Wavelet-Domain Approximation and Compression of Piecewise Smooth Images

Michael B. Wakin, *Student Member, IEEE*, Justin K. Romberg, *Member, IEEE*, Hyeokho Choi, *Senior Member, IEEE*, and Richard G. Baraniuk, *Fellow, IEEE*

***Abstract*—The wavelet transform provides a sparse representation for smooth images, enabling efficient approximation and compression using techniques such as zerotrees. Unfortunately, this sparsity does not extend to *piecewise smooth* images, where edge discontinuities separating smooth regions persist along smooth contours. This lack of sparsity hampers the efficiency of wavelet-based approximation and compression. On the class of images containing smooth C^2 regions separated by edges along smooth C^2 contours, for example, the asymptotic rate-distortion (R-D) performance of zerotree-based wavelet coding is limited to $D(R) \lesssim 1/R$, well below the optimal rate of $1/R^2$. In this paper, we develop a geometric modeling framework for wavelets that addresses this shortcoming. The framework can be interpreted either as 1) an extension to the "zerotree model" for wavelet coefficients that explicitly accounts for edge structure at fine scales, or as 2) a new atomic representation that synthesizes images using a sparse combination of wavelets and *wedgeprints*—anisotropic atoms that are adapted to edge singularities. Our approach enables a new type of quadtree pruning for piecewise smooth images, using zerotrees in uniformly smooth regions and wedgeprints in regions containing geometry. Using this framework, we develop a prototype image coder that has near-optimal asymptotic R-D performance $D(R) \lesssim (\log R)^2/R^2$ for piecewise smooth C^2/C^2 images. In addition, we extend the algorithm to compress natural images, exploring the practical problems that arise and attaining promising results in terms of mean-square error and visual quality.**

***Index Terms*—Edges, image compression, nonlinear approximation, rate-distortion, wavelets, wedgelets, wedgeprints.**

I. Introduction

WAVELETS have become pervasive in image processing, with applications ranging from denoising and estimation [1] to segmentation [2] and computer vision [3]. Indeed, wavelets are the foundation of most state-of-the-art image coders [4]–[7], including the recent JPEG-2000 standard [8]. A number of factors, both practical and theoretical, contribute to their success. Practically, the separable two-dimensional (2-D) wavelet transform of an n-pixel image can be computed in $O(n)$ time. In addition, the multiscale nature of the transform suggests a simple wavelet-domain modeling framework based on *zerotrees* that has been successfully applied in general purpose image coders [4]–[6].

The effectiveness of compression algorithms based on wavelets and zerotrees can also be understood from a theoretical standpoint. For a *smooth* image (potentially including a finite number of point singularities), the wavelet decomposition is *sparse*: most wavelet coefficients are approximately zero, and only a few large coefficients (concentrated at coarse scales and around the singularities at fine scales) are required to accurately reconstruct the image [9]. Zerotree-based image coders exploit this structured organization of the significant coefficients, essentially constructing approximations from wavelets arranged on a connected quadtree. Recent results from nonlinear approximation show that, for smooth images with isolated point singularities, such tree-based approximations properly capture the significant coefficients and achieve optimal asymptotic mean-square error (mse) decay [10]–[12]. These results can also be extended to asymptotic rate-distortion (R-D) bounds for simple zerotree image coders [12], [13].

Despite their success, however, wavelet-based image compression algorithms face significant challenges when confronted with *piecewise smooth* images, where geometric edge contours create discontinuities between smooth image regions. Again, the problem reveals itself in theory and in practice. Wavelet representations of edges are significantly less sparse; tree-based wavelet approximations of piecewise smooth images are not asymptotically optimal [14]; and practical image coders tend to introduce all-too-familar "ringing" artifacts around edges in the image at low bit rates.

In this paper, we develop a geometric modeling framework in the multiscale spirit of wavelets and zerotrees that is designed to address these shortcomings. Theoretically, we demonstrate that this framework yields a novel representation for images that provides near-optimal asymptotic approximation and compression rates for a representative class of piecewise smooth images. Practically, we also incorporate these ideas into a new coder for natural images, with promising improvements upon a state-of-the-art wavelet–zerotree coder for images containing strong geometric features.

A. Wavelets, Zerotrees, and Image Coding

We start with a brief overview of wavelets and zerotrees, the foundation for our modeling framework. For the sake of illustration, and to facilitate asymptotic analysis, we concentrate initially on images defined over a continuous domain.

Manuscript received November 22, 2004; revised April 6, 2005. This work was supported in part by a National Science Foundation (NSF) Graduate Research Fellowship, NSF Grant CCR-9973188, Office of Naval Research Grant N00014-02-1-0353, Air Force Office of Scientific Research Grant F49620-01-1-0378, and the Texas Instruments Leadership University Program. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Amir Said.

M. B. Wakin and R. G. Baraniuk are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: wakin@ece.rice.edu; richb@ece.rice.edu).

J. K. Romberg is with the Department of Applied and Computational Mathematics, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: jrom@acm.caltech.edu).

H. Choi, deceased, was with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695 USA.

Digital Object Identifier 10.1109/TIP.2005.864175

The wavelet transform decomposes an image $X \in L_2([0,1]^2)$ into a superposition of 2-D atoms that are shifted and dilated versions of bandpass mother wavelet functions $\{\psi^{\mathtt{V}}, \psi^{\mathtt{H}}, \psi^{\mathtt{D}}\}$ and a lowpass scaling function ϕ

$$X(t) = \sum_{k \in \mathcal{K}(j_0)} u_{j_0,k}\phi_{j_0,k}(t) + \sum_{b \in \{\mathtt{V},\mathtt{H},\mathtt{D}\}} \sum_{j \geq j_0} \sum_{k \in \mathcal{K}(j)} w^b_{j,k}\psi^b_{j,k}(t)$$

where $t \in [0,1]^2$ denotes a 2-D coordinate in the unit square, $\phi_{j_0,k} := 2^{j_0}\phi(2^{j_0}t - k)$, and $\psi^b_{j,k} := 2^j\psi^b(2^j t - k)$ [9]. The wavelet coefficients are indexed by a *scale* $j \geq j_0 > 0$, a 2-D *location* $k := (k_1, k_2) \in \mathcal{K}(j) = \{(1/2)2^{-j}, (3/2)2^{-j}, \cdots, (2^j - 1/2)2^{-j}\} \times \{(1/2)2^{-j}, (3/2)2^{-j}, \cdots, (2^j - 1/2)2^{-j}\}$, and a *subband* $b \in \{\mathtt{V}, \mathtt{H}, \mathtt{D}\}$ denoting the orientation (vertical, horizontal, or diagonal) of the wavelet basis function. Under certain conditions (which we assume for the rest of the paper), the set $\{\phi_{j_0,k}, \psi^b_{j,k}\}$ forms an orthonormal basis with each $\psi^b_{j,k}$ compactly supported on a square $S_{j,k}$ with sidelength $C_{\text{supp}}2^{-j}$ [15].

