Setting priorities: a new SPIHT-compatible algorithm for image compression

Diego Dugatkin, Michelle Effros

Diego Dugatkin, Michelle Effros, "Setting priorities: a new SPIHT-compatible algorithm for image compression," Proc. SPIE 4119, Wavelet Applications in Signal and Image Processing VIII, (4 December 2000); doi: 10.1117/12.408670

SPIE.

Event: International Symposium on Optical Science and Technology, 2000, San Diego, CA, United States

Invited Paper

Setting Priorities: A New SPIHT-Compatible Algorithm for Image Compression

Diego Dugatkin and Michelle Effros

Department of Electrical Engineering, MC 136-93,
California Institute of Technology, Pasadena, CA 91125, USA

ABSTRACT

We introduce a new algorithm for progressive or multiresolution image compression. The algorithm improves on the Set Partitioning in Hierarchical Trees (SPIHT) algorithm by replacing the SPIHT encoder. The new encoder optimizes the multiresolution code performance relative to a user-defined probability distribution (or priority function) over the code's rates or resolutions. The new algorithm's decoder is identical to the SPIHT decoder. The resulting code achieves the optimal expected performance across resolutions subject to the constraints imposed by the use of the SPIHT decoder and the distribution (or priorities) over resolutions set by the user. The encoder optimization yields performance improvements at the rates or resolutions of greatest importance (according to the encoder's priority function) at the expense of performance degradation at low priority rates or resolutions. The algorithm is fully compatible at the decoder with the original SPIHT algorithm. In particular, the decoder requires no knowledge of the priority function employed at the encoder. Experimental results on an image containing both text and photographic material yield up to 0.86 dB performance improvement over SPIHT at the resolution of highest priority.

Keywords: Multiresolution source coding, successive refinement compression algorithms, SPIHT, wavelets, priorities

1. INTRODUCTION

Multiresolution source codes, also known as progressive transmission, embedded, or successive refinement compression algorithms, are becoming increasingly popular for image compression applications. The success of algorithms such as the Embedded Zero-tree Wavelet (EZW) algorithm of Shapiro[1] and the Set Partitioning in Hierarchical Trees (SPIHT) algorithm of Said and Pearlman[2] has led to the proposed application of multiresolution source codes in an enormous array of applications, many of which don't even require multiresolution codes. Multiresolution codes like EZW, SPIHT, and their descendants are attractive algorithms for image compression due to both their simplicity and their good performance across a wide range of rates and distortions.

Yet there is a price associated with multiresolution coding. In general, the performance of a multiresolution code cannot be simultaneously as good at all possible rates as single-resolution codes of those same rates. This statement is verified both in theory and in practice. A source is said to be successively refinable if there is no rate-distortion penalty associated with multiresolution source coding on that source. That is, a source is successively refinable if the optimal multiresolution source coding performance of the given source achieves the distortion-rate bound at all rates. Work by Equitz and Cover[3,4] proves the existence of sources that are not successively refinable. Later works demonstrate that even the simplest of Gaussian mixtures is not successively refinable[5] and suggest that most sources encountered in practice suffer performance losses when coded with multiresolution codes.[6]

The performance costs of multiresolution codes can likewise be observed in practice. For example, the SPIHT algorithm describes an image one bit-plane at a time in a manner that guarantees a lossless image description if allowed to proceed to a high enough rate. In order to guarantee eventual lossless performance, the algorithm describes each bit-plane to perfect accuracy. Unfortunately, the accurate description of some coefficients (e.g., isolated high energy coefficients in the highest frequency bands) is very expensive in rate and achieves little distortion benefit. While this expense may be justified if lossless coding (high-rate, 0-distortion) is the highest priority, it is often not

This material is based upon work supported by the Oringer Fellowship, NSF Award No. CCR-9909026, and the Intel Technology for Education 2000 program.
D.D.: E-mail: diego@caltech.edu
M.E.: E-mail: effros@caltech.edu

