J. Select. Areas Commun.*, vol. 16, no. 1, pp. 12–27, Jan. 1998.

[11] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 205–220, Apr. 1992.