The wavelet decomposition is *multiscale* and *local*; a wavelet coefficient $w^b_{j,k} := \langle X, \psi^b_{j,k}\rangle$ is affected only by image values in the region $S_{j,k}$. As such, each of the three subbands of the wavelet transform can be naturally arranged in a *quadtree* $\mathcal{Q}^b$. Each *node* $(j,k) \in \mathcal{Q}^b$ is associated with a wavelet coefficient $w^b_{j,k}$ and a wavelet basis function $\psi^b_{j,k}$ (which is in turn associated with a square $S_{j,k}$). The *children* $\mathcal{C}_{j,k}$ of node (j,k) are the four nodes at scale $j+1$ located directly around k:

$$\mathcal{C}_{j,k} := \left\{(j+1, k') :\ k'_1 = k_1 \pm \frac{1}{4}2^{-j},\ k'_2 = k_2 \pm \frac{1}{4}2^{-j}\right\}.$$

Continuing recursively through scale, we denote the *descendants* of (j,k) at scales $j' > j$ as

$$\mathcal{U}_{j,k} := \left\{(j',k') :\ j' > j,\ k'_1 \in \left[k_1 - \frac{1}{2}2^{-j}, k_1 + \frac{1}{2}2^{-j}\right],\ k'_2 \in \left[k_2 - \frac{1}{2}2^{-j}, k_2 + \frac{1}{2}2^{-j}\right]\right\}$$

and the *subtree* of $\mathcal{Q}^b$ rooted at node (j,k) as $\mathcal{T}_{j,k} := (j,k) \cup \mathcal{U}_{j,k}$.

The wavelet basis functions possess a key *sparseness* property that makes them especially well suited for representing images: if X is "smooth" over the region $S_{j,k}$, then the corresponding $w^b_{j,k}$ have relatively small magnitude (this can be made mathematically precise if the ψ^b have vanishing moments; see [9] and [15] for details). In fact, since the supports of the descendants $\mathcal{U}_{j,k}$ nest inside $S_{j,k}$, the entire subtree $\mathcal{T}_{j,k}$ of wavelet coefficients representing the smooth region will have small magnitude.

Many wavelet-based compression algorithms (including EZW [4], SPIHT [5], and space frequency quantization (SFQ) [6]) exploit this sparsity by replacing subtrees of small coefficients with *zerotrees*, using one symbol to encode many wavelet coefficients without incurring significant error. Effectively, these coders construct an approximation of the image using wavelets on a connected quadtree. Over regions where the image is smooth, the quadtree is *pruned* back to a coarse scale. Over regions containing singularities, the quadtree is grown deeper to include wavelets at finer scales. In terms of asymptotic mse decay (as the number of preserved coefficients grows), this type of wavelet-based quadtree approximation is optimal for images that are uniformly smooth [12], [13]. Let $X_{\widehat{\mathcal{Q}}}$ denote the approximation to X constructed from wavelets on the set of pruned quadtrees $\widehat{\mathcal{Q}} := \{\widehat{\mathcal{Q}}^{\mathtt{V}}, \widehat{\mathcal{Q}}^{\mathtt{H}}, \widehat{\mathcal{Q}}^{\mathtt{D}}\}$

$$X_{\widehat{\mathcal{Q}}} := \sum_{k \in \mathcal{K}(j_0)} u_{j_0,k}\phi_{j_0,k}(t) + \sum_{b \in \{\mathtt{V},\mathtt{H},\mathtt{D}\}} \sum_{(j,k) \in \widehat{\mathcal{Q}}^b} w^b_{j,k}\psi^b_{j,k}(t) \tag{1}$$

and let $|\widehat{\mathcal{Q}}^b|$ denote the number of wavelets preserved in a pruned quadtree. If $X \in C^r([0,1]^2)$ (that is, if X is r-times continuously differentiable), then there exists for every integer N a set of quadtrees $\widehat{\mathcal{Q}}(N)$ such that $|\widehat{\mathcal{Q}}^{\mathtt{V}}| + |\widehat{\mathcal{Q}}^{\mathtt{H}}| + |\widehat{\mathcal{Q}}^{\mathtt{D}}| \leq N$ and[1]

$$\left\|X - X_{\widehat{\mathcal{Q}}(N)}\right\|_2^2 \lesssim N^{-r}. \tag{2}$$

The exponent $-r$ in (2) is optimal; there is no orthonormal basis in which the N-term approximations asymptotically go to zero more quickly for every image in C^r. Amazingly, the approximation rate does not change if we introduce a finite number of point singularities into X.

Approximation results such as (2) give insight into the quality of an image representation. A closely related measure of efficiency is compression performance, where the number of bits required to encode an approximation reflects the complexity of specifying *which* coefficients are significant in addition to their values. In the above scenario, the approximation result (2) can be translated into a similar asymptotic R-D result using a simple prototype coder [12], [13] that finds the quadtree approximation (1), codes the associated quadtrees $\widehat{\mathcal{Q}}(N)$, and then quantizes the nonzero coefficients at each node of the $\widehat{\mathcal{Q}}(N)$. Using $R \lesssim N$ bits for the quantized wavelet coefficients and the quadtree $\widehat{\mathcal{Q}}(N)$, the quantized approximation $\widetilde{X}_{\widehat{\mathcal{Q}}(N)}$ can be coded with distortion $D := \|X - \widetilde{X}_{\widehat{\mathcal{Q}}(N)}\|_2^2 \lesssim N^{-r}$. Writing the distortion D in terms of the number of bits R, we obtain the R-D performance

$$D(R; X) \lesssim R^{-r} \tag{3}$$

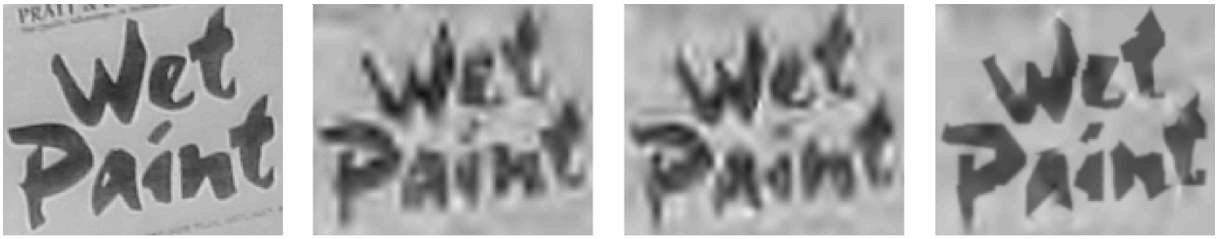
for the prototype coder. For $X \in C^r([0,1]^2)$ (potentially including a finite number of point singularities), the number of bits required is asymptotically equivalent to the Kolmogorov entropy [12]; hence, the prototype coder comes within a constant of the best possible R-D performance for this class of images.

The quadtrees used by the prototype coder [12] and by practical zerotree coders [4]–[6] *adapt* to each image. Using a tree-pruning algorithm (as in [6]), we can find the quadtrees $\widehat{\mathcal{Q}}(N)$ that optimally balance the number of bits used against the distortion incurred. Given a $\lambda > 0$, we can solve

$$\min_{\widehat{\mathcal{Q}}} \left\|X - \widetilde{X}_{\widehat{\mathcal{Q}}(N)}\right\|_2^2 + \lambda R \tag{4}$$

with a dynamic programming algorithm similar to CART [16]. As λ becomes small, (4) will return approximations with many terms (high values of R); as λ increases, the number of bits decreases. The solution to (4) for a given λ is the best approximation that uses the resulting number of bits.

[1]We focus in this paper on *asymptotic* approximation performance. We use the notation $f(k) \lesssim g(k)$ if there exists a constant C, not dependent on k, such that $f(k) \leq Cg(k)$.



(a) Original (b) JPEG-2000 (c) SFQ (d) WSFQ

Fig. 1. (a) Portion of the *Wet Paint* test image from Fig. 9 [18]. Ringing artifacts are pronounced in images compressed with wavelet-based coders: (b) JPEG-2000 [8], [19], and (c) SFQ [6]. (d) Ringing is reduced in our proposed WSFQ algorithm by combining wavelets with wedgeprints. Section V presents more details.

