Geometric methods for wavelet-based image compression

Michael Wakin, Justin Romberg, Hyeokho Choi, Richard Baraniuk
Dept. of Electrical and Computer Engineering, Rice University
6100 Main St., Houston, TX 77005

ABSTRACT

Natural images can be viewed as combinations of smooth regions, textures, and geometry. Wavelet-based image coders, such as the space-frequency quantization (SFQ) algorithm, provide reasonably efficient representations for smooth regions (using zerotrees, for example) and textures (using scalar quantization) but do not properly exploit the geometric regularity imposed on wavelet coefficients by features such as edges. In this paper, we develop a representation for wavelet coefficients in geometric regions based on the wedgelet dictionary, a collection of geometric atoms that construct piecewise-linear approximations to contours. Our *wedgeprint representation* implicitly models the coherency among geometric wavelet coefficients. We demonstrate that a simple compression algorithm combining wedgeprints with zerotrees and scalar quantization can achieve near-optimal rate-distortion performance $D(R) \sim (\log R)^2/R^2$ for the class of piecewise-smooth images containing smooth C^2 regions separated by smooth C^2 discontinuities. Finally, we extend this simple algorithm and propose a complete compression framework for natural images using a rate-distortion criterion to balance the three representations. Our Wedgelet-SFQ (WSFQ) coder outperforms SFQ in terms of visual quality and mean-square error.

Keywords: Image compression, wavelets, wedgelets, edges, geometry

1. INTRODUCTION

1.1. The wavelet transform: Models and algorithms

Effective methods for transform-domain image compression rely on the successful interplay of three related components: a *transform* with desirable properties, accurate *models* for the transform coefficients, and efficient compression *algorithms* that operate according to the models. From a practical standpoint, transform coefficients for images of interest should exhibit a certain *structure*, or behavior that can be well modeled. Ideally, such models should also lead to fast, efficient processing algorithms; hidden Markov trees (HMTs) are one example where the organization of a model leads naturally to an elegant suite of signal processing algorithms.[1]

The wavelet transform is a key ingredient in most state-of-the-art image compression algorithms,[2–5] including the recent JPEG-2000 standard.[6] Historically the use of wavelets in image processing arose primarily due to the success of wavelets in one-dimensional (1-d) signal processing. The wavelet transform provides a multiscale analysis that is localized in both space and frequency; due to this arrangement, 1-d wavelets provide efficient representations for the large and useful class of piecewise-smooth 1-d signals. Smooth components of these signals are well-localized in the frequency domain, while point singularities at discontinuities are localized spatially. As a result, the 1-d wavelet transform of a piecewise-smooth signal is *sparse*: most of the signal energy is captured by a few large wavelet coefficients. Reconstructing the signal using only these few wavelet coefficients can provide a very accurate "nonlinear approximation" of the original signal.[7]

JPEG-2000 and most other wavelet-based image coders employ separable two-dimensional (2-d) filterbanks that are a simple extension of 1-d techniques. As with 1-d wavelets, the 2-d wavelet coefficients can be interpreted and modeled using a tree-structured space-frequency representation. The convenience of tree-structured modeling has been exploited in a variety of wavelet-based compression algorithms.[2,3,5] Despite the popularity of wavelet-based approaches to compression, however, 2-d wavelets fail to provide sparse representations for geometric regions, a very important class of image features.

Email: {wakin, jrom, choi, richb}@rice.edu. Web: dsp.rice.edu. This work was supported by the National Science Foundation grants CCR–9973188, ONR grant N00014-02-1-0353, AFOSR grant F49620-01-1-0378, and the Texas Instruments Leadership University Program.

1.2. The challenge of geometry

Natural images can be viewed as collections of *geometric*, *smooth*, and *textured* regions.* Geometric features, such as edges, generally indicate transitions between smooth or textured regions and are characterized by abrupt changes in intensity that persist along straight or curved contours. Edges communicate important information, conveying the location and shape of pictured objects and many of their features. In addition, because pixel values vary rapidly in the direction orthogonal to an edge, much of an image's high-frequency energy may come from edges. For these reasons, a successful algorithm for image compression must efficiently encode geometric features.

Unfortunately, the desirable nonlinear approximation performance of 1-d wavelets does not carry over to geometric 2-d image features. As with 1-d wavelets, 2-d wavelets do provide efficient approximations for smooth regions and point singularities. In the case of an edge, however, where a singularity extends along a contour, the number of 2-d wavelet basis functions overlapping the singularity grows exponentially at finer scales; many wavelet coefficients are required to reconstruct even a simple, straight edge.[8]

The abundance of significant coefficients describing geometry is not an immediate barrier to effective wavelet-domain image processing (including compression, in particular). There is, in fact, a strong *coherency* among the coefficients which is imposed by the structure of the geometry. In the case of an isolated, sharp edge, for example, the 1-d information describing the trace of the edge contour completely determines the values of the 2-d wavelet coefficients. In theory, this coherency could be accurately exploited in a wavelet-domain model. Due in part to a lack of shift-invariance for real 2-d wavelets, however, the coherency is quite difficult to model. Most wavelet-domain algorithms resort instead to modeling collective quantities such as the variance of the coefficients (see, for example, Ref. 4); of course, this simplification has a direct impact on compression performance. A simplified model not only affects rate-distortion (R-D) efficiency, but also leads to ringing artifacts when quantization disrupts the geometric coherency of the coefficients.

Faced with the challenges presented by geometric features, two clear options are available: the first is to develop a new transform that includes properties such as sparse representations for geometry; the second is to improve the wavelet-domain models and algorithms accordingly. We briefly review below recent work that has focused on pursuing each of these options. As the first option currently faces several practical difficulties, we pursue in this paper the second.

1.2.1. Option 1: Develop alternate transforms

Recent work in harmonic analysis has focused on developing transforms that provide sparse representations for certain geometric image classes. Candès and Donoho define curvelets,[8] a nonadaptive transform that provides nearly optimally sparse representations for piecewise-smooth images. The contourlets of Do *et al.*[9] comprise a filter-bank implementation of a similar transform. Although these representations overcome the poor nonlinear approximation performance of 2-d wavelets, it is not clear how to apply them to such tasks as image compression. One difficulty is redundancy: these overcomplete transforms produce a collection of coefficients that is larger than the number of pixels in the original image. In addition, these transforms are designed specifically to account for geometry; modeling their behavior in non-geometric regions may prove to be difficult.

Interestingly, decompositions using the dual-tree *complex* wavelet transform[10, 11] show much better properties regarding geometric analysis than do those using standard real wavelets. Complex wavelets exhibit improved shift invariance properties, and although the transform coefficients are not sparse in geometric regions, there exist distinct correlations between coefficient magnitudes and phases and edge orientation and position, respectively. These properties lead to improved geometric modeling,[12] but the dual-tree transform is four-times overcomplete.† Steerable pyramids[15] provide both shift invariance and rotational invariance but at the cost of an even greater degree of redundancy. Despite the analysis properties of each of these representations, their redundancy currently poses a barrier to efficient compression algorithms.

