# Image Compression using an Efficient Edge Cartoon + Texture Model

*Michael Wakin, Justin Romberg, Hyeokho Choi, and Richard Baraniuk* *

Dept. of Electrical and Computer Engineering, Rice University
6100 S. Main St., Houston, TX 77005
{wakin, jrom, choi, richb}@rice.edu
dsp.rice.edu

**Abstract**

Wavelet-based image coders optimally represent smooth regions and isolated point singularities. However, wavelet coders are less adept at representing perceptually important edge singularities, and coding performance suffers significantly as a result. In this paper, we propose a novel two-stage image coder framework based on modeling images as edge cartoons + textures. In stage 1, we infer and efficiently code the edge information from the image using a multiscale wedgelet decomposition. In stage 2, we code the residual, "edgeless" texture image using a standard wavelet coder. Our preliminary coder improves significantly over standard wavelet coding techniques in terms of visual quality.

## 1 Introduction

The wavelet transform is responsible for much of the progress in image compression over the last ten years — culminating in the JPEG2000 standard. However, JPEG2000 and most other wavelet coders employ separable two-dimensional (2-d) filterbanks that are the simplest possible extension of 1-d techniques. Recently it has become clear that such techniques have hit a brick wall in performance and that improvements will only be possible by committing to truly two-dimensional processing that captures the inherent *geometry* of images. By "geometry" we refer to the fact that image edges and ridges tend to lie along more or less smooth curves. Edges are perceptually one of the most prominent features in images (the other is texture).

Even the best wavelet-based coders [1–3] show weakness when dealing with edges. These coders are ideally suited to representing isotropic textures and smooth (polynomial) image regions, but ill-suited to representing singularities such as edges, for the following reasons. First, the number of wavelet coefficients required to

*This work was supported by the NSF, ONR, AFOSR, DARPA, and the Texas Instruments Leadership University Program.

describe an edge depends only on how many of them lie in the locale of the edge and not on the smoothness of the edge contour. Second, each wavelet coefficient in an edge region has a "large" magnitude when compared to most of its spatial neighbors and is thus difficult to predict and code. Third, failure to consider the wavelet coefficients' joint behavior along an edge can result in visually displeasing "ringing" artifacts in the compressed image. Recent coders have tried to deal with this joint behavior implicitly, using spatially varying models that try to predict the distribution of a wavelet coefficient given the values of its within-scale neighbors [3, 4]. These have achieved some degree of success, but edges remain the single largest performance roadblock to efficient wavelet compression.

We base our approach on a simple observation: if we could vertically collapse the edges in the image (that is, make them disappear), then we would be left with a texture image that could be efficiently coded with a standard wavelet coder. Moreover, since image edges tend to be smooth, they are low-dimensional structures that are easy to describe and code explicitly (much like a smooth 1-d function can be coded efficiently). This suggests a simple image model

$$\begin{aligned} \mathsf{image} &= \{\mathsf{edge\ cartoon}\} + \{\mathsf{textures}\} \\ f(x,y) &= c(x,y) + t(x,y) \end{aligned}$$

and a two-stage coding scheme that codes first the cartoon $c$ and then the texture residual $t = f - c$.

The overall performance of the coder relies on the smoothness of the contours in the cartoon and on the smoothness of the residual textures between the contours. Efficient cartoon coding promises sharp image representations at low bitrates.

**Stage 1: Multiscale wedgelet cartoon coder:** The algorithm for estimating the cartoon image $c$ must be robust, efficient, and provide a representation that can be efficiently coded. In this paper, we propose a new approach to form edge cartoons based on the multiscale *wedgelet* decomposition [5].

The wedgelet decomposition adaptively partitions an image into edge and non-edge regions and then uses large wedges to represent smooth edge curves and concatenated small wedges to represent rapidly changing edge curves. An optimal MSE partition can be obtained by dynamic programming and encoded very efficiently. The result is a cartoon image $c$ that characterizes most of the edge information in the image. Indeed for a true step edge contour, wedgelets are rate/distortion (R/D) optimal [6]. Figure 2(c) shows a wedgelet cartoon for the *Cameraman* test image.