B. The Challenge of Geometry

Unfortunately, the success of wavelets does not extend to images containing geometrical features such as edges, ridges, or other singularities persisting along contours. The prominence and significance of these features [17] has forced researchers in image modeling and compression to search for a framework outside of wavelets and zerotrees.

The problem stems from the number of wavelet coefficients needed to represent the geometrical regions of the image. To approximate an edge accurately, a wavelet–zerotree coder must code all coefficients whose supports overlap the contour out to fine scales. The number of such coefficients grows as $O(2^j)$; that is, many wavelet coefficients are required to represent even a simple, straight edge. This lack of sparsity is reflected in the asymptotic performance of wavelet-quadtree based approximations and prototype coders [14]. For an image that is r times differentiable except along a smooth contour (across which it is discontinuous), the approximation and R-D results of (2) and (3) become stuck at $\|X - X_{\widehat{\Omega}(N)}\|_2^2 \lesssim N^{-1}$ and $D(R;X) \lesssim R^{-1}$. For $r > 1$, these asymptotic rates are considerably slower than in the isolated singularity case (2), (3). In contrast, the Kolmogorov entropy for these piecewise smooth images behaves the same asymptotically as for uniformly smooth images [12], meaning that there exists a coder that can achieve $D(R;X) \lesssim R^{-\min(r,r_c)}$, where r_c is the number of times the (1-D) contour can be differentiated.

The failure of wavelets to represent edges succinctly plagues practical image compression algorithms as well. The coder is faced with an unfortunate dilemma: either use a large number of bits to encode an edge, or suffer visually offensive "ringing" artifacts in the compressed image [see Fig. 1(a)–(c)].

The abundance of wavelet coefficients describing geometry is not an immediate barrier to effective wavelet-domain image compression, however. There is, in fact, a strong *coherency* among the coefficients imposed by the structure of the geometry. For example, in the case of an isolated, sharp edge, the 1-D information describing the trace of the edge contour completely determines the values of the 2-D wavelet coefficients. This coherency, however, is quite difficult to characterize explicitly. Most wavelet-domain algorithms resort instead to modeling collective quantities such as the variance of the coefficients (see, for example, [7]). Such simplifications lead to suboptimal R-D performance, however, and again lead to ringing artifacts when quantization disrupts the geometrical coherency. In the end, every wavelet-domain coder must face the challenge of geometry.

C. Current Approaches for Geometric Image Compression

Recent work has focused on improving models and algorithms for wavelet-based image compression. Techniques based on level lines [20], redundant dictionaries [21], and an enhanced quadtree prune/join algorithm [22] provide efficient solutions for coding geometric image features. Such schemes can also be applied in a two-stage algorithm for coding natural images: first code the geometric contour information (using a representation well-suited for geometry), and then code the remaining 2-D features (using a representation such as wavelets that is well-suited for smooth regions). Despite the efficiency of these geometric coders, however, such a two-stage approach does not provide a comprehensive framework for analyzing or considering the R-D impacts of the classifications. For example, errors in identifying or encoding a 1-D edge contour may produce unwanted artifacts in the second stage, and it may not be straightforward to achieve the R-D optimal balance of fidelity.

Additional work in harmonic analysis has focused on developing representations that provide sparse decompositions for certain geometric image classes. Examples include curvelets [14], [23] and contourlets [24], slightly redundant tight frames consisting of anisotropic, "needle-like" atoms. In [25], bandelets are formed by warping an orthonormal wavelet basis to conform to the geometrical structure in the image. A nonlinear multiscale transform that adapts to discontinuities (and can represent a "clean" edge using very few coarse scale coefficients) is proposed in [26]. Each of these new representations has been shown to achieve near-optimal asymptotic approximation and R-D performance for piecewise smooth images consisting of C^α regions separated by discontinuities along C^α curves, with $\alpha = 2$ ($\alpha \geq 2$ for bandelets). Some have also found use in specialized compression applications such as identification photos.[2]

D. Geometric Modeling Framework for Wavelet Coefficients

In this paper, we propose an alternative approach for representing and compressing piecewise smooth images, demonstrating that the challenge of geometry can be addressed *in the wavelet domain* by using a novel quadtree-based modeling framework. Our scheme is based on the simple yet powerful observation that geometric features can be efficiently approximated using local, geometric atoms in the spatial domain, and that the projection of these geometric primitives onto wavelet subspaces can therefore approximate the corresponding wavelet coefficients. We refer to these projections

[2]Let it Wave. www.letitwave.fr

as *wedgeprints*, noting the parallels to the 1-D footprints presented in [27]. Due to their organization on wavelet quadtrees, wedgeprints fit naturally into the wavelet–zerotree pruning and approximation framework. We outline the theoretical asymptotic gains in approximation and compression afforded by wedgeprints, and we discuss the practical application of these principles to discrete image compression.

Wedgeprints can be interpreted in two different ways. First, wedgeprints are an extension of zerotrees in that they implicitly *model* the coherency among wavelet coefficients in a subtree (but in high-energy regions near edges) and can be encoded using relatively few bits. Alternatively, wedgeprints can be viewed as an *adaptive atomic representation* that provides a sparse approximation for geometric image features and can be combined with wavelets for sparse approximation of piecewise smooth images. This enables a new type of quadtree pruning for piecewise smooth images, using zerotrees in uniformly smooth regions and wedgeprints in regions containing geometric contours, and it naturally leads to a top-down compression algorithm for encoding a pruned wavelet–wedgeprint quadtree. In this quadtree framework, optimizing bit rate allocation among zerotree and wedgeprint representations is straightforward; we present an efficient tree-pruning algorithm that uses an R-D criterion to strike the proper balance.

We begin in Section II by presenting a simple formulation of wedgeprints based on projecting elements of the multiscale *wedgelet dictionary* [28], a collection of geometric atoms that combine to form piecewise linear approximations to contours and have a variety of applications in image representation and analysis [29]–[31]. In Section III, we analyze the asymptotic performance of a simple prototype wavelet–wedgeprint image coder, nicknamed prototype Wedgelet-SFQ (proto-WSFQ). For the illustrative class of continuous piecewise smooth images containing smooth C^2 regions separated by edges along smooth C^2 contours,[3] we prove that this framework achieves the optimal approximation performance $||X - X_{\widehat{\mathcal{Q}}(N)}||_2^2 \lesssim N^{-2}$ and that the proto-WSFQ coder achieves near-optimal R-D performance $D(R;X) \lesssim (\log R)^2/R^2$.

While this analysis illustrates the motivation and potential benefits of the wedgeprint framework, the application to natural image compression is somewhat more involved. In Section IV, we extend the proto-WSFQ framework into a coder for natural images. As an extended version of SFQ [6] designed to accommodate wedgeprints, our Wedgelet-SFQ (WSFQ) coder is intended *by design* to leverage the existing work in wavelet-based compression; the WSFQ codebook can be viewed as a superset of the SFQ codebook. We identify, however, some practical limitations of the prototype wedgeprints proposed in Section II, and we outline a series of refinements designed to improve their practical compression performance. Among these refinements, we allow local wavelet coding of residual errors remaining from wedgeprint approximations; the tree-structured optimization framework, however, allows us to locally consider the R-D impact of each possible decision at each node. Thus, each wedgeprint is chosen to encode geometry only when it improves the coder's ultimate R-D performance.