Wavelet Applications in Signal and Image Processing VIII, Akram Aldroubi, Andrew F. Laine, Michael A. Unser, Editors, Proceedings of SPIE Vol. 4119 (2000) © 2000 SPIE · 0277-786X/00/$15.00

justified when a low-rate data description is desired. For example, the performance of zerotree-style algorithms improves when the codes are optimized for a particular target rate.[7]

The goal of this work is to achieve multiresolution coding performance that is optimal with respect to a user-defined priority function over the rates and resolutions of a multiresolution code. The given algorithm works within the SPIHT coding framework. We modify the SPIHT encoder by using a dynamic programming argument to optimize the encoded bit-stream relative to the given priority function. When all of the priority is placed on achieving zero distortion at the highest rate, the algorithm proposed here achieves performance identical to the performance of SPIHT. When other priorities are assigned, the performance of the proposed algorithm exceeds that of SPIHT. (Performance is optimized with respect to the given priority function and in the wavelet domain). Application of the optimal encoder is equivalent to distorting the image in some way and then applying the traditional SPIHT encoder. As a result, the algorithm's decoder is identical to the SPIHT decoder. In particular, the priority function need not be known to the system decoder.

The given optimization comes with an associated cost. The improvement in performance at the high-priority resolutions may lead to performance degradation at low-priority resolutions. It is important to note, however, that this cost is implicit in *all* multiresolution codes. That is, since no multiresolution code can achieve the best possible performance at all rates, each code must decide – implicitly or explicitly – where the greatest priorities lie. Fixed and variable rate multiresolution vector quantizers (MRVQs) incorporating explicit prioritization have been described previously.[8–10] The goal of the work described in this paper is to demonstrate the explicit incorporation of user-chosen priority functions in a more sophisticated multiresolution code, in particular, the SPIHT algorithm.

The remainder of this paper is organized as follows. Section 2 provides a background overview. Section 3 describes the optimized SPIHT encoder. Section 4 contains experimental results comparing the performance of SPIHT and this optimized variation. Section 5 provides a summary and conclusions.

2. BACKGROUND

The SPIHT Algorithm

Like its predecessor EZW, the SPIHT image compression algorithm is a bit-plane coder working in the wavelet domain. The bit-planes of the wavelet decomposition are described sequentially in order of decreasing significance. The code uses the bit values of the lower frequency coefficients to predict the bit values of the corresponding coefficients in higher frequency bands.

The SPIHT algorithm begins with some initial partition of the coefficients into sets. At each bit-plane, it describes the significance or insignificance of each set. (A set is called "significant" at bit-plane n if it contains one or more coefficients $C(i,j)$ such that $|C(i,j)| \geq 2^n$; that is, one or more coefficients that are non-zero in or before the nth bit-plane. Conversely the set is called "insignificant" if all coefficients in the set have magnitude less than 2^n.) Each significant set is then partitioned into subsets, and the significance or insignificance of each subset is likewise described. The process continues until each significant subset has exactly one coefficient. The algorithm next refines the descriptions for all coefficients declared significant in previous bit-planes and then begins the above procedure again at bit-plane $n-1$.

The partition used in the SPIHT algorithm groups together low and high frequency coefficients from the same spatial location in the original image. The algorithm is most efficient in smooth, "natural" images, where insignificance of low frequency coefficients is a good predictor of insignificance in the corresponding coefficients of higher frequency bands. The algorithm is least efficient when a scattering of significant coefficients in the high frequency bands causes the original partition to be broken into many subsets in order to describe a small number of significant coefficients.

Including Priority Functions in Multiresolution Source Codes

As discussed earlier, multiresolution source codes generally cannot simultaneously achieve the best possible performance at all resolutions. As a result, the optimization of multiresolution source codes relies on the use of priority functions. A priority function is a weighting describing the relative importance of the different resolutions to the system designer. The use of priority functions arises from the convex optimization employed in the derivation of multiresolution source coding bounds.[6] A summary of the implications of that theory for multiresolution code design follows.

