Fully Spatial and SNR Scalable, SPIHT-Based Image Coding for Transmission Over Heterogenous Networks

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

This paper presents a fully scalable image coding scheme based on the Set Partitioning in Hierarchical Trees (SPIHT) algorithm. The proposed algorithm, called Fully Scalable SPIHT (FS-SPIHT), adds the spatial scalability feature to the SPIHT algorithm. It provides this new functionality without sacrificing other important features of the original SPIHT bitstream such as: compression efficiency, full embeddedness and rate scalability. The flexible output bitstream of the FS-SPIHT encoder which consists of a set of embedded parts related to different resolutions and quality levels can be easily adapted (reordered) to given bandwidth and resolution requirements by a simple parser without decoding the bitstream. FS-SPIHT is a very good candidate for image communication over heterogenous networks which requires high degree of scalability from image coding systems.

Keywords: *wavelet image coding, scalability, SPIHT, progressive transmission, multiresolution.*

1 Introduction

The main objective of traditional image coding 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 network access bandwidth and processing capabilities can easily exchange data. 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 the available 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. An example is the new JPEG2000 still image compression standard [1, 2]. Because of their inherent multiresolution signal representation, wavelet-based coding schemes have the potential to support both SNR and spatial scalability. Shapiro [3] pioneered embedded wavelet-based image coding by introducing 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. The scheme provides an output bitstream that consists of data units ordered by their importance and that can be truncated at any point without degradation of the coding efficiency. Many researchers have since worked on variations of the original zerotree method [4–10]. An important development of EZW, called Set Partitioning in Hierarchical Trees (SPIHT) algorithm by Said and Pearlman [7] is one of the best performing wavelet-based image compression algorithms. 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 [11–16]. Although almost all of the state-of-the-art zerotree-based image compression methods are SNR scalable and provide bit streams for progressive (by quality) image compression, they do not explicitly support spatial scalability and do not provide a bitstream which can be adapted easily according to the type of scalability desired by the decoder.

An improvement of the EZW algorithm called predictive EZW (PEZW) was reported in [8]. The PEZW improves the EZW through better context modelling for arithmetic coding and an improved symbol set for zerotree encoding. It also uses proper syntax and markers for the compressed bitstream to allow extracting bitstreams that represent various qualities and resolutions of the original image. However the decoder needs some additional side information to decode these bitstreams. Tham et al. [17] introduced a new zerotree structure called tri-zerotree and used a layered coding strategy with the concept of embedded resolution block coding to achieve a high degree of scalability for video coding. A spatially scalable video coding scheme based on SPIHT was reported by Kim et al. in [13]. 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 reorder the encoded bitstream to arbitrary spatial resolutions and SNR's. However, the ability to reorder the bitstream is an important requirement for access to images through heterogeneous networks with a large variation in bandwidth and user-device capabilities.

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.

The rest of this paper is organized as follow. The next section, Section 2, describes the FS-SPIHT algorithm. The bitstream formation and parsing are explained in Section 3. Section 4 shows some results on the rate-distortion performance for our codec and provides comparisons with the SPIHT coder. Finally some conclusions are presented in Section 5.

2 Fully Scalable SPIHT (FS-SPIHT)

In this section, we first give a brief description of the SPIHT algorithm, then explain our modification of SPIHT (FS-SPIHT) for fully supporting SNR and spatial scalabilities. The SPIHT algorithm consists of three stages: initialization, sorting and refinement. It sorts the wavelet coefficients in three ordered lists: the list of insignificant sets (LIS), the list of insignificant pixels (LIP), and the list of significant pixels (LSP). At the initialization stage the SPIHT algorithm first defines a start threshold due to the maximum value in the wavelet coefficients pyramid, then sets the LSP as an empty list and puts the coordinates of all coefficients in the coarsest level of the wavelet pyramid (i.e. the lowest frequency band; LL band) into the LIP and those which have descendants also into the LIS. Fig. 1 shows the parent-child relationships used within the wavelet tree. The pixels in the coarsest level of the pyramid are grouped into blocks of 2×2 adjacent pixels, and in each block one of them has no descendants. In the sorting pass, the algorithm first sorts the elements of the LIP and then the sets with roots in the LIS. For each pixel in the LIP it performs a significance test against the current threshold and outputs the test result to the output bitstream. All test results are encoded as either 0 or 1, depending on the test outcome, so that the SPIHT algorithm directly produces a binary bitstream. If a coefficient is significant, its sign is coded and then its coordinate is moved to the LSP. During the sorting pass of LIS, the SPIHT encoder carries out the significance test for each set in the LIS and outputs the significance information. If a set is significant, it is partitioned into its offspring and leaves. Sorting and partitioning are carried out until all significant coefficients have been found and stored in the LSP. After the sorting pass for all elements in the LIP and LIS, SPIHT does a refinement pass with the current threshold for all entries in the LSP, except those which have been moved to the LSP during the last sorting pass. Then the current threshold is divided by two and the sorting and refinement stages are continued until a predefined bit-budget is exhausted.