Section V then tests the resulting WSFQ coder on natural images. As demonstrated in Fig. 1(d), WSFQ offers the potential for improvements both in visual quality and mse over the state-of-the-art SFQ performance. We conclude in Section VI with a discussion and directions for future research.

[3]Our choice of C^2 smoothness relates to the approximation performance afforded by wedgelets; other smoothness classes can be approximated using geometric atoms containing higher-order polynomial discontinuities [32].

II. Wedgeprints for Modeling and Approximation

State-of-the-art wavelet image coders such as EZW [4], SPIHT [5], and SFQ [6] realize tremendous savings, both in theory and in practice, by using zerotrees over smooth image regions. The idea is simple: construct a quadtree-structured wavelet approximation of an image, scalar quantize the coefficients deemed "significant," and then prune away (replace with zero) those considered "insignificant." For images that are uniformly smoothed away from isolated point singularities, simple zerotree coders are provably optimal (in an asymptotic sense) [12]. We have seen, however, that this optimality does not extend to piecewise smooth images with discontinuities along smooth contours. It simply takes too many "significant" coefficients to construct a usable approximation to an edge at fine scales.

In this section, we introduce a quadtree-based approximation strategy for images that retain much of the structure of the wavelet–zerotree paradigm, but is better suited for images with edge singularities. As before, we use zerotrees to approximate the behavior of wavelet coefficients in smooth regions of the image (thus collapsing an entire subtree into one symbol). Our strategy for wavelet coefficients around edge contours is equally simple: we collapse entire wavelet subtrees into a single *wedgeprint*, a primitive edge function projected onto a wavelet subspace.

A. Wedgelets and Wedgeprints

To begin, consider a simple method for locally approximating an image region around an edge contour. Let X be a piecewise smooth image containing an edge singularity along a C^2-smooth contour β, and let $S_{j,k} \subset [0,1]^2$ be a square subdomain (corresponding to the support of a wavelet $\psi^b_{j,k}$) through which β passes. The contour β divides $S_{j,k}$ into two subregions $S^A_{j,k}$ and $S^B_{j,k}$, as shown in Fig. 2(a). At fine scales, as the sidelength of $S_{j,k}$ becomes smaller, the region $X(S_{j,k})$ becomes "simpler" in two ways.

- The contour β becomes essentially straight as it passes through $S_{j,k}$ and can be approximated by a straight line denoted as ℓ (ℓ represents both a position and an orientation for the line).
- The image becomes essentially flat over the two subregions $S^A_{j,k}$ and $S^B_{j,k}$; $X(S^A_{j,k})$, and $X(S^B_{j,k})$ can be approximated by constants A and B.

Thus, inside $S_{j,k}$, we can approximate $X(S_{j,k})$ using a single *wedgelet* $\mu_{j,k}(t;\ell,A,B)$. A wedgelet (first introduced by Donoho [28]) is a piecewise constant function on $S_{j,k}$ that is discontinuous along the line ℓ, as shown in Fig. 2(b). This wedgelet approximation reduces all of the information in $X(S_{j,k})$ down to three parameters $\{\ell, A, B\}$. We examine

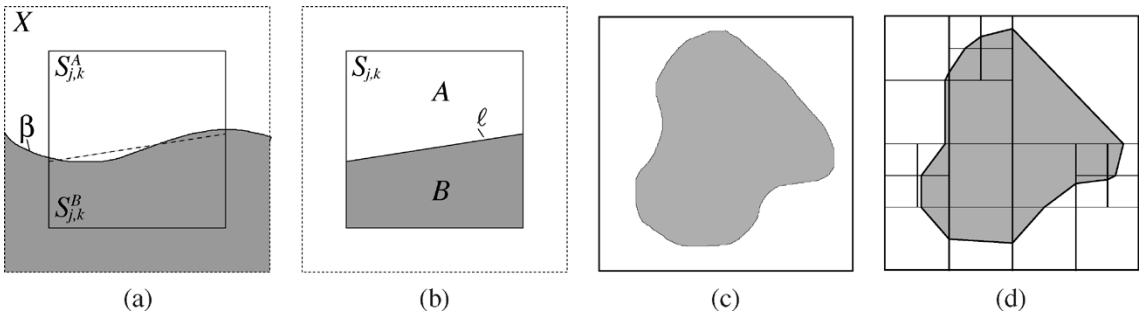


Fig. 2. (a) Square block $S_{j,k}$ of a piecewise smooth image discontinuous along a smooth contour β. (b) Parameterization of wedgelet $\mu_{j,k}(t;\ell,A,B)$: a line index ℓ describes the position and orientation of the edge, and A and B specify the grayscale intensities on each side. (c) Example of a piecewise constant image with a smooth edge contour. (d) A wedgelet tiling divides the domain of an image into dyadic squares and uses a piecewise constant wedgelet in each square to approximate the image.

the accuracy of the wedgelet approximation (with smoothness conditions on β, $X(S^A_{j,k})$, and $X(S^B_{j,k})$) carefully in Section III.

Instead of using a wedgelet in the conventional sense [28] to approximate an image segment directly in the spatial domain, we will use the wedgelet parameters to approximate the image on a wavelet subspace. Our approach is guided by the observation that the wavelet coefficients on subtree $\mathcal{T}^b_{j,k}$ (whose supports nest inside of $S_{j,k}$) should behave similarly to the wavelet coefficients of a wedgelet supported on $S_{j,k}$. Thus, in building an approximation to X we propose to replace all of the wavelets in $\mathcal{T}^b_{j,k}$ in our approximation with a single *wedgeprint* $\rho^b_{j,k}(t;\ell)$, a unit wedgelet orthogonally projected onto the subspace $W^b_{j,k} := \mathrm{span}\{\psi^b_{j',k'} : (j',k') \in \mathcal{T}^b_{j,k}\}$ (and subsequently renormalized)

$$\rho^b_{j,k}(t;\ell) = \frac{\mathrm{Proj}\left(\mu_{j,k}(\cdot;\ell,0,1) \to W^b_{j,k}\right)}{\left\|\mathrm{Proj}\left(\mu_{j,k}(\cdot;\ell,0,1) \to W^b_{j,k}\right)\right\|_2}.$$

The expansion coefficient for $\rho^b_{j,k}(\cdot;\ell)$ is denoted $p^{b;\ell}_{j,k} = \langle X, \rho^b_{j,k}(\cdot;\ell)\rangle$.

B. Wedgeprints as a Wavelet-Domain Model

Using a wedgeprint in an approximation implicitly defines specific values for the wavelet coefficients in the underlying wavelet subtree. To be precise, the approximated value for each wavelet coefficient $w^b_{j',k'} \in \mathcal{T}^b_{j,k}$ is given by

$$\begin{aligned}\widehat{w}^b_{j',k'} &= p^{b;\ell}_{j,k}\left\langle \psi^b_{j',k'}, \rho^b_{j,k}(\cdot;\ell)\right\rangle \\ &= \left\langle \psi^b_{j',k'}, \mu_{j,k}(\cdot;\ell,A,B)\right\rangle \end{aligned} \quad (5)$$

where $p^{b;\ell}_{j,k}$ is proportional to the wedgelet contrast $(B-A)$. In image regions containing edges that are nearly straight, we are able to approximately specify an entire subtree of wavelet coefficients with just a line parameter ℓ and a coefficient $p^{b;\ell}_{j,k}$. In this sense, wedgeprints are analogous to zerotrees—large numbers of wavelet coefficients are efficiently described with a few parameters—but wedgeprints do so in the high-energy regions near edges. (In fact, when $B-A=0$, a wedgeprint is equivalent to a zerotree.) Zerotrees and wedgeprints are thus merely *simple models* for wavelet behavior in smooth and edge regions, respectively. Approximating a wavelet subtree with a wedgeprint ensures a geometrical coherence among the coefficients; as shown in Fig. 3, wedgeprints leave few ringing artifacts around approximated edges in the spatial domain.