Optimal design of L-resolution vector quantizers[8–10] requires minimization of the Lagrangian performance measure

$$J = \sum_{\ell=1}^{L} (\alpha_\ell D_\ell + \beta_\ell r_\ell),$$

where D_ℓ and r_ℓ here denote the total distortion and incremental rate at resolution-ℓ, respectively. In this equation, (α^L, β^L) describes the "angle" of a hyperplane tangential to the lower convex hull of the achievable rate-distortion region at a single point. An alternative characterization is achieved by replacing the incremental rates by total rates. Using $R_\ell = \sum_{i=1}^{\ell} r_i$ to denote the total rate at resolution ℓ, the above equation becomes

$$J = \sum_{\ell=1}^{L} \alpha_\ell (D_\ell + \lambda_\ell R_\ell),$$

where α_ℓ explicitly describes the priority on the ℓ^{th} resolution and $D_\ell + \lambda_\ell R_\ell$ is the familiar rate-distortion Lagrangian for resolution ℓ.

3. ALGORITHM

In this work we consider the application of the Lagrangian J in a SPIHT-compatible code. While this performance measure can be applied to a multiresolution code with an arbitrary number of resolutions, we here simplify our work with SPIHT by defining a number of resolutions equal to the number of significance levels or coefficient bit-planes described in the SPIHT algorithm. Using this approach, we define the incremental rate r_ℓ in the ℓth resolution of the SPIHT code as the rate used in describing the ℓth most significant bit-plane. We likewise define D_ℓ as the distortion, measured as mean squared error, given by the reproduction achieved through the description of the first ℓ bit-planes.

In the SPIHT algorithm, the values r_ℓ and D_ℓ achieved in the ℓth resolution description of a particular image are a direct function of the image coefficients and do not vary relative to the user's priorities over the resolutions. We modify the SPIHT algorithm to allow for the inclusion of priority functions and the optimization of the given Lagrangian performance functional J. The modification results from the following observation: accurate bit-plane descriptions of the wavelet coefficients are not necessary for good multiresolution source coding performance in the SPIHT algorithm. In fact, requiring accurate bit-plane descriptions may actually harm the performance of the SPIHT algorithm at the resolutions of greatest interest to the compression system user. We next explain this observation and then describe the algorithm by which the *optimal* (but not necessarily *accurate*) bit-plane descriptions can be found.

The significance decisions used by SPIHT describe the true significance of all coefficients in the wavelet decomposition of the compressed image. While describing significance information accurately is necessary if one intends to continue the bit-stream decoding to a lossless description of the wavelet coefficients, the decisions resulting from this desire for high-rate accuracy come at a cost. For example, describing a single isolated significant coefficient in the highest frequency band requires a large amount of rate, since isolating a significant high-frequency coefficient requires many set partitions and therefore many set descriptions. Further, the decrease in distortion from this description may be small, since only a single coefficient's description is updated using the given description. As a result, describing that coefficient as if it were insignificant may result in better performance relative to our prioritized performance measure.

The new algorithm optimizes the choice of the resolution at which each set of coefficients is declared significant. The goal of that optimization is the minimization of $J = \sum_{\ell=1}^{L} (\alpha_\ell D_\ell + \beta_\ell r_\ell)$ relative to the user-defined priorities (α^L, β^L).[8] Since the significance decision on each coefficient or set of coefficients may affect decisions on other coefficients or sets, the optimal decisions must be found in a global fashion.

Figure 1 shows the SPIHT zerotree on the wavelet pyramid and the corresponding partition hierarchy showing the sets and component subsets used in the SPIHT algorithm. Each leaf node (i, j) of the partition hierarchy represents a single wavelet coefficient $C(i, j)$. Each intermediate node is a set of coefficients used in the partition. The children of each intermediate node give the subsets into which the given set is partitioned when the set becomes significant. The intermediate nodes are labeled according to the convention established by Said and Pearlman.[2] The set $\mathcal{D}(i, j)$ corresponds to the set of "all descendants" of node (i, j) (*all* offspring), while the set $\mathcal{L}(i, j)$ is the set of "all descendants-except-children" of node (i, j). A single root node is added above the level of the initial SPIHT