**Stage 2: Wavelet residual texture coder:** Given the wedgelet cartoon $c$, we then code the residual texture image $t = f - c$ using a conventional wavelet coder. We must be careful, since estimation or coding errors in the cartoon lead to new artifacts in the residual image that do not correspond to natural image features. A small error in the location of an edge contour in $c$, for example, will leave a tall ridge in $t$. Since ridges present the same difficulties as edges for wavelet coders, our

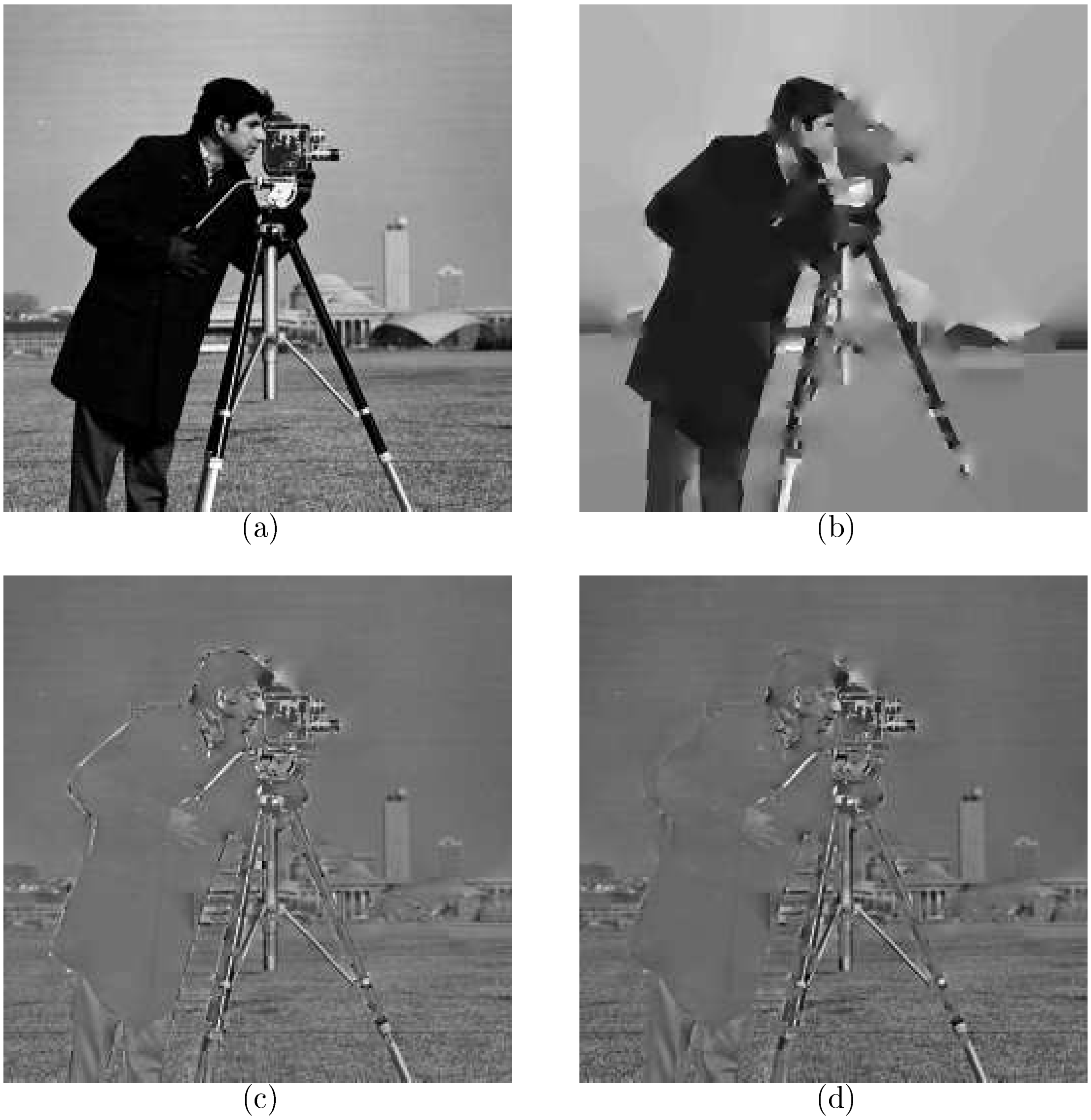


Figure 1: *(a) Original* Cameraman *image* $f(x, y)$. *(b) Coded wedgelet cartoon* $c(x, y)$ *with smoothing of grayscale values at wedgelet block borders, 0.12 bits per pixel (bpp). (c) Residual image* $t(x, y) = f(x, y) - c(x, y)$. *(d) Residual image* $t(x, y)$ *with edge-error artifacts removed by tapered masking.*