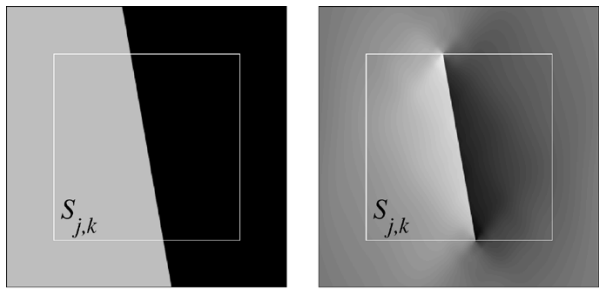


Fig. 3. (a) Portion of an image containing a wedgelet through the block $S_{j,k}$. (b) Spatial domain wedgeprint $\rho^{\mathrm{v}}_{j,k}$ obtained by projecting onto vertical-band wavelet subtree $\mathcal{T}^{\mathrm{v}}_{j,k}$.

The wedgeprint construction is similar to the *footprints dictionary* of Dragotti and Vetterli [27], a collection of scale-space vectors that model wavelet coefficients at singularities in 1-D piecewise polynomial signals. We consider only piecewise constant primitives, however, and we intend to use only one wedgeprint on a given subtree. Our 2-D implementation is also different from the edgeprints presented in [33], where footprints are applied separably to the rows and columns of a 2-D image.

The concept of projecting wedgelets into the wavelet domain can be extended to other geometrical primitives. One possibility would be to use *surflets*, a dictionary of binary functions that are discontinuous along parabolas and higher order polynomials [32]. Of course, we would need more parameters to specify these higher-order primitives, increasing both the number of bits an encoder has to use for a subtree and the computational complexity for matching a primitive to a local region in the image. Another approach, discussed in Section IV, is to project a *tiling* of wedgelets with the same grayscale parameters A, B to approximate the image along curved edges [see Fig. 2(c) and (d)].

C. Wedgeprints as a Sparse Atomic Representation

We now have at our disposal two ways to prune a section of a wavelet quadtree: using a zerotree, we annihilate all of the coefficients in a subtree; using a wedgeprint, we replace all the coefficients in a subtree with those of an ideal edge along a specified line segment. Again, we can recast this process as building up an approximation on a quadtree using wavelets

and wedgeprints. This leads us to a second interpretation for wedgeprints, as an adaptive atomic representation that can be combined with wavelet basis functions to approximate piecewise smooth images.

As in (1), let $X_{\widehat{\mathcal{Q}}}$ be an approximation to X constructed from wavelets and wedgeprints on the quadtrees $\widehat{\mathcal{Q}} = \{\widehat{\mathcal{Q}}^{\mathtt{V}}, \widehat{\mathcal{Q}}^{\mathtt{H}}, \widehat{\mathcal{Q}}^{\mathtt{D}}\}$. We will use wavelets on the interior nodes of the $\widehat{\mathcal{Q}}^b$ and wedgeprints on the leaves:

$$X_{\widehat{\mathcal{Q}}} = \sum_{k \in \mathcal{K}(j_0)} u_{j_0,k}\phi_{j_0,k}(t) + \sum_{b \in \{\mathtt{V},\mathtt{H},\mathtt{D}\}} \left(\sum_{(j,k) \in \mathrm{Interior}(\widehat{\mathcal{Q}}^b)} w^b_{j,k}\psi^b_{j,k}(t) + \sum_{(j,k) \in \mathrm{Leaves}(\widehat{\mathcal{Q}}^b)} p^{b;\ell}_{j,k}\rho^b_{j,k}(t;\ell) \right). \quad (6)$$

A wedgeprint with $\ell = \emptyset$ specifies a zerotree in (6); $p^{b;\emptyset}_{j,k} = 0$. We let $|\widehat{\mathcal{Q}}^b|$ denote the total number of wavelet and wedgeprint atoms preserved on each pruned quadtree.

By augmenting the wavelet approximation with wedgeprints, we can prune the quadtree back to relatively coarse scales both over smooth regions in the image (using zerotrees) and also over regions containing contours that are essentially straight (using wedgeprints). In Section III, we show that for images that are C^2 away from a C^2 contour, the wavelet–wedgeprint approximation (6) has asymptotic behavior

$$\left\| X - X_{\widehat{\mathcal{Q}}(N)} \right\|_2^2 \lesssim N^{-2} \quad (7)$$

a marked improvement over the wavelet rate of N^{-1}—the combined wavelet–wedgeprint representation is demonstrably more sparse for this image class than are wavelets alone. Just as with quadtree-based wavelet approximation [6], there exists a fast dynamic program (the CART algorithm) to find the quadtrees $\widehat{\mathcal{Q}}$ and the wedgeprint parameter ℓ on the leaves of the $\widehat{\mathcal{Q}}^b$ that optimally balance the approximation error against the number of terms.

III. Prototype Wavelet–Wedgeprint Image Coder

In this section, we demonstrate the benefits of wedgeprints by designing a simple prototype wavelet-based image coder that combines wedgeprints and zerotrees and by proving that it achieves near-optimal asymptotic R-D performance on a class of piecewise smooth images. Our prototype proto-WSFQ coder builds on the structure of the zerotree-based SFQ algorithm [6]. Due to their organization on wavelet subtrees, wedgeprints are a natural fit for the quadtree-structured SFQ framework.

To quantitatively characterize the performance of the prototype wavelet–wedgeprint coder, we analyze its asymptotic performance on the class of simple "C^2/C^2" images. An image $X \in C^2/C^2$ consists of two smooth regions separated by a smooth contour

$$X(t_1,t_2) = X_1(t_1,t_2)H_\beta(t_1,t_2) + X_2(t_1,t_2)\,(1 - H_\beta(t_1,t_2))$$

where $X_1, X_2 \in C^2([0,1]^2)$, and H_β is a *Horizon class* [28] image, $H_\beta(t_1,t_2) = 1_{\{t_2 > \beta(t_1)\}}$, where $\beta \in C^2([0,1])$ represents a smooth "horizon" that separates the two image regions.[4]

Our proto-WSFQ coder encodes each wavelet quadtree from the top down. To each node (j,k) in quadtree $\mathcal{Q}^b$, the encoder assigns a label (SQ, ZT, or WP) corresponding to one of three types of action.

1) SQ [scalar quantization]: The wavelet coefficient at node (j,k) is scalar quantized using a predetermined (and uniform) stepsize. The encoder also assigns labels to the node's children $\mathcal{C}_{j,k}$.
2) ZT [zerotree]: The wavelet coefficient at node (j,k) is scalar quantized while all descendants $\mathcal{U}_{j,k}$ are jointly quantized to zero. The encoder does not assign labels to any descendant node.
3) WP [wedgeprint]: All wavelet coefficients in the subtree $\mathcal{T}_{j,k}$ rooted at node (j,k) are jointly quantized using a wedgeprint. The encoder specifies a line segment ℓ and a single projection coefficient $p^{b;\ell}_{j,k}$. The encoder does not assign labels to any descendant node $(j',k') \in \mathcal{U}_{j,k}$.