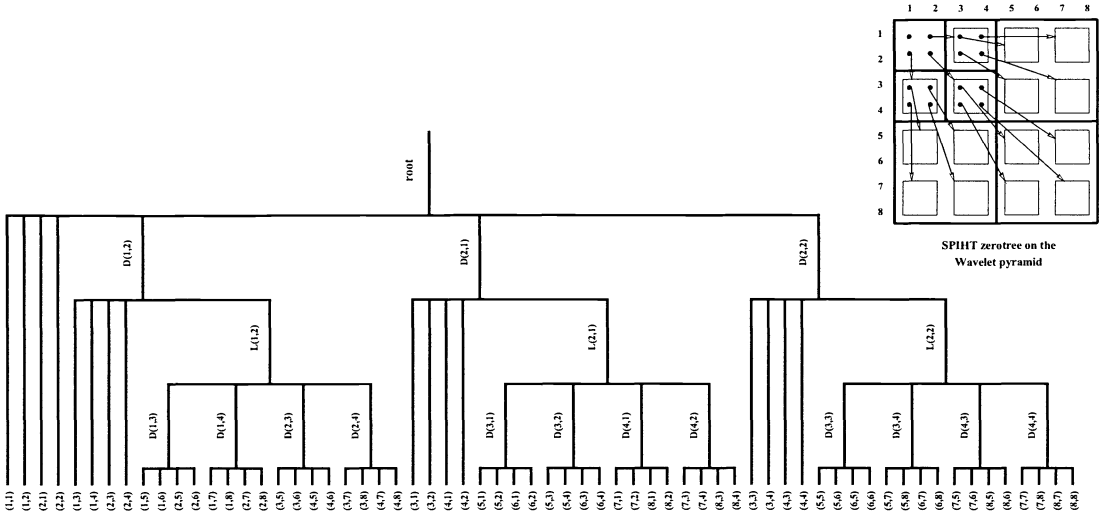

Figure 1. Decision-tree for a 2-level wavelet decomposition of an 8 x 8 image.

partition. The root node is the only node in the partition hierarchy of Figure 1 that does not arise directly from the SPIHT algorithm. The motivation for its addition becomes clear later in the algorithm description.

Our algorithm uses dynamic programming on the partition tree to find the collection of significance decisions that globally minimizes J. The dynamic programming algorithm is accomplished in two passes – a backward pass from the leaves to the root of the tree and a forward pass from the root to the leaves. The goal of the backward pass is to calculate and store an array of J-values, one for every resolution ℓ at each node of the tree. The goal of the forward pass is to encode the data using the best significance decisions from in the backward pass.

The sequence of decisions of set significance in our algorithm is the same as the sequence of decisions in SPIHT. That is, at each resolution the algorithm decides whether or not the sets in its current partition are significant, partitions the significant sets into subsets, and makes decisions about the significance or insignificance of those sets as well – continuing the process until all significant coefficients are isolated. The difference between the SPIHT algorithm and the approach described here is that the decisions themselves are based on a different criterion. In SPIHT, a set becomes significant at resolution ℓ if it contains at least one element with a one in bit-plane ℓ and no elements have ones in earlier bit-planes. In our method, a set becomes significant at resolution ℓ if declaring it significant at that resolution yields the best possible multiresolution rate-distortion trade-off.

The two passes are described in greater detail below.

Backward Pass (from the leaves to the root)

At every node, starting at the leaves and working toward the root, calculate a number of partial J-values equal to the number of resolutions in the data description. Next, store the calculated values, one for each resolution at each node, in a table to be used later in the algorithm.

