Low-Memory Packetized SPIHT Image Compression[†]

Frederick W. Wheeler and William A. Pearlman
Rensselaer Polytechnic Institute
Electrical, Computer and Systems Engineering Dept.
Troy, NY 12180, USA
wheeler@cipr.rpi.edu, pearlman@ecse.rpi.edu

Abstract

The SPIHT image compression algorithm is modified for application to large images with limited processor memory. The subband decomposition coefficients are partitioned into small tree-preserving spatial blocks which are each independently coded using the SPIHT algorithm. The bitstreams for each spatial block are assembled into a single final bitstream through one of two packetization schemes. The final bitstream can be embedded in fidelity with a small expense in rate. SPIHT encoding and decoding of the spatial blocks can be done in parallel for real-time video compression.

1. Introduction

1.1. Motivation

The SPIHT algorithm is a fast and efficient technique for image compression [9]. Like EZW [10] and other embedded wavelet compression schemes [11, 6], SPIHT generally operates on an entire image at once. The whole image is loaded and transformed, and then the algorithm requires repeated access to all coefficient values. There is no structure to the order in which the coefficient values are accessed.

The random coefficient access requirement of the SPIHT algorithm hinders its use in certain memory constrained environments. Consider a processing architecture with two memory segments, a small fast memory and large slow memory. Depending on the application framework, these segments may be considered the on-chip cache, and external RAM of a microprocessor. In a virtual memory system, the segments would be external RAM and a hard disk swap space. Using full-image SPIHT in such a framework can cause time consuming memory swapping as data that must be accessed will frequently lie in the slower memory segment. It is more desirable to use a compression algorithm that can work with a portion of data that fits in the small fast memory, and only occasionally exchanges data between the memory segments.

The capability to encode a large image without storing the entire image in memory is an important feature in the JPEG 2000 requirements specification. The system described here has been proposed to the JPEG 2000 standardization committee. The encoder used to generate results for this paper is implemented as a module in the JPEG 2000 verification module test software.

1.2. Overview

To effectively apply the SPIHT algorithm in constrained memory environments, we have modified the overall algorithm so that the conversion of the subband coefficients to an encoded bitstream is done only for a small portion of the coefficients at a given time. Thus, only a small portion of the coefficients must be stored in fast cache memory.

Instead of placing quantized coefficients in one full-size subband decomposition, they are placed in many small spatial blocks. The partitioning is done in such a way that each hierarchical coefficient tree in the full-size subband decomposition appears in one of the spatial blocks.

The basic SPIHT algorithm is applied to each spatial block independently to produce a fidelity embedded sub-bitstream for each block. These independent embedded sub-bitstreams are then assembled to make a final bitstream that is either fixed rate or fidelity embedded. If it is fidelity embedded, it may be coarsely or finely embedded, depending on parameters chosen at the encoder.

Several codecs have been developed that apply an embedded tree-based codec independently to spatial blocks. Creusere has used this technique with EZW for parallel processing [3], channel error resilience [4], and region of interest decoding [5]. Independent spatial block coding is used for a parallel VLSI design in [2, 7]. Rogers and Cos-

[†] This work was supported in part by the National Science Foundation under grant numbers NCR-9523767 and EEC-9812706, and by a fellowship from Sun Microsystems.

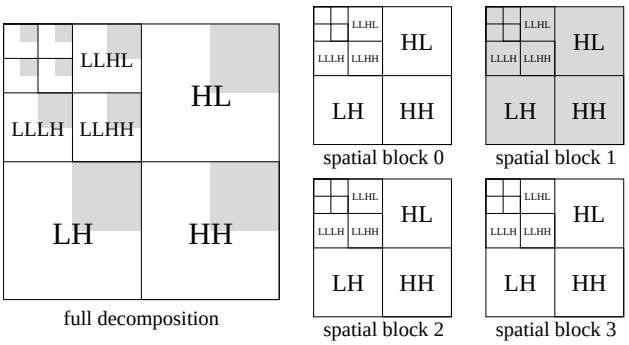

Figure 1. A full subband decomposition is partitioned into 4 equal sized spatial blocks is depicted. The coefficients in the gray regions are all in spatial block 1.

man have also used similar methods for error resilience [8]. Our emphasis in this work is reduced processor memory achieved through independent spatial block coding with the SPIHT algorithm while still generating a global bitstream that is progressive in fidelity across the whole.