effort in edge cartooning would be in vain. Fortunately, given the wedgelet cartoon, we have information about the artifacts we can expect in the residual (for example, where they will occur and how large they will be); this information can be leveraged to create a residual better suited to standard wavelet coding. In Section 2.2, we describe a simple scheme for modifying the residual image near edges to remove these artifacts.

See Figure 3 for a comparison of a preliminary implementation of our coder with a representative zerotree (EZW) wavelet coder [2]. Most importantly, although our techniques tend to degrade PSNR performance, we note a significant increase in visual performance, with sharply defined edges and minimal ringing artifacts.

In related work, Froment [7] and Dragotti [8] have developed techniques for explicitly extracting contours from images, while Mallat and Pennec have proposed bandelets [9] as a contour estimate that explicitly deals with nearby artifacts. Meyer [10] has further justified that this sort of mixture modeling of image features can yield efficient coding algorithms tailored to individual types of features. Our primary contribution in this paper is a new and efficient technique for extracting and coding the cartoon sketch based on wedgelets. We also propose further refinements to Froment's approach to residual coding. Our framework is general and can be extended to integrate all aspects of our coder into a jointly optimal R/D framework.

# 2 Cartoon+Texture Coder

To code an image, we first describe the edge structure in the image using a scheme similar to Donoho's wedgelet decomposition [5] and code the wedgelet coefficients using an efficient tree-structured, top-down, predictive framework. Second, we treat the residual image for artifacts caused by removing the edges and compress it using a standard wavelet coder.

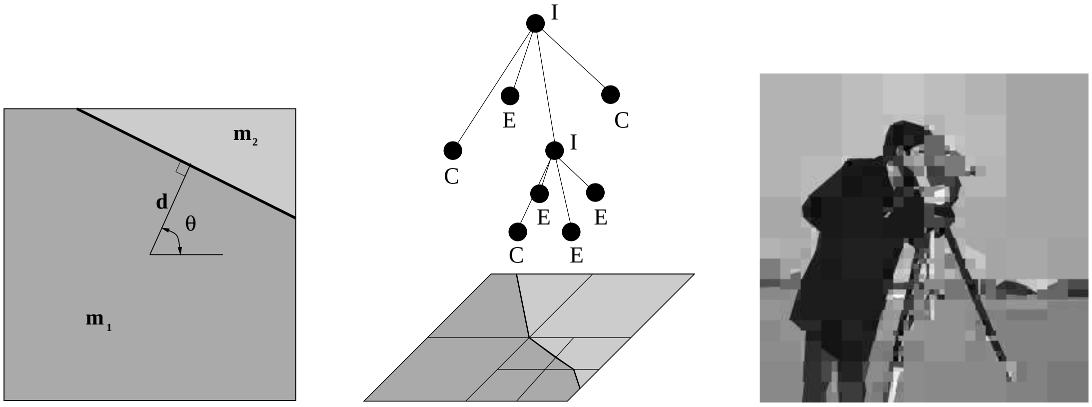
## 2.1 Stage 1: Multiscale wedgelet cartoon coder

### 2.1.1 Wedgelet decomposition

The wedgelet decomposition partitions an image into dyadic blocks that are each either constant (with value equal to the mean of the image over that block) or a wedgelet, (containing a straight edge of arbitrary slope and intercept) [5]. Constant blocks provide local grayscale information and are described by a single parameter (the mean $m$). Wedgelet blocks provide local geometry information and require four parameters $(d, \theta, m_1, m_2)$; see Figure 2(a). The wedgelet decomposition of an image corresponds to a pruned quadtree decorated with these parameters; see Figure 2(b). In particular, note that wedgelet blocks do not have to all be the same size.

The wedgelet representation is adaptive. Image regions that are approximately constant are best represented by a few large dyadic blocks decorated with constants. Image regions containing long, straight edges are best represented by a few large dyadic blocks decorated with wedges. Image regions containing contours are best represented by several small dyadic blocks decorated with wedges aligned along the contours. The wedgelet representation is a nearly optimal representation (in terms of approximation decay rate) for the simple "horizon class" of images consisting of two constant regions separated by a smooth boundary [5].