As shown in Figure 1, each node of the tree is either a single coefficient $C(i,j)$ (a leaf node) or a set of coefficients $\mathcal{D}(i,j)$, $\mathcal{L}(i,j)$, or the tree root (an internal node). For each leaf node (i,j) and each resolution ℓ, we calculate a partial Lagrangian $\hat{J}_\ell(i,j)$. Calculation of $\hat{J}_\ell(i,j) = \sum_{m=1}^{L} \alpha_m D_{\ell,m}(i,j) + \sum_{m=\ell}^{L} \beta_m r_{\ell,m}(i,j)$ requires calculation of the L distortions $D_{\ell,1}(i,j), D_{\ell,2}(i,j), ..., D_{\ell,L}(i,j)$ and $L-\ell+1$ rates $r_{\ell,\ell}(i,j), r_{\ell,\ell+1}(i,j), ..., r_{\ell,L}(i,j)$ associated with declaring coefficient $C(i,j)$ significant at resolution ℓ. If the binary expansion of $C(i,j)$ first becomes significant at resolution ℓ then $\hat{C}_\ell(i,j) = C(i,j)$. If the accurate description of $C(i,j)$ first becomes significant before resolution ℓ, then the binary description of $\hat{C}_\ell(i,j)$ is given by 0...01...1 ($\ell - 1$ zeros and the remainder ones) since this is the best possible reproduction for $C(i,j)$ subject to the resolution-ℓ significance constraint. In each case, the distortions $D_{\ell,1}(i,j), ..., D_{\ell,L}(i,j)$ are the squared error distortions in the resolution 1 through L reproductions of $\hat{C}_\ell(i,j)$, and the rates $r_{\ell,\ell}(i,j), ..., r_{\ell,L}(i,j)$ are the corresponding incremental rates. (There is never a gain associated with

declaring a coefficient significant early, and thus such ℓ values need not to be considered.) Since the incremental rates $r_{\ell,1}(i,j),...,r_{\ell,\ell-1}(i,j)$ associated with coefficient $C(i,j)$ becoming significant at resolution ℓ depend on when the parent node becomes significant and that decision has not yet been made, only the partial J-value $\hat{J}_\ell(i,j)$ is calculated and stored.

For each internal node p and each resolution ℓ, $\hat{J}_\ell(p)$ gives the best partial Lagrangian that can be achieved if internal node p first becomes significant in resolution ℓ. Finding the best performance for each internal node and each resolution requires making optimal decisions on the significance of that node's children. These optimal decisions are accomplished through the use of the J-approximations previously calculated for those children and the addition of previously omitted rate values. (These rate values become available once an assumption is made about the resolution at which the parent node becomes significant.) In particular for any internal node p and any resolution ℓ, the optimal (partial) J-value $\hat{J}_\ell(p)$ is the weighted sum of the rate required to declare the given node significant at resolution ℓ plus a collection of values associated with the node's children. More specifically, if $\mathcal{C}$ is the set of all (internal of leaf) children of internal node p, then

$$\hat{J}_\ell(p) = \sum_{c\in\mathcal{C}} \min_{n\geq\ell}[\hat{J}_n(c) + \sum_{m=\ell}^{n-1} \beta_m r_{\ell,m}(c)].$$

Since the calculation of $\hat{J}$-values begins at the leaves of the tree and works up, layer by layer, to the root, $\hat{J}_n(c)$ for all $\ell \in 1,...,L$ and all $c \in \mathcal{C}$ are available prior to the calculation of $\hat{J}_\ell(p)$. Further, the incremental rates $r_{\ell,\ell}(c),...,r_{\ell,n-1}(c)$ achieved when p becomes significant at resolution ℓ and c becomes significant at resolution n are now known. The resulting minimized values $(\hat{J}_1(p),...,\hat{J}_L(p))$ are stored in the tree, and the resolutions used at the node's children to achieve the given minima are also stored. The process is continued, layer by layer, up to the root of the tree.

Notice that at the outcome of the above procedure, the L Lagrangian values stored at the root of the tree are complete rather than partial values since the tree root has no unknown parent from which to inherit uncertainty. Further, note that once the $\hat{J}$ values for a given layer of the tree have been calculated, the J-values from the previous (deeper) level are no longer required. As a result, the data structure originally used to store $\hat{J}$ values may be written over with resolution choices as the process works from leaves to root through the tree. Finally, notice that while the minimizations performed at each internal node rely on partial J-values, the unknown rate values at each calculation are constant for all terms in the minimization and thus do not affect its outcome.