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. An algorithm that provides full spatial scalability would encode the different resolution levels separately, allowing a parser 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.

In [18] we modified SPIHT to support both spatial and SNR scalability by adding a new list to the SPIHT lists and modifying the SPIHT sorting pass. The FS-SPIHT algorithm proposed in this paper solves the spatial scalability problem through the introduction of multiple resolution-dependent lists and a resolution-dependent sorting pass. For each spatial resolution level we define a set of LIP, LSP and LIS lists, therefore we have LIP$_k$, LSP$_k$, and LIS$_k$ for

$k = k_{max}, k_{max} - 1, \ldots, 1$ where k_{max} is the maximum number of spatial resolution levels supported by the encoder. In each bitplane, the FS-SPIHT coder starts encoding from the maximum resolution level (k_{max}) and proceeds to the lowest level (level 1). For the resolution-dependent sorting pass of the lists that belong to level k, the algorithm first does the sorting pass for the coefficients in the LIP$_k$ in the same way as SPIHT and then processes the LIS$_k$ list. During processing the LIS$_k$, sets that lie outside the resolution level k are moved to the LIS$_{k-1}$. After the algorithm has finished the sorting and refinement passes for level k it will do the same procedure for the lists related to level $k - 1$. According to the magnitude of the coefficients in the wavelet pyramid, coding of higher resolution bands usually starts from lower bitplanes. The total number of bits belonging to a particular bitplane is the same for SPIHT and FS-SPIHT, but FS-SPIHT arranges them according to their spatial resolution dependency.

In the following we first define the sets and symbols required by FS-SPIHT. These are the same as for the original SPIHT algorithm. Then we list the entire FS-SPIHT coding algorithm.

Definitions:

- $c(i, j)$: wavelet transformed coefficient at coordinate (i, j)
- $O(i, j)$: set of coordinates of all offspring of node (i, j)
- $D(i, j)$: set of coordinates of all descendants of node (i, j)
- $L(i, j)$: set of coordinates of all leaves of node (i, j). $L(i, j) = D(i, j) - O(i, j)$.
- H: set of coordinates of all nodes in the coarsest level of wavelet coefficients pyramid
- $S_n(i, j)$: significance test of a set of coordinates $\{(i, j)\}$ at bitplane level n

$$S_n(i,j) = \begin{cases} 1 & \text{If } \max_{\{(i,j)\}}\{|c(i,j)|\} \geq 2^n \\ 0 & \text{otherwise} \end{cases}$$

- Type A sets: for sets of type A the significance tests are to be applied to all descendants $D(i, j)$.
- Type B sets: for sets of type B the significance tests are to be applied only to the leaves $L(i, j)$.
- n_{max}: maximum bitplane level needed for coding

$n_{max} = \lfloor \log_2(\max_{\{(i,j)\}}\{|c(i,j)|\}) \rfloor$

- k_{max}: maximum level of spatial scalability to be supported by the bitstream ($1 \leq k_{max} \leq N+1$).
- β_k: A set of subbands in the decomposed image that belong to spatial resolution level k ($1 \leq k \leq k_{max}$) of the image.