The key to implementing the wedgelet decomposition is to find the best wedge fit over all possible constants and $(d, \theta, m_1, m_2)$'s for a given dyadic block. This problem




(a) (b) (c)

Figure 2: *(a) Parameterization of a wedgelet: $d$ is the normal distance from the edge to the center of the block, $\theta$ is the angular orientation of the normal line, and $m_1$, $m_2$ are the average intensities of the image on each side of the edge. (b) Multiscale wedgelet decomposition of an image. Each leaf node is decorated with wedgelet parameters – either $m$ for a constant block $C$ or $(d, \theta, m_1, m_2)$ for a wedgelet block $E$. Interior nodes $I$ include parameters $(d, \theta, m_1, m_2)$ for use in predictive coding of the tree. (c) Cartoon drawing of* Cameraman *image based on information in the leaf nodes of the wedgelet decomposition.*

is non-trivial; the set of wedges over an $N \times N$ block forms a nonlinear, 2-d manifold in $\mathbb{R}^{N^2}$. We propose two approaches. First, the best projection for each $(d, \theta)$ can easily be calculated, given the Radon transform of the image block. Second, a fast, approximate projection can be obtained based on the phase relationships of the complex wavelet transform [11].

By using small enough dyadic blocks, we can obtain an arbitrarily good approximation to the image with a wedgelet decomposition. Of course, more blocks mean more coefficients to code. The optimal wedgelet decomposition for a given complexity can be found using the venerable CART algorithm by weighing the coding cost of splitting a coarse dyadic block into four subblocks against the gain in edge description accuracy [5]. The algorithm returns a pruned quadtree, as shown in Figure 2(b), with each node labeled $E$ for edge (or wedge), $C$ for constant, or $I$ for interior (that is, not a tree leaf). By "drawing" a wedge or a constant value according to the parameters on the leaves of the tree, we obtain the wedgelet cartoon approximation $c$ (see Figure 2(c)).

### 2.1.2 Cartoon coder

We code the tree-based wedgelet cartoon $c$ using a top-down predictive framework. Our algorithm makes all coding decisions in a R/D consideration, where $D$ reflects the effective mean-square distortion of the coded image. The coder transmits all information in a single pass, starting with the root node of the wedgelet tree. Three types of information must be sent: (1) a symbol from $\{E, I, C\}$ denoting the state of each node, (2) wedge parameters $(d, \theta)$ when appropriate, and (3) grayscale values $(m)$ or $(m_1, m_2)$. After designating a node $E$ or $C$, we code no further information regarding its descendants in the tree.

For a given node to be coded, we predict its edge parameters and grayscale values based on the previously coded parameters of its parent (necessarily an $I$ node). This causality assures that the decoder can make the same predictions. We make the prediction based on a simple spatial intuition: the parent's wedgelet is divided dyadically to predict the wedgelets of its four children. This provides a prediction for all edge parameters at the node; the coder transmits only the deviations from the true wedgelet parameters (the prediction errors).

We model each prediction error as a beta$(p, q)$ random variable, with zero mean and symmetric distribution ($p = q$). For the prototype implementation in this paper, we use a simple uniform quantizer with deadzone. For each variable to be coded, a beta shape parameter $\widehat{p}$ is estimated from previously coded instances of the same parameter, and a table lookup (using $\widehat{p}$ and the overall desired R/D slope $\lambda$) provides the optimal quantization step sizes. The coder easily computes the bin probabilities and uses them for arithmetic coding of each quantization bin index.

After coding the pruned wedgelet tree, we translate it into the cartoon sketch $c$. The wedgelets provide the locations of the edges in the cartoon, and grayscale values describe the shading in regions away from the edges. Because neighboring constant blocks may have slightly different grayscale values, we smooth such regions to prevent the creation of phantom edge artifacts at the block boundaries. Figure 1(b) shows an example on the *Cameraman* image. Note that this smoothing preserves all desired edges from the wedgelet sketch in Figure 2(c) but removes blocking artifacts at the borders of wedgelet blocks.