*We loosely define textured regions as those that do not fit into the other two categories, or which contain complicated combinations thereof.

†Nonredundant implementations have been developed for complex wavelets but are currently less amenable to geometric modeling.[13, 14]

1.2.2. Option 2: Improve wavelet-domain models

Additional work has recently focused on improving models and algorithms for wavelet-based image compression. Some approaches have attempted to encode geometric contour information separately from the remaining 2-d features. Bandelets,[16] for example, use 1-d wavelets to encode the 1-d contour information, and then warp a 2-d wavelet basis around that contour to capture the remaining information. In Ref. 17, Froment uses level lines to encode the geometry of an image and wavelet packets[18] to represent the remaining features. Shukla *et al.*[19] develop an efficient prune-join quadtree algorithm for compressing piecewise-smooth images with piecewise-smooth contours; such an approach could be combined with a second stage for compressing residual texture information. Although these approaches have achieved moderate success in modeling geometry, they generally do not provide a complete solution for natural image compression. Specifically, a complete image coder must successfully balance geometric descriptions with smooth and texture encodings. Few solutions have examined the R-D impacts of their classifications or considered the problem of optimally allocating bits among the different representations.

1.3. Contributions

In this paper, we propose a novel multiscale three-part model for 2-d wavelet coefficients that accounts for geometric, smooth, and textured image features. Tree-structured modeling provides a convenient framework for choosing among different encoding strategies and naturally leads to an algorithm for making optimal classifications. We demonstrate that a simple image coder based on our model can achieve near-optimal R-D performance for the class of piecewise-smooth images containing C^2 regions separated by C^2 discontinuities. In addition, we extend this concept to a more practical setting and present a complete compression algorithm for natural images that optimizes bit allocation among the three representations.

Section 2 describes our wavelet-domain models for smooth, textured, and geometric image regions. The smooth and textured models are motivated by the space-frequency quantization (SFQ) compression algorithm;[5] SFQ uses a mix of zerotrees (for smooth regions) and scalar quantization (for texture regions) to compress a quadtree wavelet decomposition and relies on an efficient tree-pruning algorithm to find the R-D optimal configuration of the two. We introduce a representation for wavelet coefficients in geometric regions based on the wedgelet dictionary,[20] a collection of geometric atoms that construct piecewise-linear approximations to contours. Our *wedgeprint representation* implicitly models the coherency among geometric wavelet coefficients. We analyze the potential R-D performance of an image coder combining wedgeprint representations with zerotrees and scalar quantization.

In Section 3, we enhance our geometric model with an emphasis on practical performance. We present a method for encoding wedgelets jointly, and we focus on the problem of determining the optimal wedgelet representation for an image region. The resulting wedgeprints are able to model more complicated instances of geometry.

In Section 4, we present our Wedgelet-SFQ (WSFQ) compression algorithm that uses an R-D criterion to optimize its its bit allocation among geometry, smooth, and texture representations. In Section 5, we present performance results for WSFQ. The compression gains we achieve over SFQ illustrate the true promise of geometric compression techniques. We conclude in Section 6 with a final discussion.

2. WAVELET-DOMAIN MODELS

2.1. Dyadic blocks, wavelets, and quadtrees

In this section, we develop wavelet-domain models for textured, smooth, and geometric image regions. Due to the dyadic nature of the wavelet decomposition, we find it convenient to organize our models in the context of 2-d wavelet quadtrees.[7] The 2-d wavelet transform is typically arranged in three subbands, corresponding to the vertical, horizontal, and diagonal orientations of the wavelet basis functions. Each subband can be organized into a quadtree, as described below.

In the quadtree interpretation of the 2-d wavelet transform, each node i is labeled with a wavelet coefficient w_i, where the corresponding wavelet basis function ψ_i has approximate support on a square, dyadic block B_i in the image. The width of this block is given by $M = 2^{-\ell}N$, where ℓ is the depth of node i in the quadtree, and N is the width in pixels of the square image (assumed to be a power of two).

Except at the finest level, each node has four children representing $\frac{M}{2} \times \frac{M}{2}$ dyadic blocks that combine to tile the same $M \times M$ image block as their parent. The set of the four children of node i is denoted C_i, and the subtree of all descendants of node i is denoted U_i (note that this does not include node i).

Each of the three directional subbands comprises a quadtree that spans the entire image. In particular, every dyadic block in the image has one corresponding node in *each* of the three quadtrees. In this paper, we encode each directional subband as its own independent quadtree.

2.2. Scalar quantization for textured regions

In this paper, we define "texture" somewhat loosely, and allow it to include any features that are not otherwise well characterized. Similarly, we place few specifics in the wavelet-domain texture model. Although we believe that more sophisticated models may be developed for particular kinds of texture, we find the generality of our model helpful for combining it with other models.

Suppose node i has support on a dyadic image block B_i that is characterized as texture. Then we assume the wavelet coefficient w_i has uniform probability distribution

$$p_{w_i}(w_i) \sim \mathcal{U}[-Z, Z]$$

for some known bound Z. We assume that this coefficient is statistically independent of neighboring coefficients.

Under this simple model, a uniform scalar quantizer provides a reasonably efficient and practical compression scheme for texture wavelet coefficients. We note here that such a coder requires bits to encode the quantization bin, and results in a distortion related to the quantization step-size.

2.3. Zerotrees for smooth regions

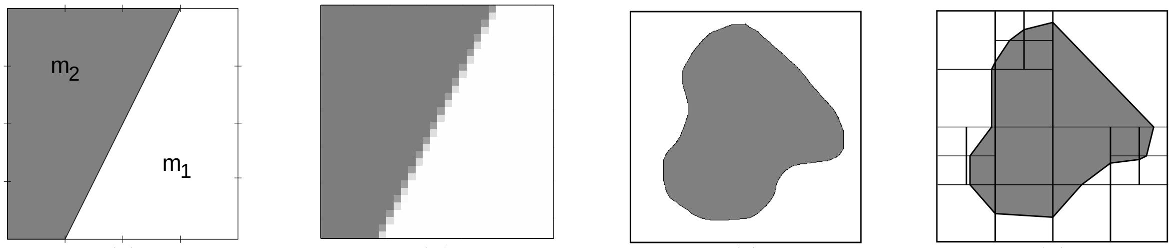
Several mathematical characterizations exist for the notion of smoothness; in general such definitions involving wavelets refer to the decay rate of wavelet coefficients through scale.[7] In this paper, we refrain from adopting a rigid definition for "smoothness," but intuitively we expect smooth image regions to result in small wavelet coefficients.

Suppose node i has support on a dyadic image block B_i that is characterized as smooth. Because all nodes descending from i can also be characterized as smooth, our model simply assumes that $w_i = 0$ and that $w_j = 0,\ \forall j \in U_i$. This model does not incorporate probability; it is simply a fixed approximation to the wavelet coefficients in smooth regions. This tree-structured approximation is popularly known as a *zerotree*.[2] Encoding a zerotree requires very few bits and results in distortion equal to the energy of the approximated coefficients.