FS-SPIHT coding steps:

1. **Initialization**
   - $n = n_{max}$, and output n;
   - LSP$_k = \emptyset$, $\forall k, 1 \leq k \leq k_{max}$;
   - LIP$_k = \begin{cases} \emptyset & \text{for } 1 \leq k < k_{max} \\ \{(i,j)\},\ \forall (i,j) \in H & k = k_{max} \end{cases}$
   - LIS$_k = \emptyset$, $\forall k, 1 \leq k < k_{max}$;
   - LIS$_{k_{max}} = \{(i,j)\}$ as type A, $\forall (i,j) \in H$ which have descendants;
   - $k = k_{max}$;
2. **Resolution-Dependent Sorting Pass**
   - SortLIP(n,k);
   - SortLIS(n,k);
3. **Refinement Pass**
   - RefineLSP(n,k);
4. **Resolution Scale Update**
   - if $(k > 1)$
     - $k = k - 1$;
     - go to step 2;
   - else, $k = k_{max}$;
5. **Quantization-Step Update**
   - if $(n > 0)$

```
        – $n = n - 1$;
        – go to step 2;
    • else, end of coding.
```

Pseudo Code:

```
SortLIP$(k, n)\{$
  • for each entry (i,j) in the LIP$_k$ do:
      – output $S_n(i, j)$;
      – if $(S_n(i, j) = 1)$, then move $(i, j)$ to the LSP$_k$, output the sign of $c(i, j)$;
  $\}$
```

```
SortLIS$(n, k)\{$
  for each entry $(i, j)$ in the LIS$_k$
  • if the entry is of type A
      – if $(\forall(x, y) \in D(i, j) : (x, y) \notin \beta_k)$, then move $(i, j)$ to LIS$_{k-1}$ as type A;
      – else
          ∗ output $S_n(D(i, j))$;
          ∗ if $(S_n(D(i, j)) = 1)$ then for each $(p, q) \in O(i, j)$
              · output $S_n(p, q)$;
              · if $(S_n(p, q) = 1)$, add $(p, q)$ to the LSP$_k$, output the sign of $c(p, q)$;
              · else, add $(p, q)$ to the end of the LIP$_k$;
          ∗ if $(L(i, j) \neq \emptyset)$, move $(i, j)$ to the end of the LIS$_k$ as an entry of type B;
          ∗ else, remove entry $(i, j)$ from the LIS$_k$;
  • if the entry is of type B
      – if $(\forall(x, y) \in L(i, j) : (x, y) \notin \beta_k)$, then move $(i, j)$ to LIS$_{k-1}$ as type B;
      – else
```

```
        * output $S_n(L(i,j))$;
        * if $(S_n(L(i,j)) = 1)$
            · add each $(p,q) \in O(i,j)$ to the end of the LIS$_k$ as an entry of type A;
            · remove $(i,j)$ from the LIS$_k$.
  }
```

```
RefineLSP$(n,k)${
  • for each entry $(i,j)$ in the LSP$_k$, except those included in the last sorting pass (i.e. the
    ones with the same $n$), output the $n^{th}$ most significant bit of $|c(i,j)|$.
  }
```

Note that the total storage requirement for all lists LIP$_k$, LSP$_k$, and LIS$_k$ for $k = k_{max}, k_{max} - 1, \ldots, r$ is the same as for the LIS, LIP, and LSP used by the SPIHT algorithm.

To support bitstream parsing by an image server/parser, 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

Fig. 2 shows the bitstream structure generated by the encoder. The bitstream is divided into different parts according to the different bitplanes. Inside each bitplane part, the bits that belong to the different spatial resolution levels 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 image dimension, number of wavelet decomposition levels, and the maximum quantization level. At the beginning of each bitplane there is an additional header that provides the information required to identify each resolution level.

A single encoded bitstream for the full-resolution image is stored on an image server. Different users with different requirements send their request to the server and the server or a parser within the network provides them with properly tailored bitstreams that are easily generated by selecting the related parts of the original bitstream and ordering them in such a way that the user requests are fulfilled. Fig. 3 illustrates the principle. To carry out the parsing process, the image server or parser does not need to decode any parts of the bitstream.

Fig. 4 shows an example of a reordered bitstream for spatial resolution level r. In each bitplane only the parts that belong to the spatial resolution levels greater or equal to the requested level are kept and the other parts are removed. Note that all header information for identifying the individual bitplanes and resolution levels are only used by the image parser and does not need to be sent to the decoder.

The decoder required for decoding of the reordered bitstream follows the encoder with the output command replaced by an input command, similar to the original SPIHT algorithm. It needs to keep track of the various lists (LIS, LIP, LSP) only for resolution levels greater or equal to the required one. It can recover all information for updating the lists during the 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.