Our analysis of the proto-WSFQ coder has three key components, each of which is described in detail in the remainder of this section.

- *Approximation*: In Section III-A, we show that for an image $X \in C^2/C^2$, *there exists* an approximation $X_{\widehat{\mathcal{Q}}}$ resulting from pruned wavelet quadtrees $\{\widehat{\mathcal{Q}}^{\mathtt{V}}, \widehat{\mathcal{Q}}^{\mathtt{H}}, \widehat{\mathcal{Q}}^{\mathtt{D}}\}$ using $N := |\widehat{\mathcal{Q}}^{\mathtt{V}}| + |\widehat{\mathcal{Q}}^{\mathtt{H}}| + |\widehat{\mathcal{Q}}^{\mathtt{D}}|$ wavelet–wedgeprint terms with error $\|X - X_{\widehat{\mathcal{Q}}}\|_2^2 \le \mathrm{Const} \cdot N^{-2}$.
- *Quantization*: In Section III-B, we show that we can quantize the approximation $X_{\widehat{\mathcal{Q}}}$ using R bits with total distortion $D(R;X) \le \mathrm{Const} \cdot (\log_2 R)^2 R^{-2}$.
- *Optimization*: In Section III-C, we outline a fast dynamic program to *find* the configuration of labels $\{\mathtt{SQ}, \mathtt{ZT}, \mathtt{WP}\}$ that optimally balances the number of bits used to code the image against the distortion.

Thus, the proto-WSFQ image coder is *near-optimal*; for the class of C^2/C^2 images, the best asymptotic rate that any coder can hope to achieve is $D(R;X) \lesssim R^{-2}$ [12], [14]. The proto-WSFQ coder is also *tractable*; there exists a fast algorithm to find the quantized wavelet–wedgeprint configuration that balances the number of bits used in the representation against the distortion (quantized approximation error).

It can also be shown, although we will not do so here, that the class on which the proto-WSFQ coder achieves $D(R;X) \lesssim (\log_2 R)^2 R^{-2}$ is slightly larger than C^2/C^2. Isolated singularities can be introduced into the curve ("kinks") or the smooth background without changing the asymptotic behavior.

A. Approximation

Consider the quadtrees $\{\mathcal{Q}^{\mathtt{V}}, \mathcal{Q}^{\mathtt{H}}, \mathcal{Q}^{\mathtt{D}}\}$ representing the three wavelet subbands for an image $X \in C^2/C^2$, and fix a finest scale $J > 0$. In this section, we construct [as in (6)] an approximation $X_{\widehat{\mathcal{Q}}(J)}$ of X by pruning each of these quadtrees, using zerotrees and wedgeprints, to maximum depth J. In the

[4]Our analysis is readily extended to the more general *Star* class of images defined in [28].

TABLE I
PROTO-WSFQ APPROXIMATION SUMMARY

scales	type S nodes	type E nodes	# coeffs	error
$0,\ldots,J/2$	wavelet	wavelet	$\sim 2^J$	0
$J/2+1,\ldots,J-1$	prune	wavelet	$\sim 2^J$	$\sim 2^{-2J}$
J	prune	wedgeprint	$\sim 2^J$	$\sim 2^{-2J}$

resulting approximation, we have $N = |\widehat{\mathcal{Q}}^{\mathrm{V}}(J)| + |\widehat{\mathcal{Q}}^{\mathrm{H}}(J)| + |\widehat{\mathcal{Q}}^{\mathrm{D}}(J)| \lesssim 2^J$ terms with error $\|X - X_{\widehat{\mathcal{Q}}(J)}\|_2^2 \lesssim 2^{-2J}$. We construct this approximation treating each of the three quadtrees in an identical manner; we therefore present the remaining discussion for a generic wavelet quadtree $\mathcal{Q}^b$.

We classify the node (j,k) as "type E" if the support of $\psi_{j,k}$ intersects the edge contour β; otherwise we classify (j,k) as "type S." Since the supports of descendant wavelets nest inside the supports of their ancestors, we know that if (j,k) is of type S, then all of its descendants will also be of type S. Similarly, if (j,k) is type E, then all of its ancestors will also be type E.

Since the contour β is smooth, it has finite length. As such, the number of wavelets at scale j that it will touch grows inversely to the sidelength of their support. We can bound the number of type E and type S coefficients at scale j by

$$N_{\mathsf{E}}(j) \lesssim 2^j \quad \text{and} \quad N_{\mathsf{S}}(j) \leq 2^{2j}.$$

The image is uniformly C^2 on the support of a type S wavelet $\psi_{j,k}$; as a result, the coefficient magnitudes decay quickly across scale [9, Ch. 9]. For type E coefficients, the decay is much slower. We have[5]

$$\begin{aligned} |w_{j,k}|^2 &\lesssim 2^{-2j} \qquad \forall\,(j,k) \text{ type E} \\ |w_{j,k}|^2 &\lesssim 2^{-6j} \qquad \forall\,(j,k) \text{ type S}. \end{aligned} \tag{8}$$

In order to correctly balance the number of wavelet and wedgeprint atoms against the approximation errors, we construct our approximation $X_{\widehat{\mathcal{Q}}}$ top-down in three stages (see Table I).

1) *Scales* $j = 0,\ldots,J/2$. At the coarsest scales, we preserve the wavelets at all nodes (using no zerotrees or wedgeprints). This requires a total of $N_1 \lesssim 2^J$ basis elements while introducing no error: $E_1 = 0$.[6]
2) *Scales* $j = J/2+1,\ldots,J-1$. At the intermediate scales, we keep all wavelets at type E nodes while pruning below all type S nodes (using zerotrees at each type S node not already captured by an ancestor's zerotree). For this stage, the total number of wavelets kept is $N_2 \lesssim 2^J$. The total error incurred by pruning below type S nodes at these scales is
$$E_2 \lesssim \sum_{j=\frac{J}{2}+2}^{\infty} N_{\mathsf{S}}(j) 2^{-6j} \lesssim 2^{-2J}.$$
3) *Scale* $j = J$. We prune below all remaining type S nodes at scale J. Thus, we preserve a total of $N_{3,\mathsf{S}} \lesssim 2^J$ wavelet atoms at type S nodes and introduce truncation error $E_{3,\mathsf{S}} \lesssim 2^{-2J}$ below these nodes. For each type E node (J,k) at scale J, we replace the entire wavelet subtree $\mathcal{T}_{J,k}$ with a single wedgeprint. The total number of wedgeprints we use is $N_{3,\mathsf{E}} \lesssim 2^J$. As shown next, the error incurred by replacing the subtrees of wavelets with wedgeprints is
$$E_{3,\mathsf{E}} \lesssim 2^{-2J}. \tag{9}$$
This configuration of scales is carefully chosen for the C^2/C^2 image class to balance the wavelet approximation errors (in smooth regions) against the wedgeprint approximations (in edge regions). In a more general setting, neither the image class nor the edge locations may be known *a priori*, and so a tree-pruning algorithm is required to make the proper decisions (see Section III-C).