2. SPIHT Compression in Independent Spatial Blocks

2.1. Spatial Blocks

To apply the SPIHT algorithm, or any tree-based codec, to small portions of the subband decomposition at a time, the decomposition is partitioned into spatial blocks, as shown in Fig. 1. A spatial block is a subset of the coefficients consisting of one or more hierarchical trees. The number of trees is chosen to meet a desired block size, typically 64 by 64 or 128 by 128 coefficients. The trees in a spatial block are rearranged in a smaller 2-D array such that the usual hierarchical relationship between the coefficients is preserved. Due to the nature of the subband decomposition, each spatial block holds the coefficients needed to reconstruct a contiguous block of the original image, neglecting overlap stemming from the width of the wavelet filters.

2.2. Subband Transform

The first step in the SPIHT image compression algorithm is the dyadic subband transform. This transform is usually computed for the entire image at once. For our application it is important that the transform instead be implemented with a rolling algorithm. Just as 1-D FIR filters can be implemented to produce filtered data after a lag in the time domain, the 2-D separable filters used for the subband decomposition can be implemented to produce the coefficients

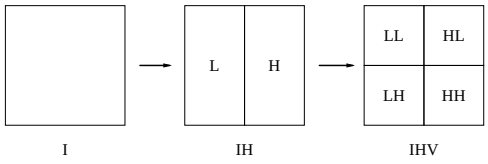

Figure 2. Computation stages for one level of a subband decomposition

filling a spatial block after a lag in the spatial domain. With spatial blocks processed as they become available, and then released, the codec requires a fraction of the memory that would be needed to hold an entire image or subband transform.

A practical means to accomplish this is with a line-based transform. Initially, consider just the first level of decomposition, and its two processing stages. The original image I is transformed horizontally to generate the IH image, and then vertically to generate the IHV image, seen in Fig. 2.

After each row of I is moved into memory, wavelet filters are immediately applied to produce the corresponding row of IH. The rows of IHV depend on several adjacent rows of IH, with the exact number depending on the filter length, so the rows of IH are kept in a rolling buffer that moves down the image. As soon as the rolling buffer holds enough rows of IH a vertical wavelet filter is applied to generate a row of IHV. The rolling buffer releases memory for rows which will no longer be needed and thus uses memory proportional to the width of the image, but not the height. This procedure produces the exact same coefficient values that would be produced by a full-image transform. All that is different is the order in which the computations are performed.

To produce a dyadic transform, the above technique is applied recursively to the low-pass band. As soon as a row of the low-pass band is produced, it is passed to an independent transform engine operating on its level.

The line-based subband transform engine produces a row of spatial blocks at a time. As soon as a row of spatial blocks is produced, each block is passed in turn to a SPIHT encoder which generates the sub-bitstream for that spatial block. The buffer memory used for coefficients of encoded spatial blocks is immediately released.

On a Multiple Instruction Multiple Data (MIMD) multiprocessor architecture, several of the spatial blocks could be processed by the SPIHT codec simultaneously.

2.3. SPIHT Encoding

Each spatial block of subband coefficients is moved to fast cache memory and processed by the SPIHT encoder once, as soon at it is available from the rolling wavelet transform engine. The encoder generates an independent, embedded in fidelity, sub-bitstream for each spatial block.

These sub-bitstreams are stored in external slow memory until all spatial blocks have been encoded.

A full-image SPIHT encoder simply processes and outputs bits until the full-image bit budget is met. Since each spatial block is processed only once, in turn, by the encoder and at the time of processing it is not known what rate will later be assigned to a particular block, over-coding, coding to a higher bit rate, is performed. Each block is encoded to 2.5 or 3 times the desired overall rate in bits per pixel. Unneeded portions of the sub-bitstreams will be pruned after rate allocation. An alternative system could avoid the over coding at the expense of repeatedly swapping spatial block data into cache memory.