Forward Pass (from the root to the leaves)

The result of the backward pass is a table with a number of entries equal to the number of nodes in the tree multiplied by the number of resolutions in the original SPIHT description. The values $J_1(root),...,J_L(root)$ given at the root of the tree describe the optimal Lagrangian performance achievable if the coding process begins at resolutions 1 through L, respectively. The remainder of the table is filled with the decisions used in achieving a particular J-value. For example, entries 1 through L associated with some child c of the root node give the resolutions at which child c should become significant if the root node becomes significant at resolutions 1 through L, respectively. The forward pass uses these values by first comparing the values $J_1(root),...,J_L(root)$ and choosing the value of ℓ that gives the best Lagrangian performance. The corresponding ℓ-value describes the first bit-plane to be used in the data description – effectively setting the n_{max} value to be used in SPIHT. If $\ell^* = \arg\min_{1\leq\ell\leq L} J_\ell(root)$, then reading entry ℓ^* for each of the root's children gives the optimal resolution at which each of those children should become significant given that the root became significant at resolution ℓ^*. Reading the corresponding information for each of those nodes' children and so on down the tree gives the optimal significance levels for all sets in the tree. We then encode the data using the SPIHT algorithm, but replacing SPIHT's original decisions about set significance with the optimal collection of decisions.

4. RESULTS

Figure 2 compares the performance of the algorithm described here to the performance of SPIHT on a 512 pixel by 512 pixel, 8 bit per pixel gray-scale image scanned from a page of the *IEEE Spectrum Magazine*. The image contains both photographic material and text. Both algorithms use the same 9-7 tap filters[11] in their wavelet decomposition.

We use an implementation of SPIHT that follows Said and Pearlman's original paper,[2] without any modifications or additions. No entropy coding is employed. The detailed description of SPIHT in Said and Pearlman's patent

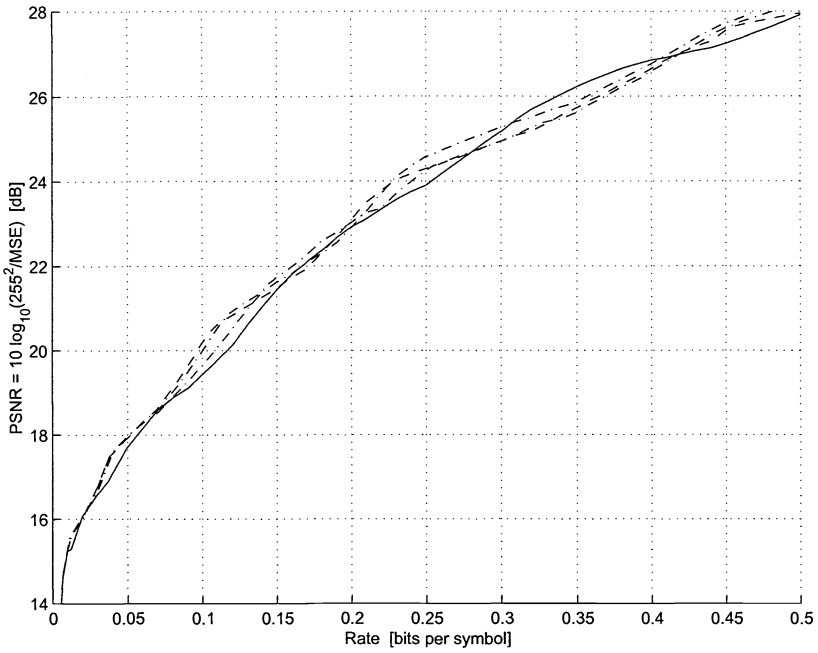

Figure 2. PSNR versus rate results of the new method optimized for three different priority schedules (dashed lines) and the standard SPIHT (solid line), for a 512×512, 8 bits per pixel gray-scale image. Each different priority schedule describes a different distribution of priorities across the different resolutions. The image, scanned from a page of the *IEEE Spectrum Magazine*, contains both photographic material and text.

as well as some existing SPIHT software use techniques not specifically mentioned in the original paper. While the performance of both SPIHT and our method could be improved using those techniques, our results correspond to a version of SPIHT that strictly follows the cited paper.