A combination of zerotrees and scalar quantization is sufficient to implement an image coder. Indeed, these are essentially the two ingredients of Shapiro's EZW coder.[2] In our terminology, the EZW coder classifies as smooth any image regions where all wavelet coefficients fall below some threshold. Other image regions (including those containing isolated instances of large wavelet coefficients) are classified as texture.

More recently, the space-frequency quantization (SFQ) coder[5] employs the same two encoding strategies but with a more refined method of classification. In SFQ, classification into smooth and textured regions is performed using a bottom-up tree-pruning algorithm. This tree-pruning algorithm finds the R-D near-optimal configuration of zerotrees. A standard SFQ optimization generally results in the use of zerotree symbols to represent smooth regions and low-energy features, with scalar quantization used to code high-energy features such as edges.

Despite its success — SFQ outperforms JPEG-2000 at most rates — the SFQ coder fails to model the joint behavior of wavelet coefficients along an edge. The model of independently distributed wavelet coefficients that is used within geometric regions clearly neglects the inherent structure in such regions. In the following sections, we develop a model that captures the joint behavior of geometric wavelet coefficients and explain how this model can be integrated into an R-D optimized compression algorithm such as SFQ.



(a) (b) (c) (d)

Figure 1. *(a) Wedgelet parameterization on an $M \times M$ dyadic block: a position index k describes the endpoints of the edge, and m_1 and m_2 specify the grayscale intensities on each side. (b) Picture of a wedgelet, $M = 32$. (c) Example of a simple "cartoon image." (d) A wedgelet decomposition divides the domain of an image into dyadic squares, using a piecewise-constant function in each square to approximate the image.*

2.4. Wedgeprints for geometric regions

We begin by modeling the simplest possible kind of geometry: an isolated, straight, sharp edge. The *wedgelet dictionary*, introduced by Donoho,[20] provides a convenient framework for developing our model. A *wedgelet* $\mathcal{W}(B; k, m_1, m_2)$ is a dyadic block B containing a picture of a single straight edge. As shown in Fig. 1(a,b), the edge separates two constant regions of grayscale intensity m_1 and m_2; pixel values along the edge are computed by an appropriate weighted averaging. We restrict the allowable endpoints of the edge so that its position may be indexed by a single discrete parameter $k \in \{1, 2, \ldots, K\}$. The wedgelet dictionary is the collection of all possible wedgelets on all dyadic blocks. The task of organizing a convenient dictionary is discussed in Ref. 21.

In our simplified task of modeling straight, sharp edges, a typical instance of "geometry" on a dyadic image block would resemble a wedgelet in the spatial domain. Due to the relatively localized support of the wavelet basis functions, we can obtain a reasonable approximation to geometric wavelet coefficients by taking the wavelet transform of a wedgelet. To be more precise, assume node i has been classified as geometry. After choosing the appropriate wedgelet parameters, we construct a *wedgeprint* for node i by projecting the wedgelet $\mathcal{W}(B_i; k, m_1, m_2)$ onto the subspace spanned by the basis functions ψ_i and $\{\psi_j : j \in U_i\}$. Each wedgelet, therefore, may generate up to three distinct wedgeprints (one in each directional subband); as discussed in Sec. 2.1, we consider each one individually.

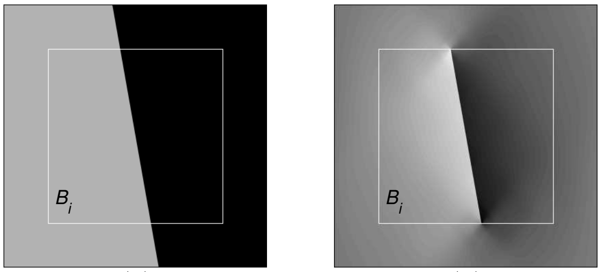
Wedgeprints can be used as a tool for representing groups of wavelet coefficients; the single wedgeprint described above approximates coefficient w_i and *all* of its descendants. In addition, wedgeprints implicitly model the geometric coherency among wavelet coefficients. As shown in Fig. 2, wedgeprint representations leave few ringing artifacts around the approximated edges. Through its underlying wedgelet parameters, the wedgeprint representation can be adapted to local geometry. To use a wedgeprint for compression, only the wedgelet parameters $\{k, m_1, m_2\}$ must be encoded; the decoder can use these to reconstruct the entire subtree of wavelet coefficients. Wedgeprints are similar to zerotrees — large numbers of wavelet coefficients are efficiently encoded with few parameters — but wedgeprints do so in the high-energy regions near edges.

The wedgeprint construction is similar to the approach by Dragotti *et al.* in developing the footprints dictionary,[22] a collection of scale space vectors that model wavelet coefficients at singularities in 1-d piecewise-polynomial signals. Our 2-d implementation is different, however, from the edgeprints presented in Ref. 23 where footprints are applied separately to the rows and columns of a 2-d image.

Wedgeprints can also be used to model more sophisticated instances of geometry. Contours in an image can be approximated by a *wedgelet decomposition*, a tiling of wedgelets chosen from the wedgelet dictionary (see Fig. 1(c,d)). In Sec. 3, we discuss the possibility of *jointly* encoding wedgelet parameters. In Sec. 4, we construct more descriptive wedgeprints by projecting entire wedgelet decompositions to the wavelet domain.

2.5. Theoretical compression performance

By adding the wedgeprint representation to zerotree and scalar quantization techniques, we significantly improve the potential performance of an image coder. As an illustration, we consider the problem of encoding synthetic



(a) (b)

Figure 2. *(a) Portion of an image containing a wedgelet through the dyadic block B_i. (b) Spatial domain picture of the wedgeprint on B_i constructed by keeping only the wavelet coefficients on the vertical-band subtree rooted at i.*

continuous images of type "C^2/C^2." The class C^2/C^2 contains images $X(t_1, t_2)$ defined on the unit square $[0, 1]^2$ that can be constructed as follows

$$X(t_1, t_2) = X_1(t_1, t_2) \cdot H_c(t_1, t_2) + X_2(t_1, t_2) \cdot (1 - H_c(t_1, t_2)),$$

where the components X_1, X_2 are smooth

$$X_1(t_1, t_2), X_2(t_1, t_2) \in C^2([0, 1]^2),$$

and H_c is a *Horizon-class image*[20] with a smooth discontinuity

$$H_c(t_1, t_2) \;= \begin{cases} 1, & t_2 > c(t_1) \\ 0, & t_2 \le c(t_1), \end{cases} \qquad c(t_1) \in C^2([0, 1]).$$

In words, C^2/C^2 images contain two smooth regions separated by a smooth discontinuity.