4 Experimental Results

In this section we present some numerical results for the FS-SPIHT algorithm. All results were obtained with 8 bit per pixel (bpp) monochrome images of size 512×512 pixels. We first applied five levels of wavelet decomposition with the 9/7-tap filters of [19] and symmetric extension at the image boundaries. The FS-SPIHT encoder was set to produce a bitstream that supports six levels of spatial scalability.

After encoding, the FS-SPIHT bitstream was fed into a parser to produce progressive (by quality) bitstreams for different spatial resolutions. The bitstreams were decoded with different rates and the fidelity was measured by the peak signal-to-noise ratio (PSNR). The bit rates for all levels were calculated according to the number of pixels in the original full size image.

Figs. 5 and 6 compare rate-distortion results of FS-SPIHT and SPIHT at different spatial resolution levels for test images. For spatial resolution level 1, the bitstream needed by the FS-SPIHT decoder can be obtained by simply removing the bitplane headers from the encoder output bitstream. The results clearly show that the FS-SPIHT completely keeps the progressiveness (by SNR) property of the SPIHT algorithm. The small deviation between FS-SPIHT and SPIHT is due to a different order of coefficients within the bitstreams. For resolution levels 2 and 3, the FS-SPIHT decoder obtained the proper bitstreams tailored by the parser 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 reordered FS-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 FS-SPIHT is much better than for SPIHT for resolution levels greater than one. As the resolution level increases, the difference between FS-SPIHT and SPIHT becomes more and more significant. All the results are obtained without extra arithmetic coding of the output bits. As shown in [7], an improved coding performance (about 0.3-0.6 dB) for SPIHT and consequently for FS-SPIHT can be achieved by further compressing the binary bitstreams with an arithmetic coder.

5 Conclusions

We have presented a fully scalable SPIHT algorithm that produces a bitstream which supports spatial scalability and can be used for multiresolution parsing. 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 parser without decoding the bitstream. The embeddedness is so fine granular that almost 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 algorithm is extendable for combined SNR, spatial and frame-rate scalable video coding and also for fully scalable coding of arbitrarily shaped still and video objects. 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.

Acknowledgment

The first author would like to acknowledge the financial support provided for him by the Ministry of Science, Research and Technology (MSRT), Iran and Kurdistan University, Sanandaj, Iran during doing this research as a part of his PhD study at the University of Wollongong, Australia.

References

[1] D. S. Taubman and M. W. Marcellin, *Jpeg2000: Image Compression Fundamentals, Standards, and Practice*, Kluwer, Boston, MA, 2002.

[2] C. Christopoulos, A. Skordas, and T. Ebrahimi, “The jpeg2000 still image coding system: an overview,” *IEEE Trans. Consumer Electronics*, vol. 46, no. 4, pp. 1103–1127, Nov. 2000.

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

[4] A. Zandi, J. D. Allen, E. L. Schwartz, and M. Boliek, “CREW: Compression with revesible embedded wavelet,” in *Proc. IEEE Data Compression Conf.*, Mar. 1995, pp. 212–221.

[5] Y. Chen and W. A. Pearlman, “Three-dimentional subband coding of video using the zero-tree method,” in *Proc. SPIE 2727-VCIP’96*, Mar. 1996, pp. 1302–1309.

[6] S. A. Martucci and I. Sodagar, “Entropy coding of wavelet coefficientsfor very low bit rate video,” in *Proc. IEEE Int. Conf. Image Processing*, Sept 1996, vol. 2, pp. 533–536.

[7] 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.

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

[9] Q. Wang and M. Ghanbari, “Scalable coding of very high resolution video using the virtual zerotree,” *IEEE Trans. Circ. and Syst. for Video Technology*, vol. 7, no. 5, pp. 719–727,

Oct. 1997.

[10] S. A. Martucci, I. Sodagar, T. Chiang, and Y.-Q. Zhang, “A zerotree wavelet video coder,” *IEEE Trans. Circ. and Syst. for Video Technology*, vol. 7, no. 1, pp. 109–118, Feb. 1997.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] H. Danyali and A. Mertins, “Highly scalable image compression based on SPIHT for network applications,” in *Proc. IEEE Int. Conf. Image Processing*, Rochester, NY, USA, Sept. 2002, 217-220.

[19] 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.