Combining Stages 1–3, we use
$$N = N_1 + N_2 + N_{3,\mathsf{S}} + N_{3,\mathsf{E}} \lesssim 2^J \tag{10}$$
wavelet and wedgeprint atoms to construct the approximation $X_{\widehat{\mathcal{Q}}(J)}$ of the image X with total error bounded by
$$\left\|X - X_{\widehat{\mathcal{Q}}(J)}\right\|_2^2 \leq E_1 + E_2 + E_{3,\mathsf{S}} + E_{3,\mathsf{E}} \lesssim 2^{-2J}. \tag{11}$$
Using (10) and (11), we can write the error as a function of the number of terms in the approximation $X_{\widehat{\mathcal{Q}}}$
$$\|X - X_{\widehat{\mathcal{Q}}}\|_2^2 \lesssim N^{-2}. \tag{12}$$
This approximation rate agrees with the optimal bound achievable for any orthonormal basis [34]. Our combined wavelet–wedgeprint dictionary does not constitute a basis, however. Instead, the wavelet–wedgeprint quadtree approximation is *adaptive*, and the locations of the wedgeprints and wavelets are not known *a priori*. For the approximation result to be meaningful, we must carefully account for the complexity of this adaptation. In the next section, we show that we can translate (12) (with the addition of a log factor) into an asymptotic R-D bound.

To establish (9), we return briefly to the spatial domain and examine how wedgelets can locally approximate X around the contour. Let $S_{j,k}$ be the square subregion of the domain of X with sidelength $C_{\text{supp}} 2^{-j}$ corresponding to the region of support of wavelet $\psi_{j,k}$ through which the contour β passes. Inside $S_{j,k}$, we approximate β using a straight line, denoted by ℓ. Since $\beta \in C^2$, there is a strip Ω of width $\lesssim 2^{-2j}$ around the line ℓ that contains β (see Fig. 4). In the subregions of $S_{j,k}$ on either side of the strip (we label these regions $S_{j,k}^A$ and $S_{j,k}^B$), we approximate X with constants A and B. Since X has bounded first derivative away from β, we can find A and B such that
$$\left|X\left(S_{j,k}^A\right) - A\right| \lesssim 2^{-j}, \quad \left|X\left(S_{j,k}^B\right) - B\right| \lesssim 2^{-j}.$$

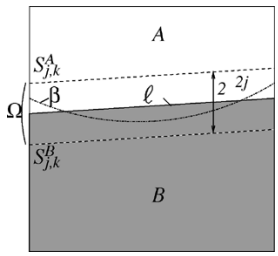


Fig. 4. Analysis of wedgelet approximation on square block $S_{j,k}$.

[5]We assume that the wavelets $\psi_{j,k}$ have at least two vanishing moments.

[6]Because the wavelets form an orthonormal basis, ℓ_2 wavelet-domain error equals L_2 spatial-domain error.

Using a wedgelet $\mu_{j,k}(t;\ell,A,B)$ to approximate $X(S_{j,k})$, we have the errors

$$\left\|X\left(S^A_{j,k}\right)-A\right\|_2^2 \lesssim 2^{-4j}, \quad \left\|X\left(S^B_{j,k}\right)-B\right\|_2^2 \lesssim 2^{-4j} \quad (13)$$

outside of the strip Ω and

$$\|X(\Omega)-\mu_\Omega(t;\ell,A,B)\|_2^2 \lesssim \mathrm{Area}(\Omega) \lesssim 2^{-3j} \quad (14)$$

inside the strip. As a result, $\|X(S_{j,k})-\mu_{j,k}(t;\ell,A,B)\|_2^2 \lesssim 2^{-3j}$.

In Stage 3 above, we replace an entire subtree of wavelets rooted at node (J,k) with a single wedgeprint. We can bound the error of this replacement by translating our wedgelet approximation result into the wavelet domain. Using the Bessel inequality, we have

$$\left\|p^{b;\ell}_{J,k}\rho^b_{J,k}(\cdot;\ell)-\mathrm{Proj}\left(X\to W^b_{J,k}\right)\right\|_2^2 \le \|X(S_{J,k})-\mu_{J,k}(t;\ell,a,b)\|_2^2 \lesssim 2^{-3J}.$$

Thus, the total error for Stage 3 is $E_{3,\mathrm{E}} \lesssim N_{3,\mathrm{E}}2^{-3J} \lesssim 2^{-2J}$.

B. Quantization

In the context of image compression, we are interested more in asymptotic R-D performance than approximation decay. An image coder must spend bits not only on the expansion coefficients, but also on the *addressing information* specifying which atoms were chosen for the approximation. For the combined wavelet–wedgeprint representation, this addressing information takes two forms: we must code the quadtree pruning map underlying the approximation, and we must quantize and encode the line parameters of the wedgeprints used at the quadtree leaves without introducing significant additional error.

The structure of the wavelet–wedgeprint approximation allows us to translate (12) into an R-D bound without too much difficulty. Given the quadtree-based approximation $X_{\widehat{\mathcal{Q}}(J)}$ constructed in the previous section, our simple prototype coder encodes $X_{\widehat{\mathcal{Q}}(J)}$ using $R \lesssim J2^J$ bits with total distortion (approximation error plus quantization error) $D \lesssim 2^{-2J}$.

The encoder starts by coding the symbols tree. Interior nodes of $\widehat{\mathcal{Q}}^b(J)$ are labeled with symbol `SQ`; leaves with zerotrees below are labeled `ZT`; and leaves with wedgeprints below are labeled `WP`. Using a straightforward, top-down algorithm, the encoder spends $R_{\widehat{\mathcal{Q}}} \lesssim |\widehat{\mathcal{Q}}^b| \lesssim 2^J$ bits on the labels.

At each node in $\widehat{\mathcal{Q}}^b(J)$, the encoder quantizes an expansion coefficient (either a wavelet or wedgeprint) using a scalar quantizer with fixed stepsize Δ_J, forming a quantized approximation $\widetilde{X}_{\widehat{\mathcal{Q}}(J)}$. We have $N \le C_N 2^J$ coefficients and suffer total approximation error $\|X-X_{\widehat{\mathcal{Q}}(J)}\|_2^2 \le C_E 2^{-2J}$. To ensure that the quantization error $D_q(J)=\|X_{\widehat{\mathcal{Q}}(J)}-\widetilde{X}_{\widehat{\mathcal{Q}}(J)}\|_2^2$ does not dominate the approximation error, we need $N\Delta_J^2 \le C_E 2^{-2J}$, and so we choose[7] $\Delta_J=\sqrt{C_E/C_N}2^{-3J/2}$.

[7]We assume that the constants C_N and C_E are known to the encoder/decoder pair. Otherwise, they can be calculated from the data and encoded without any effect on the asymptotic results.

With the quantizer step-size fixed, the number of bits we will use for a coefficient depends on its possible range of values. From (8), we have that $|w_{j,k}| \le \mathrm{Const}\cdot 2^{-j}$, and it is easily shown that $|\langle X,\rho_{J,k}(\cdot;\ell)\rangle| \le \mathrm{Const}\cdot 2^{-J}$. Using step size Δ_J requires

$$\mathrm{bins}(j)=\frac{\mathrm{Const}\cdot 2^{-j}}{\Delta_J}=\mathrm{Const}\cdot 2^{\frac{3J}{2}-j}$$

quantization bins at scale j, making the number of bits the encoder spends on each expansion coefficient

$$\mathrm{bits}(j)=\log_2(\mathrm{bins}(j))=\mathrm{Const}+\frac{3J}{2}-j. \quad (15)$$

The total number of bits spent on quantizing expansion coefficients is

$$R_q \le \sum_{j=0}^{J} N(j)\mathrm{bits}(j) \lesssim J2^J \quad (16)$$

where $N(j)$ is the number of wavelets and wedgeprint coefficients used in the approximation $X_{\widehat{\mathcal{Q}}(J)}$ at scale j in $\widehat{\mathcal{Q}}^b(J)$. The total quantization error is of the same order as the approximation error, $D_q \lesssim 2^{-2J}$.