Clearly, such images are simple to describe; only one smooth 1-d function and two smooth 2-d functions are needed to fully represent an image. An oracle coder could use 1-d wavelets to encode the function c, and 2-d wavelets to encode the images X_1 and X_2. From the analysis presented in Refs. 24, 25, it follows that an oracle coder could achieve asymptotic R-D behavior $D(R) \sim R^{-2}$.

In practice, of course, image coders must function without the oracle information. As discussed in Sec. 1.2, most current wavelet-based image coders would fail to exploit the simplicity of these images, spending too many bits to encode the smooth edge discontinuity. In Ref. 25, we demonstrate that, using a proper arrangement of scalar quantizations, zerotrees, and wedgeprints, a simple image coder can achieve the asymptotically near-optimal rate distortion performance $D(R) \sim (\log R)^2/R^2$. In this arrangement, a series of wedgeprints is used to construct a piecewise-linear approximation to the discontinuity. Scalar quantization is used for wavelet coefficients at coarse scales, while zerotrees are used for the fine-scale wavelet coefficients in smooth regions.

This result illustrates the potential effectiveness of wedgeprint representations when combined with the appropriate compression techniques for smooth and textured regions. A few practical issues must be addressed, however, before transcribing these ideas to a natural image coder. First, we prefer to more fully develop our geometric model. In particular, we believe that some practical (though perhaps not asymptotically significant) gains can be made by jointly encoding wedgelet parameters. These refinements are discussed in Sec. 3. Second, our result in Ref. 25 establishes only the existence of an efficient representation. The encoder must have an effective method for finding the proper balance of scalar quantization, zerotrees, and wedgeprints. Building upon the tree-pruning algorithm of the SFQ coder, we present in Sec. 4 our Wedgelet-SFQ (WSFQ) coder. The WSFQ algorithm uses a tree-pruning R-D optimization to find the proper balance among the three options.

3. ENHANCING GEOMETRIC MODELS

Consider a node i in the wavelet quadtree. In Sec. 2.4, we obtained a wedgeprint for the wavelet coefficient subtree rooted at i by constructing a single wedgelet on the dyadic block B_i and projecting to the wavelet

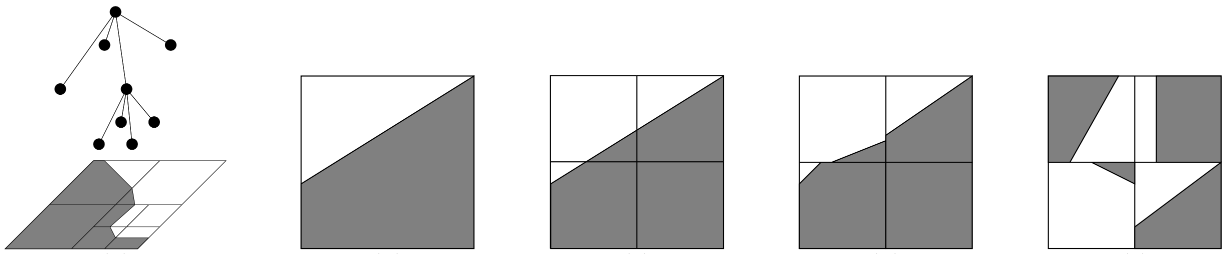

(a) (b) (c) (d) (e)

Figure 3. *(a) A wedgelet decomposition can be interpreted as a pruned quadtree, where each node includes a set of wedgelet parameters and leaf nodes specify the pictured wedgelets. (b) Wedgelet on a dyadic block. (c) Predictions for the block's children, considered to be their most likely configuration. (d) A slightly less likely configuration for the children. (e) A significantly less likely configuration.*

domain. A single wedgelet is not the only available description for the geometry on B_i, however. In particular, a *wedgelet decomposition* could also be used, allowing a tiling of wedgelets to more precisely describe the geometry on B_i. Projecting such a description to the wavelet domain, we would achieve a more precise approximation to the wavelet coefficients descending from i. Such an approach may be beneficial if an efficient method can be developed for encoding a wedgelet decomposition.

In this section, we first extend our wedgelet dictionary to parameterize the smoothness of the edge profile. We then present a method for jointly encoding the wedgelets in a wedgelet decomposition. We also discuss a technique for finding the R-D optimal wedgelet decomposition on a particular dyadic block. Our WSFQ coder, explained in Sec. 4, uses the wedgeprints that result from projecting these optimized local wedgelet decompositions into the wavelet domain.

3.1. Parameterizing edge profile smoothness

To allow for more flexible wedgelet descriptions, we include a notion of smoothness across the profile of the edge. A sharp edge, for example, makes an abrupt step-edge transition from m_1 to m_2 in the profile (see Fig. 1(a)). Alternatively, a smooth (or blurred) edge may take several pixels to transition from m_1 to m_2. There are many possible methods for parameterizing this smoothness. For the purposes of this paper, we will simply include a parameter s that specifies the smoothness of the pictured wedgelet. In practice, we first construct a picture of a sharp wedgelet and then apply a blurring filter specified by s.

3.2. Jointly encoding wedgelet decompositions

As shown in Fig. 3(a), a dyadic wedgelet decomposition can be interpreted as a pruned quadtree, where each node i includes a set of wedgelet parameters Θ_i describing the corresponding dyadic block B_i. In such a quadtree, each node includes a map symbol $\eta_i \in \{L, I\}$ indicating whether the node is a leaf or interior node. Leaf nodes are used to assemble the picture of the wedgelet decomposition, while interior nodes are useful for predicting and encoding parameters at the leaf nodes. Finer approximations to a contour can be obtained by dividing a leaf node into four children.

We implement a top-down, predictive scheme for encoding a wedgelet decomposition, using a simple Markov-1 model that captures the dependency of a wedgelet orientation on the wedgelet encoded at its parent node (see Figs. 3(b)-(e)). Our algorithm exploits the redundancy among wedgelet parameters that results because a node and its four children describe the same spatial location. Thus, once a wedgelet has been encoded for a node, we can obtain predictions for the wedgelets at its four children by drawing a picture of the wedgelet and dividing the picture into four quadrants. In practice, this is accomplished with a simple lookup when using the dictionary in Ref. 21. We encode the actual children wedgelets according to a conditional probability model on the Hausdorff distance $\delta(k, \widehat{k})$ between the true and predicted wedgelets. The complete algorithm for encoding a wedgelet decomposition follows.

Step 1. Encode $\{m_1, m_2, s\}$, which are assumed constant for the entire quadtree.

Step 2. Let i be the root node of the quadtree. Encode the wedgelet index k_i and map symbol η_i. If $\eta_i = L$, then terminate. Otherwise, let $\ell = 0$.

Step 3. For every node α at level ℓ such that $\eta_\alpha = I$, perform the following for each $j \in C_\alpha$:
1. Predict $\widehat{k_j}$ from k_α.
2. Encode the wedgelet index k_j given its prediction $\widehat{k_j}$ using entropy coding according to the distribution

$$p(k_j|\widehat{k_j}) \sim e^{-\gamma\delta(k_j,\widehat{k_j})}.$$

3. Encode η_j. If $\eta_j = L$, then assume $\eta_x = L$ for all $x \in U_j$.

Step 4. Increment $\ell \leftarrow \ell + 1$. Repeat Step 3 until ℓ exceeds the depth of the quadtree.