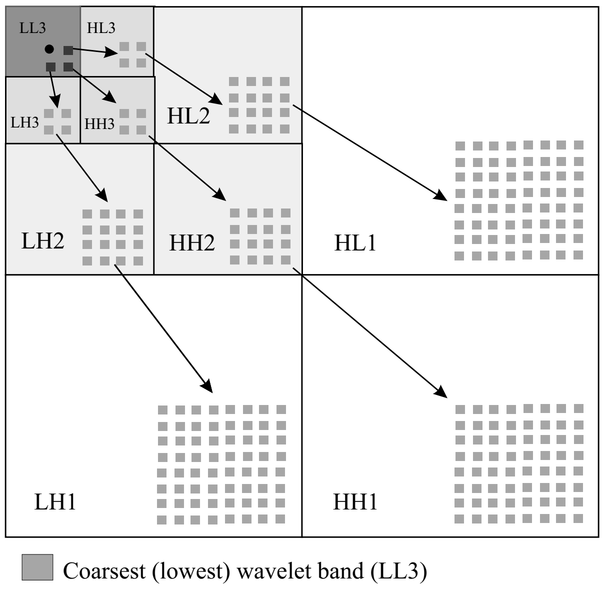


Figure 1: Orientation of trees across wavelet bands.

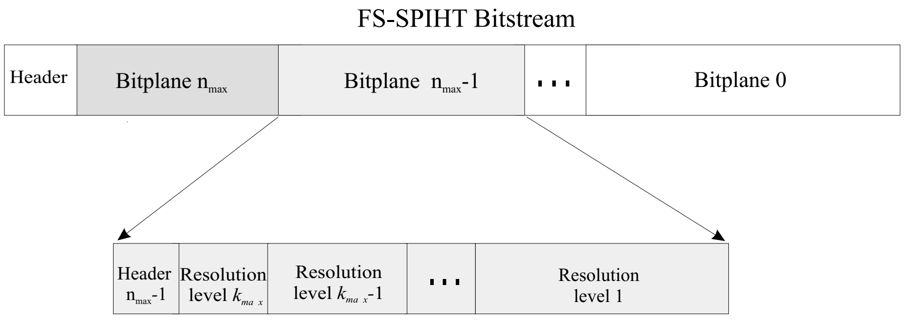


Figure 2: Structure of FS-SPIHT encoder bitstream which is made up of different parts according to spatial resolution and quality.

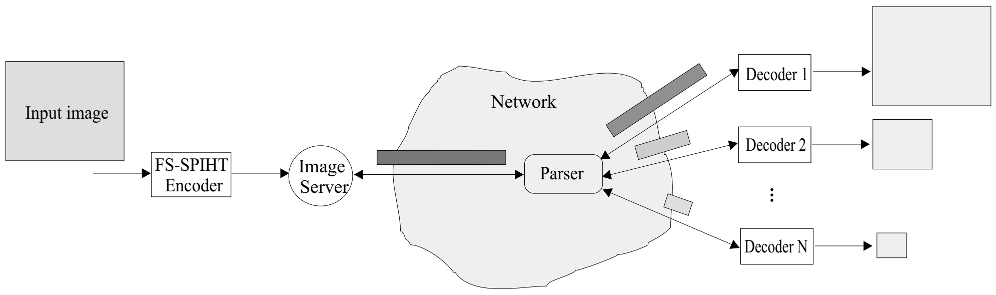


Figure 3: An example of an image server and a parser in a network for providing various bit-streams for different resolutions or/and quality levels requested by different users.

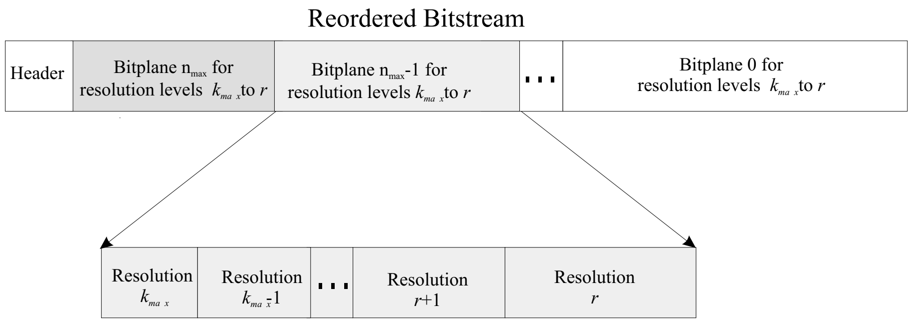

Reordered Bitstream
Header
Bitplane n_{max} for resolution levels k_{max} to r
Bitplane n_{max}-1 for resolution levels k_{max} to r
Bitplane 0 for resolution levels k_{max} to r
Resolution k_{max}
Resolution k_{max}-1
Resolution r+1
Resolution r

Figure 4: Reordered FS-SPIHT bitstream for spatial resolution level r decoding.