2.4. Rate Allocation

Within each bitplane, the SPIHT algorithm has three passes, corresponding to the processing of the LIS, LIP and LSP [9]. During the LIS and LIP, or sorting, passes, newly significant coefficients and their signs are revealed to the decoder. During the LSP, or refinement, pass coefficients found significant for previous bitplanes have their bit from the current plane sent to the decoder.

Given a total rate budget, we desire to allocate rate from each sub-bitstream such that the RMSE between the original and reconstructed images is minimized. Rate will be allocated to each block according to the activity in that block. Since we are computing MSE in the transform domain, we rely on the fact that the wavelet transform is nearly unitary.

For each spatial block, we need to know the approximate distortion that will result from truncating the sub-bitstream at any given point. The encoder has the original image and knows how the decoder will reconstruct each coefficient after each information bit is received. Thus, the encoder is capable of finding the exact RMSE of a reconstructed spatial block no matter where the sub-bitstream is truncated, but at a significant computational cost.

Instead, we use a fast and simple technique. For each spatial block, each bitplane, and each pass, we count the number of newly significant coefficients revealed (or the number refined if in a refinement pass) during the pass, and the number of bits placed on the sub-bitstream during the pass. We then approximate the reduction in distortion gained by decoding the pass as follows.

Suppose that a sorting or refinement pass is operating in bitplane n with significance threshold $\tau = 2^n$, and during the pass B bits are placed on the sub-bitstream. Let N_s be the number of newly significant coefficients found if the pass is a sorting pass, and let N_r be the number of coefficients refined if the pass is a refinement pass. All bits sent to the sub-bitstream are counted in B. In a sorting pass, B includes all significance test bits and all sign bits. In a refinement pass, B equals N_r, the number of coefficients refined. The encoder records B and N_s or N_r, as appropriate, for each pass.

For a sorting pass, assuming that the original values of the N_s newly significant coefficients are random and have magnitudes evenly distributed between τ and 2τ, and that the reconstructed values will now be $\pm 1.5\tau$ instead of 0, the expected reduction in squared error from decoding these B bits is $\frac{27}{12} N_s \tau^2$. In a refinement pass that refines N_r coefficients, the expected reduction in squared error is $\frac{1}{4} N_r \tau^2$.

With this data collected by the SPIHT encoder, we generate an approximate operational rate-distortion characteristic for each sub-bitstream. The slope of the convex hull of this function is readily determined. Given a bit budget, we are now faced with a standard rate allocation problem which can be nearly optimally solved by iteratively increasing the number of bits taken from the sub-bitstream with the best cost-benefit ratio, that is, with the most negative rate-distortion slope.

2.5. Packetization

The final step is to combine the stored sub-bitstreams into a single final bitstream. We have implemented two final bitstream assembly modes, a fixed rate mode and a progressive in fidelity mode. The fixed rate mode is actually a special case of the fidelity progressive mode.

In the fixed rate mode, the final bitstream consists of a single global header holding basic parameters and then a series of packets, one for each spatial block. Each block has an index known to both the encoder and decoder, as seen in Fig.1. In index order, a packet for each spatial block is inserted into the final bitstream. The fixed size packet header holds the length of the data section of the packet, and the data section of the packet holds bits from the sub-bitstream.

Assembly of this final bitstream is straightforward once the bit allocation among the sub-bitstreams is performed. The encoder starts with the total bit budget, subtracts the size of the global header and the total size of all of the required packet headers, in advance. The remaining bits are reserved for packet data and are allocated to the sub-bitstreams using the technique of the previous section.

In fidelity embedded mode, the final bitstream holds a global header and then a series of packets, one for each spatial block, as above. However, there may be any number of subsequent series of packets. This type of final bitstream has intermediate points called rate layers at which the rate used so far is allocated such that the fidelity of the whole image is maximized. Rate allocation for fidelity embedded mode is performed in stages. First, rate is allocated for the initial rate layer, exactly as in fixed rate mode. One packet for each spatial block is then immediately inserted onto the final bitstream. For each subsequent rate layer, additional

rate is allocated to each spatial block, and a packet for each spatial block is inserted on the final bitstream. In this mode, the sub-bitstreams are not contiguous on the final bitstream.