In Step 3.2, $\gamma > 0$ is a constant that controls the preference given to accurate predictions. According to this distribution, accurate predictions require few bits to encode, while large prediction errors are more costly. As a result, the R-D performance of this predictive coder depends on the *regularity* of the approximated contour. This intuitively satisfying behavior is not demonstrated by most 2-d wavelet-based coders.

3.3. Optimizing the wedgelet decomposition

We use a Viterbi-like algorithm to find a near-optimal wedgelet decomposition, under an R-D criterion. Letting R and D be the total rate and squared-error distortion incurred by encoding a wedgelet decomposition, respectively, we fix a Lagrangian parameter λ and seek the wedgelet decomposition minimizing $D + \lambda R$. The optimization algorithm iterates between two stages: tree-pruning and parameter estimation. The tree-pruning stage operates similar to Donoho's dynamic programming method for complexity-penalized tree-pruning,[20] using a bottom-up technique for determining leaf nodes. In addition, however, our method chooses the R-D optimal wedgelet index at each node. The parameter estimation stage uses the geometry of the pruned tree to obtain new estimates for $\{m_1, m_2, s\}$.

4. WEDGELET-SFQ FOR NATURAL IMAGE COMPRESSION

We now apply all of the above ideas and develop Wedgelet-SFQ (WSFQ), an algorithm for natural image compression that combines the two-class SFQ strategy with a representation and model for geometric compression. WSFQ incorporates scalar quantization, zerotrees, and wedgeprints, and uses a bottom-up tree-pruning to find the R-D near-optimal balance among the three options. Many of the implementation details of WSFQ follow naturally from the SFQ algorithm; Ref. 5 contains a detailed explanation of the SFQ coder. We present in this section the relevant details of the WSFQ coding algorithm and optimization scheme; performance results are presented in the following section.

4.1. WSFQ quantization strategies

After determining the proper classification of image regions into smooth, texture, and geometry, WSFQ encodes each of the three directional wavelet quadtrees in a single pass from the top down.[‡] For each node i, WSFQ encodes a map symbol $n_i \in \{\mathrm{S}, \mathrm{T}, \mathrm{G}\}$ indicating the quantization strategy for descendants of that node. The quantization scheme for a given wavelet coefficient is actually specified by one of its ancestors, a small deviation from the conventions used in previous sections of this paper.

For smooth regions, symbol $n_i = \mathrm{S}$ indicates *zerotree quantization*. Under zerotree quantization, all descendants U_i are quantized to zero. No further information (including map symbols) is encoded for these nodes.

For textured regions, symbol $n_i = \mathrm{T}$ indicates *scalar quantization*. This symbol specifies that the four children C_i are *significant*: a scalar quantization bin is encoded for each. All significant wavelet coefficients are quantized uniformly with a common uniform scalar quantizer; the quantization step-size q is optimized for the target rate. The symbol $n_i = \mathrm{T}$ dictates the quantization strategy *only* for nodes in C_i; an additional map symbol is encoded at each child to describe the quantization of its descendants.

[‡]Scaling coefficients are coded separately; our approach is mentioned in Section 5, but the particular details are not relevant to the WSFQ algorithm.

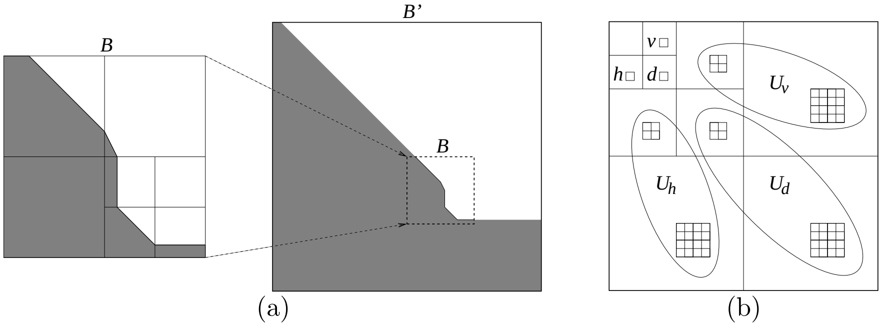


Figure 4. *Obtaining a wedgeprint. (a) After a wedgelet decomposition is encoded for a dyadic block B, we create a larger block B', using linear extensions for the edges at the border of B. (b) Taking the wavelet transform of B', we may extract any of the subtrees of wavelet coefficients U_v, U_d, U_h, where nodes v, d, h have support on the block B.*

For geometric regions, symbol n_i = G indicates the use of a *wedgeprint*. In this situation, the encoder constructs and encodes a wedgelet decomposition for the block B_i, and the resulting wedgeprint is used to infer the descendant wavelet coefficients on the subtree. For each $j \in U_i$, we denote the implied wedgeprint coefficient as $w^*_{i,j}$. The wedgelet decomposition is constructed and encoded using the techniques of Sec. 3. For our purposes, it is not necessary to encode both m_1 and m_2 explicitly; the contrast $m_2 - m_1$ alone will suffice. We denote by $R_{\mathcal{W}_i}$ the rate required to encode the wedgelet decomposition at node i.[§] To obtain the wavelet transform of the wedgelet decomposition while minimizing border artifacts, we create a larger temporary block containing the coded wedgelet decomposition at the appropriate location, take its wavelet transform, and extract the appropriate wavelet coefficients. This process is illustrated in Fig. 4.[¶]

4.2. WSFQ tree-pruning

To obtain a near-optimal configuration of the three quantization symbols, we use a generalization of the SFQ tree-pruning algorithm. Optimization in WSFQ is performed in two stages: Phase I iteratively prunes the tree based roughly on the rate and distortion costs of quantization, while Phase II adjusts the configuration to account for the rate cost of encoding the map symbols.

4.2.1. Phase I: Quantization costs

Phase I tree-pruning starts at the bottom of the tree and proceeds upward. In the beginning, it is assumed that all coefficients are scalar quantized, and decisions must be made regarding whether to use zerotree or wedgeprint representations. The coder uses several bottom-up iterations until the tree-pruning converges. At the beginning of each iteration, the coder estimates the probability density p of the collection of significant coefficients; this yields an estimate of the entropy (and hence coding cost) of each scalar quantization. Ultimately, we use adaptive arithmetic coding[26] to encode these quantization bin indices.