Future improvements to the coder should consider the previously coded parameters of a node's neighbors, in addition to its parent. This should yield more intelligent predictions of edge parameters and reduce the number of bits required to transmit prediction errors.

## 2.2 Wavelet residual texture coder

After transmitting the wedgelet cartoon $c$, it is necessary to code the residual image $t = f - c$. Unfortunately, errors in the wedgelet approximation and its lossy encoding will create ridge-like artifacts in the residual image $t$. These artifacts do not correspond directly to natural features of the image; they are merely the difference between the true edges and the sketch. We identify three approaches for coding the residual and dealing with these artifacts.

**1.** Assuming that we wish the cartoon to provide the "only" coded information about the edges, then we should discard these difference artifacts before coding the residual. A simple technique is to "mask" the edge artifacts out of the texture image, setting $t = 0$ near all cartoon edges (as in [7]). With much of the highfrequency information removed from the image, a standard wavelet coder efficiently codes the residual.

In this paper, to eliminate the ridge artifacts in the residual image, we implement a tapered masking scheme. First, for each pixel $(x, y)$ in the image, we compute the distance $\delta(x, y)$ to the nearest cartoon edge. We define a tapering function $T(\delta)$ such that $T(0) = 0$ and $T$ has a smooth transition up to a value of 255 (the maximum pixel intensity). For each pixel in the residual image, we truncate its value to ensure that its magnitude is less than $T(\delta(x, y))$. Near the edge, this forces residual values to be small (eliminating large ridges); away from the edge this leaves the residual unchanged. The smooth transition of the envelope $T$ prevents us from creating discontinuities when tapering the residual image.

As we will see, this process can be visually effective, but it discards information that can have a dramatic impact on PSNR. It also discards legitimate texture information close to edges, which cannot easily be separated from edge artifacts.

**2.** An obvious alternative to the above technique is simply to code the entire residual $t$ using a standard wavelet coder. This allows the residual coder to correct the errors in the wedgelet approximation (the difference between the straight edges and the contour). The numerous ridges can be expensive to code, but because we are not intentionally throwing information away (by "masking"), we gain in overall PSNR performance compared to technique #1 above. In [7], Froment demonstrates the effectiveness of coding this type of residual using wavelet packets.

**3.** A third, and perhaps better, alternative is to use information about the cartoon image when coding the residual. Although the edge artifacts are tall, thin ridges, we know that they exist along the edge contour, possibly alternating between positive and negative as the contour crosses the wedgelet cartoon boundary. "Bandelets" [9] offer one approach to separating regions around a contour from the rest of the residual, but they have yet to yield a viable compression algorithm. Future work will focus on improving this stage of our coder by attempting to intelligently code the information near the edge approximations. For example, the simple oscillatory patterns of the artifact ridges may lead to an efficient means of coding them explicitly.

# 3 Results

To illustrate our prototype implementation, as well as the issues concerned with residual coding, we consider the 256×256 *Cameraman* image shown in Figure 3(a). For comparison purposes, we show in Figure 3(b) the EZW coded image [2] at 0.40 bits per pixel (bpp) and a PSNR of 28.98 dB (other state-of-the-art coders yield similar results).

We prune the wedgelet tree and code the cartoon using the approach of Section 2. Figure 1(b) displays the wedgelet cartoon $c$, coded at 0.12 bpp, with smoothing


Figure 3: *(a) Original* Cameraman *image $f$. (b) Standard EZW coding of image, 0.40 bpp, PSNR = 28.98 dB. Note the ringing around the predominant edges. (c) Wedgelet cartoon + EZW coded residual (no masking), total 0.40 bpp, PSNR = 28.48 dB. (d) Wedgelet cartoon + EZW coded residual (with masking), 0.40 bpp, PSNR = 27.35 dB. Note the absence of ringing around the edges.*

applied to the wedgelet block boundaries. Subtracting this sketch from the original image creates the residual image $t$ shown in Figure 1(c). As expected, significant artifacts occur along the sketched edge approximations. As described in technique #2 of Section 2.2, we code the residual image using EZW at a bitrate of 0.28 bpp and add it back to the wedgelet cartoon; this yields the coded image in Figure 3(c). This result uses a total coding rate of 0.40 bpp to achieve a PSNR of 28.48 dB. While the PSNR is lower than the original EZW result, we notice much less ringing near the dominant edges of the image.