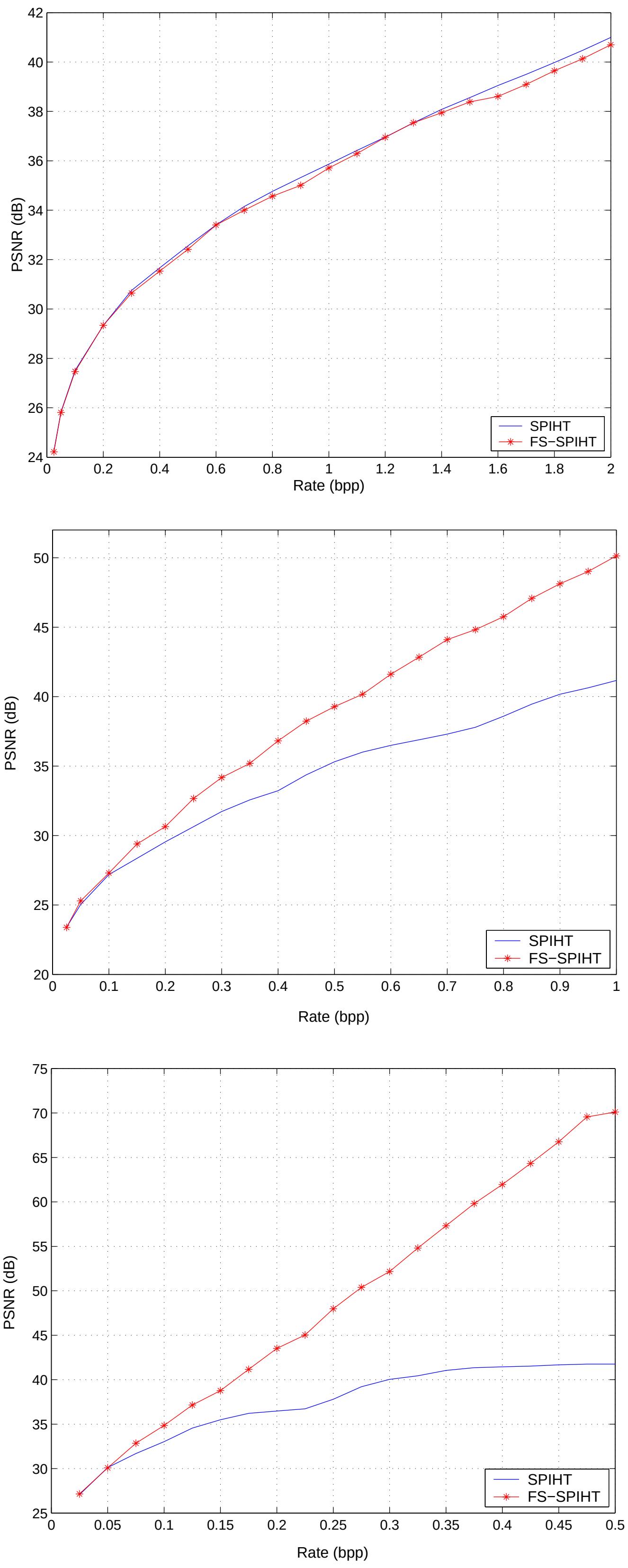
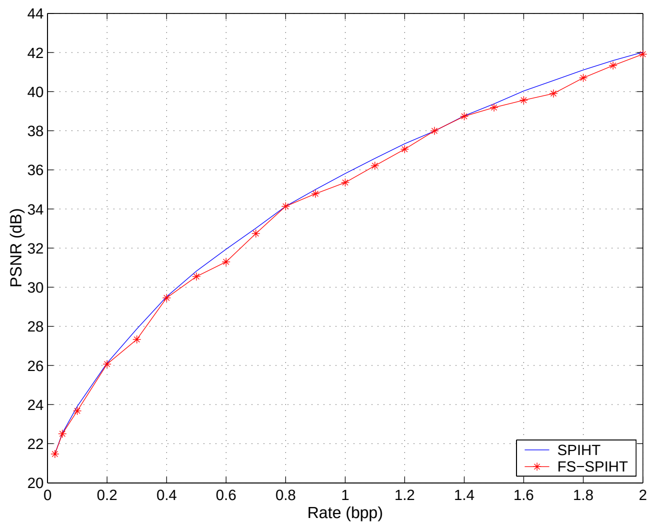
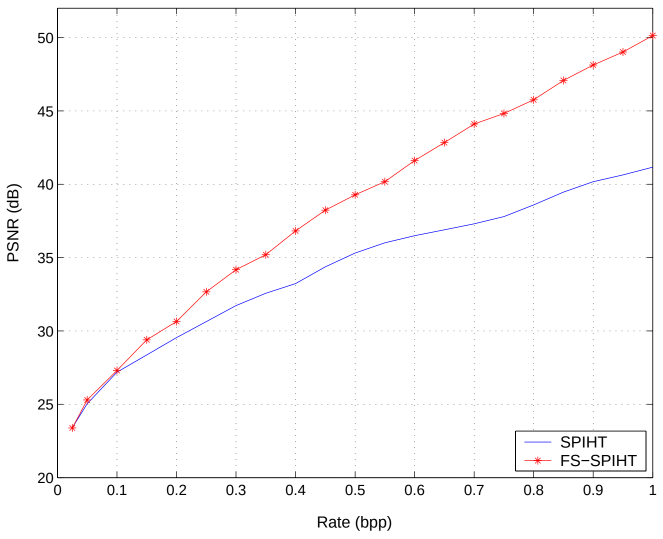