During each iteration of the Phase I optimization, only nodes currently labeled significant are examined. The coder has three options at each such node i: create a zerotree (symbol n_i = S), maintain the significance (symbol n_i = T), or create a wedgeprint (symbol n_i = G). Each option requires a certain number of bits for quantization and results in a certain distortion relative to the true wavelet coefficients; the coder chooses the option that minimizes the total R-D impact on the subtree descending from node i. The first option, zerotree quantization of the subtree of descendants, requires $R_i^{(\mathrm{S})} = 0$ bits, because no information is encoded besides the map symbol. By quantizing all coefficients in U_i to zero, this option results in distortion

$$D_i^{(\mathrm{S})} = \sum_{j \in U_i} {w_j}^2.$$

[§]Up to three subbands may request a wedgelet decomposition for the same dyadic block. In such a case, it is encoded only once, but for simplicity, we do not reduce the anticipated cost $R_{\mathcal{W}_i}$.

[¶]Because the wavelet basis functions are not perfectly localized, this process does introduce a small amount of error that was not considered in pruning the wedgelet decomposition.

The second option is to send a significance symbol for n_i as well as the quantization bins for $\{w_j : j \in C_i\}$. For this option, we must consider the (previously determined) rate and distortion costs of nodes in C_i as well. Letting $\widehat{w_j}$ denote a wavelet coefficient quantized by step-size q, we have

$$R_i^{(\mathrm{T})} = \sum_{j \in C_i} -\log_2 [p(\widehat{w_j})] + \sum_{j \in C_i} R_j.$$

This option results in distortion

$$D_i^{(\mathrm{T})} = \sum_{j \in C_i} (w_j - \widehat{w_j})^2 + \sum_{j \in C_i} D_j.$$

The third option is to send a wedgeprint symbol for n_i, encode a wedgelet decomposition, and use the corresponding wavelet coefficients as the quantized values for descendants of node i. Unlike the cases $n_i = \mathrm{S}$ or T, which we expect to be relatively common, we expect relatively few wedgeprint symbols to be encoded, since each one represents many possibly significant coefficients. Each wedgeprint map symbol therefore requires a nontrivial amount of added bit rate to encode. We find it useful, then, to consider a rough estimate ρ_{G} of the probability of sending symbol G. Choosing a suitably low value for ρ_{G}, it follows that the Phase I rate cost for the wedgeprint option is given by

$$R_i^{(\mathrm{G})} = -\log_2 [\rho_{\mathrm{G}}] + R_{\mathcal{W}_i}$$

and the resulting distortion is simply

$$D_i^{(\mathrm{G})} = \sum_{j \in U_i} (w_j - w_{i,j}^*)^2.$$

The decision between the three options is made to minimize the Lagrangian cost $J_i = D_i + \lambda R_i$, where λ is an optimization parameter controlling the trade-off between rate and distortion (the same value of λ is used for pruning the wedgelet decompositions for wedgeprints).

After optimizing the symbol n_i, the tree-pruning proceeds upward. Once the top of the tree is reached, the process repeats from the bottom-up if any significant map symbols have changed. Convergence is guaranteed because the number of significant coefficients can only decrease. The Phase I tree-pruning algorithm converges to a near-optimal configuration of map symbols (see Sec. 5 for performance results). As discussed in Ref. 5, the constantly changing distribution of significant coefficients affects R-D costs everywhere and may prevent a bottom-up tree-pruning algorithm from finding the globally optimal solution.

4.2.2. Phase II: Map symbol costs

Phase II adjusts the tree-pruning to better account for the costs of encoding map symbols.[‖] These costs are obtained by considering how the symbols will be encoded. Specifically, in the top-down encoding of the quadtree, map symbols are predicted based on the variance of local, causal quantized wavelet coefficients. Low variances indicate the likelihood of symbol S, while high variances indicate the likelihood of symbols T and G. We encode whether a particular symbol is S according to this expected behavior, and we then distinguish between symbols T and G using adaptive arithmetic coding. Phase II adjusts the tree-pruning to better account for the *first* of these costs, scanning the quadtree to determine if any nodes should be changed to (or from) symbol S. A switch is made if the savings in map symbol rate exceeds the loss in Phase I R-D efficiency.

4.3. Residual compression

When a wedgeprint is used to represent a wavelet subtree, it provides an approximate description of the local image geometry. This approximation produces errors, however, and as a matter of practicality, we wish to correct as many errors as efficiently possible.[**] As an attempt at correcting geometric errors, we implement standard SFQ compression on each residual subtree resulting from a wedgeprint. Thus, encoding a symbol $n_i = \mathrm{G}$ involves the following additional steps:

Step 1. Encode the wedgelet decomposition and compute the wedgeprint coefficients $\{w_{i,j}^* : j \in U_i\}$.

‖Our Phase II implementation closely follows that in Ref. 5.
**Correcting these errors was not necessary to obtain the near-optimal asymptotic R-D performance in Ref. 25.

(a) (b)

Figure 5. *(a) Synthetic test image, and (b) portion of* Peppers *test image used in compression experiments.*

Step 2. Compute the residual error subtree $E_i = \{e_{i,j} : j \in U_i\}$ where $e_{i,j} = w_j - w^*_{i,j}$.

Step 3. Prune and encode the subtree E_i using symbols S and T of SFQ, with the same parameter λ, quantization step-size q, and probability model p used elsewhere.

Step 4. Add the quantized residual $\widetilde{E_i}$ to the wedgeprint coefficients for the final encoded values.

The complete R-D cost for this procedure is computed before determining the symbol n_i.

SFQ enables a spatially adaptive approach for encoding the residual subtree. The zerotree symbol gives the residual coder the option of ignoring geometric artifacts that it cannot efficiently compress. Due to the flexibility of SFQ, some textures away from the edges in B_i may still be encoded.

5. RESULTS

We implemented both SFQ and WSFQ using MATLAB. For each image, we performed a 4-level wavelet decomposition using biorthogonal 10-18 wavelets.[27] SFQ and WSFQ provide methods for encoding the three directional wavelet coefficient subtrees; any efficient technique may be used to compress the scaling coefficients separately. For both SFQ and WSFQ, we compressed the scaling coefficients in a raster scan, predicting each coefficient from its quantized causal neighbors. The prediction errors were quantized and encoded, with quantization optimized for a generalized Gaussian distribution. For comparison purposes, we also compressed images using the wavelet-based JPEG-2000 coder.[28]

5.1. Artificial images

For an example of the effectiveness of wedgeprint representations, we first constructed a synthetic 256×256 image consisting of a sharp Horizon-class image with added texture (see Fig. 5(a)). In Fig. 6, we compressed this image using both SFQ and WSFQ at a rate of 0.098 bits per pixel (bpp). For a point of reference, JPEG-2000 compression yielded a PSNR of 30.89dB at this bit rate.[††] Fig. 6(a) shows the SFQ-compressed image. SFQ compression yielded a PSNR of 32.84dB, and the SFQ tree-pruning left a total of 1948 wavelet coefficients described by scalar quantization. The tree-pruned wavelet-domain segmentation is shown in Fig. 6(b). As expected, many of the significant coefficients occurred along the edge.