Figure 2 shows the peak signal-to-noise ratio (PSNR) as a function of rate, where $\text{PSNR} = 10\log_{10}(255^2/\text{MSE})$ dB, for both the optimized code and the original SPIHT algorithm. The optimized code is optimized relative to three different priority functions to show a spectrum of achievable results. By changing the priorities, we achieve better performance at the resolutions of highest priority at the expense of a degradation in performance at low priority resolutions. The advantage of the new method is the algorithm's flexibility, which allows the user to explicitly set the priorities in a manner that reflects the relative importance of the different resolutions. This contrasts with the SPIHT algorithm's implicit prioritization over the resolutions, which places priority on the seldom-used highest rate description. Figure 2 shows gains of up to 0.86 dB for the priority functions tested.

The flexibility benefits of the new algorithm come at the cost of higher computational complexity and greater memory requirements than those needed for the original SPIHT algorithm. The added computation is associated with the calculation of the Lagrangian performance functions during the backward pass, plus the subsequent sequence of decisions. The additional memory is required for the storage of the Lagrangian values (and the resolution choices that are written over them later). Each node of the tree requires as many Lagrangian calculations as the total number of resolutions (or bit-planes). Each calculation uses all J_ℓ Lagrangians of the descendants of that node, making the number of calculations or comparisons for any node proportional to L^2, the square of the total number of resolutions used in coding. The total number of nodes in the tree depends on the total number of wavelet decomposition levels, and is at most 1.3125 times the image size. Thus, the total number of Lagrangian calculations for the whole tree is, at most, $1.3125 \times \text{ImageSize} \times \text{L}^2$.

5. SUMMARY AND CONCLUSIONS

We have presented a low complexity multiresolution algorithm that incorporates the choice of a priority schedule over the resolutions into a SPIHT-compatible wavelet-based code. The resulting encoder produces a bit stream that is compatible with the standard SPIHT decoder, requiring no knowledge of the priority function at the decoder yet obtaining benefits relative to those priorities.

REFERENCES

1. J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing* **41**, pp. 3445–3462, December 1993.
2. A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology* **6**, pp. 243–250, June 1996.
3. W. H. R. Equitz, *Successive refinement of information.* Ph. D. Dissertation, Stanford University, Stanford, CA, 1989.
4. W. H. R. Equitz and T. M. Cover, "Succesive refinement of information," *IEEE Transactions on Information Theory* **37**, pp. 269–275, March 1991.
5. J. Chow and T. Berger, "Failure of succesive refinement for symmetrical Gaussian mixtures," *IEEE Transactions on Information Theory* **43**, pp. 350–352, January 1997.
6. M. Effros, "Distortion-rate bounds for fixed- and variable-rate multi-resolution source codes," *IEEE Transactions on Information Theory* **45**, pp. 1887–1910, Sept. 1999.
7. M. Effros, "Zerotree design for image compression: toward weighted universal zerotree coding," *Proceedings of the IEEE International Conference on Image Processing* , (Santa Barbara, CA), October 1997.
8. M. Effros, "Practical multi-resolution source coding: TSVQ revisited," in *Proceedings of the Data Compression Conference*, pp. 53–62, IEEE, (Snowbird, UT), Mar. 1998.
9. D. Dugatkin and M. Effros, "Multi-resolution VQ: parameter meaning and choice," in *Proceedings of the Thirty-Second Asilomar Conference on Signals, Systems, and Computers*, IEEE, November 1998.
10. M. Effros and D. Dugatkin, "Multi-resolution vector quantization," 2000. In preparation.
11. M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing* **1**, pp. 205–220, April 1992.