In Figure 1(d) we show the results of the tapered masking applied to the residual image. As in technique #1 of Section 2.2, we code this masked residual using EZW at a bitrate of 0.28 bpp and add it back to the wedgelet cartoon; this yields the coded image in Figure 3(d). This result uses a total coding rate of 0.40 bpp to achieve a PSNR of 27.35 dB. The PSNR drops, because we do not allow the texture coder to correct for errors in the cartoon coder's approximation of the edge contour or shape. Note that while this technique yields the lowest overall PSNR of the group, it offers the best overall visual performance, in particular the greatest reduction of ringing artifacts near edges. In addition, we note that this technique achieves the best PSNR in regions of the image away from edges, such as the sky and the grass.

# 4 Discussion & Conclusions

We have presented a simple yet effective scheme for identifying, extracting, and coding a cartoon sketch of the edges in an image. The residual texture features of the image can then be coded in an efficient manner using a standard wavelet coder.

Some modifications may help the efficiency of our scheme. The wedgelet estimations, currently performed independently on each block, may be improved by considering the spatial correlation of neighboring wedgelets. Also, an examination of the residual image in Figure 1(c) shows that a great deal of edge structure remains after the cartoon sketch is subtracted from the image. In addition to the ridges caused by wedgelet approximations, some edges are completely ignored by the tree-pruning. These edges may occur in low-contrast regions (such as the tall building) where they have little effect on distortion, or they may occur as thin ridges (such as the bottom of the right leg of the tripod) where they require many small wedgelets, and hence many bits to code. Future work should focus on exploiting the structure in these regions as well.

All coders must make errors somewhere in the coded image. The present incarnation of our coder reduces ringing edge artifacts considerably by transferring error energy into a thin region immediately surrounding the cartoon edges, where it is perceptually less noticeable. This leaves a residual that is better suited to standard wavelet-based compression (when compared to the original image), but unfortunately the result degrades the overall PSNR. We believe that a more intelligent treatment of the residual artifacts near edge regions will lead to a solution that outperforms current wavelet coders, both visually and in terms of PSNR.

Most importantly, though, future research will concentrate on merging our two-stage algorithm into an intelligent progressive encoder that makes all decisions in an optimal R/D sense. One possibility may be to incorporate wedgelet blocks into the zerotree-pruning of the SFQ coder [12]; such an approach would only place wedges when globally R/D efficient.

# References


[1] A. Said and W. A. Pearlman, “A new fast and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Trans. Circuits and Systems for Video Tech.*, vol. 6, pp. 243–250, June 1996.

[2] J. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” *IEEE Trans. Signal Proc.*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.

[3] S. LoPresto, K. Ramchandran, and M. T. Orchard, “Image coding based on mixture modeling of wavelet coefficients and a fast estimation-quantization framework,” in *Data Compression Conference ’97*, Snowbird, Utah, 1997, pp. 221–230.

[4] X. Li, *Spatially Adaptive Statistical Modeling and its Applications to Image Processing*, Ph.D. thesis, EE Dept. Princeton University, March 2000.

[5] D. Donoho, “Wedgelets: Nearly-minimax estimation of edges,” *Annals of Stat.*, vol. 27, pp. 859–897, 1999.

[6] 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.

[7] 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 Academic, 2001.

[8] 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.

[9] E. L. Pennec and S. Mallat, “Image compression with geometrical wavelets,” in *IEEE Int. Conf. on Image Proc. – ICIP ’01*, Thessaloniki, Greece, Oct. 2001.

[10] F. G. Meyer, A. Z. Averbuch, and R. R. Coifman, “Multi-layered image representation,” in *Wavelets in Signal and Image Analysis*, A. A. Petrosian and F. G. Meyer, Eds. Kluwer Academic, 2001.

[11] J. K. Romberg, H. Choi, and R. G. Baraniuk, “Multiscale edge grammars for complex wavelet transforms,” in *IEEE Int. Conf. on Image Proc. – ICIP ’01*, Thessaloniki, Greece, Oct. 2001.

[12] Z. Xiong, K. Ramchandran, and M. T. Orchard, “Space-frequency quantization for wavelet image coding,” *IEEE Trans. Image Processing*, vol. 6, no. 5, pp. 677–693, 1997.