At the same bit rate, Fig. 6(c) shows the synthetic image compressed using WSFQ. A PSNR of 34.19dB was attained, an improvement of 1.35dB over the standard SFQ technique and 3.30dB over JPEG-2000. Fig. 6(d) shows the tree-pruned WSFQ segmentation. In regions described by wedgeprints, ringing artifacts were noticeably reduced compared to the SFQ result. In this case, 16 distinct wedgelet decompositions were encoded for

[††]Peak Signal-to-Noise Ratio (PSNR) is a commonly used measure of distortion; assuming a maximum possible intensity of 255, $\mathrm{PSNR} = 10\log_{10}\frac{255^2}{\mathrm{MSE}}$.

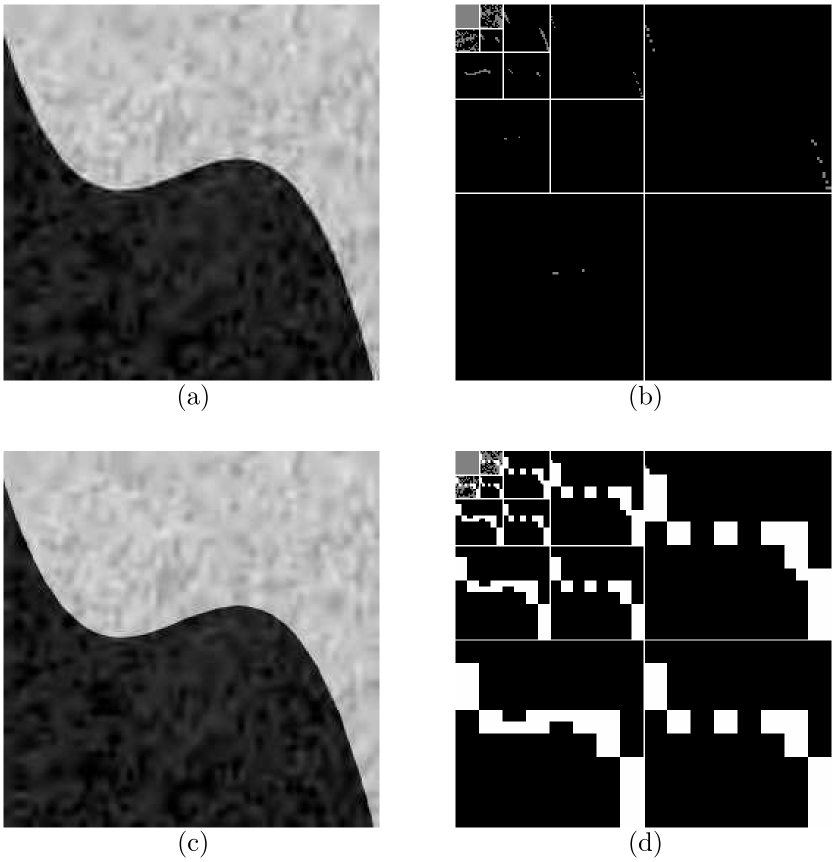


Figure 6. *(a) Synthetic image coded using SFQ. (b) Multiscale wavelet-domain segmentation from SFQ tree-pruning. Zerotrees are represented in black; significant coefficients are gray. (c) Image coded using WSFQ. (d) WSFQ segmentation. Wedgeprints are represented in white; note that the three subbands used slightly different wedgeprint configurations.*

wedgeprints, leaving 1544 wavelet coefficients described by scalar quantization. The compression for wedgeprint prediction errors encoded 244 residual coefficients using scalar quantization.

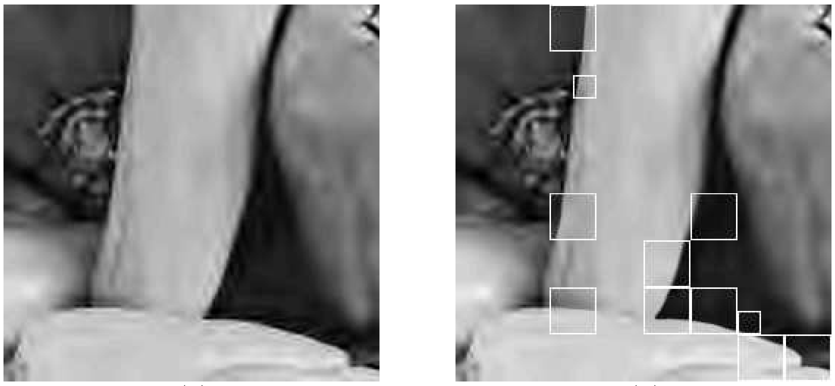
As expected, wedgeprints offered efficient representations for this synthetic image, which contained a strong, sharp edge that was easily modeled with wedgelet decompositions. Several parameters in this example affected the compression performance relative to SFQ. By adjusting the energy of the texture, for example, we could achieve gains up to several dB above standard SFQ. These variations on the synthetic image reflect the potential features encountered in natural images.

5.2. Natural images

As an example using a natural image, we compressed the 512×512 *Peppers* image using both SFQ and WSFQ at a bit rate of 0.07bpp. For a point of reference, JPEG-2000 compression yielded a PSNR of 28.57dB at this bit rate. Fig. 7(a) shows a portion of the SFQ-compressed image (see Fig. 5(b) for the original version). SFQ compression yielded a PSNR of 29.08dB, and the SFQ tree-pruning left a total of 5512 wavelet coefficients described by scalar quantization.

At the same bit rate, Fig. 7(b) shows the *Peppers* test image compressed using WSFQ. A PSNR of 29.25dB was attained, an improvement of 0.17dB over the standard SFQ technique and 0.68dB over JPEG-2000. In regions described by wedgeprints, ringing artifacts were noticeably reduced compared to the SFQ result. In this case, 44 separate wedgeprints were encoded, leaving 4436 wavelet coefficients described by scalar quantization. The compression for wedgeprint prediction errors encoded 160 residual coefficients using scalar quantization.

Fig. 8(a) shows the SFQ and WSFQ performance as the target bit rate was increased. Both algorithms outperformed JPEG-2000 at most rates. At higher rates, however, we see that WSFQ had diminishing performance



(a) (b)

Figure 7. *Portion of* Peppers *test image coded using (a) SFQ and (b) WSFQ. A white box indicates a dyadic block described by a wedgeprint in one or more subbands.*

gains relative to SFQ. This behavior has several possible causes. At higher rates, for example, it is essential to accurately encode textures very near to edges. In addition, wedgelet decompositions may not offer high enough precision to code natural instances of geometry at high rates. Such factors can be incorporated into future implementations of WSFQ.

Tests on other natural images performed similarly at a variety of bit rates; for images such as *Cameraman* with isolated, sharp edges, WSFQ performed up to 0.30dB better than SFQ (see Fig. 8(b)). For images such as *Lenna* that contain smoother edges with surrounding textures, WSFQ gains were more modest, around 0.05dB relative to SFQ.