Suppose a 512 by 512 image is partitioned into 16 64 by 64 spatial blocks and encoded to 1 bit per pixel in fixed rate mode. The overhead of the 4 byte packet headers is 512 bits, or 0.20%. This 0.20% is incurred for each rate layer in fidelity embedded mode. Five layers would mean a packet header overhead of 1%.

If channel error resilience is required, a logical approach is to protect early portions of each sub-bitstream more robustly than later parts, where errors will have less of an effect on the reconstructed image. Even if an information bit is corrupted, the effect on the reconstructed image is localized to the region affected by a single spatial block. Resynchronization markers between packets can allow recovery from missing partitions of the bitstream.

The ability to decode a small region of interest of the image is a natural feature of the packetized SPIHT bitstream. To decode a subset of the spatial blocks, the decoder can skip packets it does not need.

Our driving goal is to minimize image distortion given a fixed bit budget. An alternative to rate control, however, is distortion control. A distortion control mode could encode a predetermined set of bitplanes, up to a particular pass in the final plane, for fractional bitplane resolution. All encoded bits would then be transmitted, regardless of rate. In the context of our rate allocation and packetization scheme, this in effect sets the rate for each spatial block for the final rate layer. Additional intermediate rate layers may still be chosen using the rate allocation technique. If no intermediate layers are desired there is the added benefit that the sub-bitstreams need not be stored. As soon as they are produced, they can be placed in a packet, inserted into the final bitstream and transmitted. An important benefit of distortion control used in this way is that there is no need to over code each spatial block.

3. Results and Conclusions

Coding results for the 512 by 512 grayscale lena and goldhill images are plotted in Fig. 3. For these tests, we used a five-level decomposition using the 9/7 biorthogonal wavelet [1]. The solid lines show rate vs. distortion performance for full-image SPIHT without arithmetic coding. The dash-dot lines show the performance using independent 64 by 64 spatial blocks and packetization for fidelity embedding with only a few rate layers, at 0.1, 0.5 and 2 bpp. A scalloping effect is seen in these plots as the performance of the block based SPIHT codec catches up the the full-image SPIHT codec at these rates. The dashed lines show the performance with rate layers at increments of 0.2 bpp. At rates beyond the first layer at 0.2 bpp, this curve follows full-

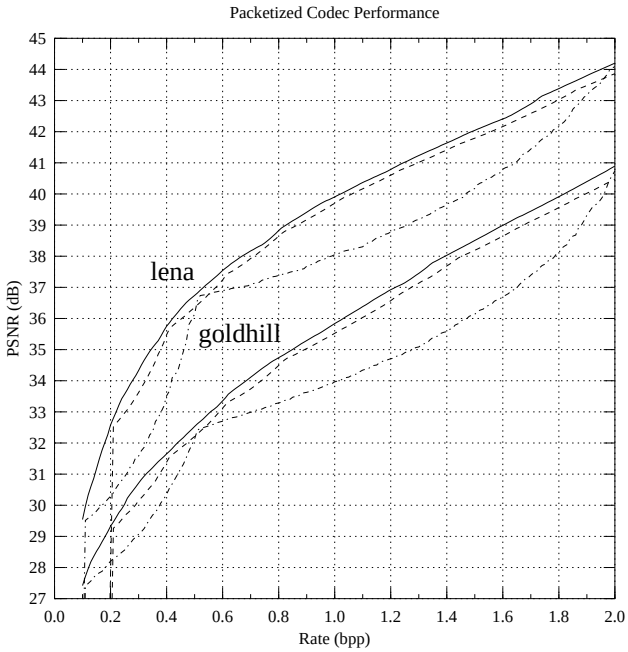

Figure 3. Coding performance for the lena and goldhill images

image SPIHT closely, with some loss due to packet header overhead. On all plots for the packetized codecs we observe a very low PSNR before the first layer is decoded. Before this point some spatial blocks receive zero encoded bits and contribute dramatically to the distortion. All decoded images in each curve were recovered from a single embedded encoded file, truncated at the desired rate.