Figure 5: Comparison of rate-distortion results for the Goldhill test image at different spatial resolution levels. Top: level 1 (original image size 512×512); middle: level 2 (256×256); bottom: level 3 (128×128).





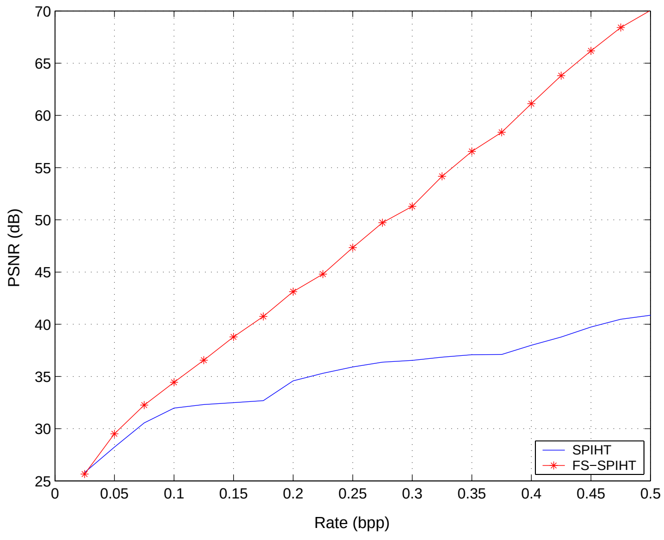


Figure 6: Comparison of rate-distortion results for the Barbara test image at different spatial resolution levels. Top: level 1 (original image size 512×512); middle: level 2 (256×256); bottom: level 3 (128×128).

Figure Captions

Fig. 1. Orientation of trees across wavelet bands.

Fig. 2. Structure of FS-SPIHT encoder bitstream which is made up of different parts according to spatial resolution and quality.

Fig. 3. An example of an image server and a parser in a network for providing various bitstreams for different resolutions or/and quality levels requested by different users.

Fig. 4. Reordered FS-SPIHT bitstream for spatial resolution level r decoding.

Fig. 5. Comparison of rate-distortion results for the Goldhill test image at different spatial resolution levels. Top: level 1 (original image size 512×512); middle: level 2 (256×256); bottom: level 3 (128×128).

Fig. 6. Comparison of rate-distortion results for the Barbara test image at different spatial resolution levels. Top: level 1 (original image size 512×512); middle: level 2 (256×256); bottom: level 3 (128×128).