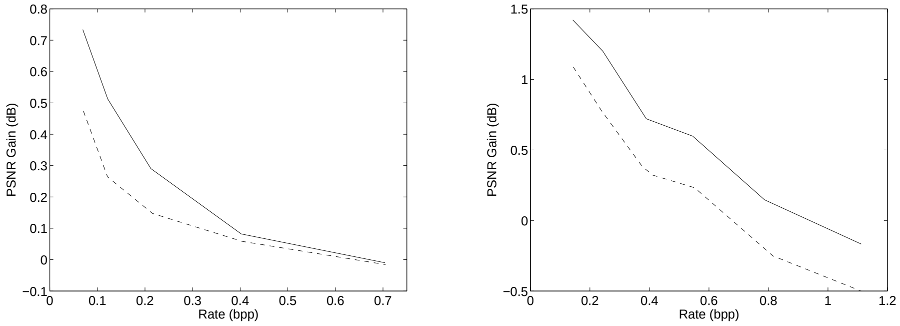
6. CONCLUSIONS

A compression approach involving explicit geometric descriptions appeals to the notion that such descriptions are meaningful (by providing an understanding of objects in the scene), useful (by capturing significant high-frequency energy), and easy to compress. In this paper, we have taken a careful approach in developing a complete coder, using an R-D optimized framework to balance geometric descriptions against smooth and texture representations. A key step in developing the WSFQ algorithm was a constructing a geometric model that interfaces naturally with existing wavelet-domain models.

Despite our somewhat modest approach, restricting geometric descriptions to isolated contours within dyadic blocks, the WSFQ coder outperforms the current state-of-the-art wavelet-based algorithms at most rates. This demonstrates the true potential of geometric image compression. Several current topics of research may lead to more significant breakthroughs in compression performance for natural images; the best solution may lie in some combination of better geometric models, alternative compression frameworks, and new harmonic bases motivated by geometry.

REFERENCES

1. J. K. Romberg, H. Choi, and R. G. Baraniuk, "Bayesian tree-structured image modeling using wavelet domain hidden Markov models," *IEEE Trans. Image Processing* **10**, July 2001.
2. J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing* **41**, pp. 3445–3462, Dec. 1993.
3. A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.* **6**, pp. 243–250, June 1996.
4. S. LoPresto, K. Ramchandran, and M. T. Orchard, "Image coding based on mixture modeling of wavelet coefficients and a fast estimation-quantization framework," in *Proceedings, IEEE Data Compression Conference — DCC '97*, pp. 221–230, (Snowbird, Utah), March 1997.
5. Z. Xiong, K. Ramchandran, and M. T. Orchard, "Space-frequency quantization for wavelet image coding," *IEEE Trans. Image Processing* **6**(5), pp. 677–693, 1997.
6. D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice*, Kluwer, Boston, 2002.



(a) (b)

Figure 8. *Performance improvement of WSFQ (solid) and SFQ (dashed) relative to JPEG-2000 compression for (a)* Peppers *image and (b)* Cameraman *image.*

7. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, second ed., 1999.
8. E. J. Candès and D. L. Donoho, "Curvelets — A suprisingly effective nonadaptive representation for objects with edges," in *Curve and Surface Fitting*, A. Cohen, C. Rabut, and L. L. Schumaker, eds., Vanderbilt University Press, 1999.
9. M. N. Do and M. Vetterli, "Contourlets: A directional multiresolution image representation," in *IEEE Int. Conf. on Image Proc. — ICIP '02*, (Rochester, New York), Oct. 2002.
10. N. Kingsbury, "Image processing with complex wavelets," *Phil. Trans. R. Soc. Lond. A* **357**, pp. 2543–2560, September 1999.
11. I. W. Selesnick, "The design of approximate Hilbert transform pairs of wavelet bases," *IEEE Trans. Signal Processing* **50**, May 2002.
12. J. K. Romberg, M. B. Wakin, H. Choi, and R. G. Baraniuk, "A geometric hidden Markov tree wavelet model," in *Proc. SPIE's 48th Ann. Mtg., Int. Sym. on Optical Sci. and Tech.*, (San Diego), 2003.
13. F. C. A. Fernandes, R. L. C. van Spaendonck, and C. S. Burrus, "A new framework for complex wavelet transforms," *IEEE Trans. Signal Processing* **51**, July 2003.
14. R. L. van Spaendonck, T. Blu, R. G. Baraniuk, and M. Vetterli, "Orthogonal Hilbert transform filter banks and wavelets," in *Proc., IEEE Int. Conf. Acoust., Speech, Signal Proc. — ICASSP '03*, (Hong Kong), April 2003.
15. E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multi-scale transforms," *IEEE Trans. Inform. Theory* **38**, March 1992.
16. E. L. Pennec and S. Mallat, "Image compression with geometrical wavelets," in *IEEE Int. Conf. on Image Proc. — ICIP '01*, (Thessaloniki, Greece), Oct. 2001.
17. J. Froment, "Image compression through level lines and wavelet packets," in *Wavelets in Signal and Image Analysis*, A. A. Petrosian and F. G. Meyer, eds., Kluwer, 2001.
18. F. G. Meyer, A. Z. Averbuch, and J. Strömberg, "Fast adaptive wavelet packet image compression," *IEEE Trans. Image Processing* **9**, May 2000.
19. R. Shukla, P. L. Dragotti, M. Do, and M. Vetterli, "Rate distortion optimized tree structured compression algorithms," *IEEE Trans. Image Processing* , submitted 2003.
20. D. L. Donoho, "Wedgelets: Nearly-minimax estimation of edges," *Annals of Stat.* **27**, pp. 859–897, 1999.
21. J. K. Romberg, M. B. Wakin, and R. G. Baraniuk, "Multiscale wedgelet image analysis: Fast decompositions and modeling," in *IEEE Int. Conf. on Image Proc. — ICIP '02*, (Rochester, New York), 2002.
22. P. L. Dragotti and M. Vetterli, "Wavelet footprints: Theory, algorithms and applications," *IEEE Trans. Signal Processing* **51**, May 2003.
23. P. L. Dragotti and M. Vetterli, "Footprints and edgeprints for image denoising and compression," in *IEEE Int. Conf. on Image Proc. — ICIP '01*, (Thessaloniki, Greece), Oct. 2001.
24. M. N. Do, P. L. Dragotti, R. Shukla, and M. Vetterli, "On the compression of two-dimensional piecewise smooth functions," in *IEEE Int. Conf. on Image Proc. — ICIP '01*, (Thessaloniki, Greece), Oct. 2001.
25. J. K. Romberg, M. B. Wakin, and R. G. Baraniuk, "Approximation and compression of piecewise smooth images using a wavelet/wedgelet geometric model," in *IEEE Int. Conf. on Image Proc. — ICIP '03*, (Barcelona), 2003.
26. I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM* **30**, pp. 520–540, June 1987.
27. M. Tsai, J. Villasenor, and F. Chen, "Stack-run image coding," *IEEE Trans. Circuits Syst. Video Technol.* **6**, Oct. 1996.
28. M. D. Adams, *The JasPer Project home page.* www.ece.uvic.ca/$\sim$mdadams/jasper/.