No arithmetic coding was used on the symbols produced by the SPIHT algorithm for these results. Back-end arithmetic coding with using contexts and joint encoding generally improves SPIHT by about 0.5 dB, and we would expect that improvement uniformly here as well.

A couple of examples illustrate the nature of the optional coarse fidelity embedding feature. Figure 4 shows a reconstructed lena image after encoding the image to 1 bpp in fixed rate mode, but then decoding only half of the bitstream, or 0.5 bpp. Because subband domain spatial blocks correspond to image domain blocks, we see clearly that about half of the spatial blocks have been decoded, following a raster order. A few more than half are decoded because of the greater image activity in the lower half of this image. Figure 5 shows the lena image after encoding the image to 1 bpp in fidelity progressive mode with rate layers at 0.02 and 1 bpp, but then only decoding to 0.5 bpp. Decoding to 0.5 bpp fully decodes all of the packets in the first rate layer at 0.02 bpp because they come first in the bitstream. Thus, all of the image is reconstructed to at least the fidelity achievable with 0.02 bpp. The 1 bpp layer is then partially decoded and about half of the image is reconstructed to a

Figure 4. A fixed rate packetized SPIHT decoded image after decoding half of the bitstream

Figure 5. A coarsely fidelity progressive packetized SPIHT decoded image after decoding half of the bitstream

higher fidelity.

To demonstrate the peril in not heeding the size of cache memory we ran two codecs in a very large image using a general purpose 167 MHz Sun Ultra 1 computer with 64M RAM. In this experiment, we consider RAM to be the fast cache memory and the disk to be the external slow memory.

We applied both a full-image and independent spatial block SPIHT codec to a 2k column by 5k row pixel image. Note that this implementation is not heavily optimized. The independent spatial block SPIHT codec encoded the image in 220 sec. and decoded in 43 sec. total with no significant disk swapping. The SPIHT algorithm itself, took only 20.90 sec. for encoding and 5.16 sec. for decoding. The full-image SPIHT encoder never finished encoding this large image. It was stopped manually after an hour of spending 97% of the time waiting for memory to be swapped from the disk.

References

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Processing*, 1(2):205–220, April 1992.

[2] J. Bae and V. K. Prasanna. A fast and area-efficient VLSI architecture for embedded image coding. In *Proc. of the International Conf. on Image Processing*, pages 452–454, 1995.

[3] C. D. Creusere. Image coding using parallel implementations of the embedded zerotree wavelet algorithm. In *Proc. of the IS&T/SPIE Symposium on Electronic Imaging*, volume 2668, 1996.

[4] C. D. Creusere. A new method of robust image compression based on the embedded zerotree wavelet algorithm. *IEEE Trans. on Image Processing*, 6(10):1436–1442, October 1997.

[5] C. D. Creusere. Spatially partitioned lossless image compression in an embedded framework. In *Proc. of the 31st Asimolar Conf. on Signals, Systems and Computers*, November 1997.

[6] M. J. Gormish, E. L. Schwartz, A. Keith, M. Boliek, and A. Zandi. Lossless and nearly lossless compression for high quality images. In *Proc. of SPIE, Vol. 3025*, February 1997.

[7] N. Park, J. Bae, and V. K. Prasanna. Synthesis of VLSI architectures for tree-structured image coding. In *Proc. of the International Conf. on Image Processing*, pages 999–1002, 1996.

[8] J. K. Rogers and P. C. Cosman. Wavelet zerotree image compression with packetization. *IEEE Signal Processing Letters*, 5(5):105–107, May 1998.

[9] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchal trees. *IEEE Trans. on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[10] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12):3445–3462, December 1993.

[11] A. Zandi, J. D. Allen, E. L. Schwartz, and M. Boliek. CREW: Compression with reversible embedded wavelets. In *Proc. of the Data Compression Conf.*, pages 212–221, 1995.