Finally, for each `WP` node, we need to quantize the line parameter ℓ. Using $\mathrm{Const}\cdot J$ bits, we can quantize ℓ (denoting the quantized value $\widetilde{\ell}$) of a unit wedgelet and suffer error $\|\mu_{J/2,k}(t;\ell,0,1)-\mu_{J/2,k}(t;\widetilde{\ell},0,1)\|_2^2 \lesssim 2^{-3J}$ [35], [36]. Thus, we suffer total error $D_\ell \lesssim 2^{-2J}$ while using

$$R_\ell \lesssim J2^J \quad (17)$$

bits to code the wedgeprint line parameters.

Collecting the results above, our prototype image coder uses

$$R=R_{\widehat{\mathcal{Q}}}+R_q+R_\ell \lesssim J2^J \quad (18)$$

bits to code a quantized approximation with total error

$$D=\left\|X-\widetilde{X}_{\widehat{\mathcal{Q}}(J)}\right\|_2^2 \le \left\|X-X_{\widehat{\mathcal{Q}}(J)}\right\|_2^2+D_q+D_\ell \lesssim 2^{-2J}.$$

Writing the distortion as a function of the rate, we have

$$D(R;X) \lesssim \frac{(\log R)^2}{R^2}. \quad (19)$$

Thus, the proto-WSFQ coder comes within a logarithmic factor of the best possible asymptotic R-D performance. The log factor in our final bound (19) is a result of the bounds (16) and (17) growing like $J2^J$ rather than 2^J. It is worth noting that the bound in (17) can be sharpened to $R_\ell \lesssim 2^J$ by encoding the wedgeprint line parameters *jointly* as in [31]. The additional $\log R$ factors are really caused by the quantization of the expansion coefficients—to remove them, we would need $\mathrm{bits}(j)$ in (15) to be something like $J-j$ rather than $3J/2-j$. It is possible that some joint encoding strategy for the wedgeprint coefficients could tighten (16) to $\sim 2^J$, giving us truly optimal asymptotic distortion decay, but we will not pursue this question here.

C. Finding the Optimal Configuration

We have shown the existence of an efficient configuration of labels for each image $X \in C^2/C^2$. In this section, we describe a straightforward *tree-pruning* algorithm that can be used to find the *best* configuration (in an R-D sense) of symbols $\{\texttt{SQ}, \texttt{ZT}, \texttt{WP}\}$ on the quadtree. This technique is a variation on the standard CART algorithm [16], where we seek to optimize the total R-D performance $\Gamma := D + \lambda R$ for a fixed Lagrangian parameter λ. Our R-D optimized approach is a straightforward extension of the two-symbol tree-pruning used in the SFQ coder [6].

For a node (j,k), we let $R_{j,k}^{\texttt{sym}}$ denote the total rate required to encode the subtree $\mathcal{T}_{j,k}$ given that symbol $\texttt{sym} \in \{\texttt{SQ}, \texttt{ZT}, \texttt{WP}\}$ is encoded at node (j,k). Similarly, we let $D_{j,k}^{\texttt{sym}}$ denote the total distortion (approximation and quantization error) that results on the subtree. Using a bottom-up approach, the tree-pruning algorithm chooses a symbol at each node that minimizes the total R-D impact $\Gamma_{j,k}$ on the subtree $\mathcal{T}_{j,k}$. We denote the minimum R-D cost by

$$\Gamma_{j,k}^* := \min_{\texttt{sym} \in \{\texttt{SQ},\texttt{ZT},\texttt{WP}\}} D_{j,k}^{\texttt{sym}} + \lambda R_{j,k}^{\texttt{sym}} \tag{20}$$

and the corresponding rate and distortion by $R_{j,k}^*$ and $D_{j,k}^*$, respectively.

The per-symbol costs at node (j,k) are determined as follows. To compute rate and distortion costs under option $\texttt{SQ}$, we must account for the bits used to encode the particular map symbol (among three options), the costs for scalar quantization of $w_{j,k}$ [see (15)], as well as the (previously determined) costs at the children nodes. For the wavelet coefficient at node (j,k), we denote its quantized value by $\widetilde{w}_{j,k}$. It follows that

$$\begin{aligned} R_{j,k}^{\texttt{SQ}} &= \log_2 3 + \text{bits}(j) + \sum_{(j',k') \in \mathcal{C}_{j,k}} R_{j',k'}^* \\ D_{j,k}^{\texttt{SQ}} &= (w_{j,k} - \widetilde{w}_{j,k})^2 + \sum_{(j',k') \in \mathcal{C}_{j,k}} D_{j',k'}^*. \end{aligned}$$

Under the option $\texttt{ZT}$, the costs are straightforward

$$\begin{aligned} R_{j,k}^{\texttt{ZT}} &= \log_2 3 + \text{bits}(j) \\ D_{j,k}^{\texttt{ZT}} &= (w_{j,k} - \widetilde{w}_{j,k})^2 + \sum_{(j',k') \in \mathcal{U}_{j,k}} w_{j',k'}^2. \end{aligned}$$

Under option $\texttt{WP}$, let $R_{\rho_{j,k}}$ denote the number of bits required to quantize both the wedgeprint line parameter and projection coefficient. For each node $(j',k') \in \mathcal{T}_{j,k}$, we denote the approximated wavelet coefficient by $\widehat{w}_{j',k'}$, as in (5). We then have

$$\begin{aligned} R_{j,k}^{\texttt{WP}} &= \log_2 3 + R_{\rho_{j,k}}, \\ D_{j,k}^{\texttt{WP}} &= \sum_{(j',k') \in \mathcal{T}_{j,k}} (w_{j',k'} - \widehat{w}_{j',k'})^2. \end{aligned}$$

After optimizing the symbol for each node at scale j, the tree-pruning proceeds upward until reaching the tree root. The R-D optimized tree-pruning algorithm can be summarized as follows:

1) *Initialization*. Choose a maximum scale J and set $j \leftarrow J$. For each node (j,k) at scale j, set $R_{j,k}^* = R_{j,k}^{\texttt{ZT}}$ and $D_{j,k}^* = D_{j,k}^{\texttt{ZT}}$.
2) *Bottom-up progression*. If $j > 0$, set $j \leftarrow j-1$. Otherwise, terminate.
3) *Subtree optimization*. For each node (j,k) at scale j, choose the optimal symbol according to (20), and set $R_{j,k}^*$ and $D_{j,k}^*$ accordingly. Then return to Step 2.

Due to the additive nature of the objective cost function (20), this tree-pruning yields the configuration of map symbols that minimizes the total Lagrangian cost Γ. This operating point (R,D) corresponds to the lowest possible distortion achievable by the prototype coder for the total rate R [6]; the target rate can be adjusted by changing the parameter λ. It follows that, by constructing a series of tree-prunings as $\lambda \to 0$, the coder will obey the asymptotic performance (19).[8] The complexity of this algorithm scales with the number of initial quadtree nodes (approximately 2^{2J}, or N^2), given previously computed candidate wedgeprints for each node; Section IV-A discusses this added cost for discrete images. Thus, the proto-WSFQ coder is *tractable*; the near-optimal rate-distortion performance is in fact achievable using the quantized wavelet–wedgeprint configurations obtained with a fast algorithm.

IV. Extension to Discrete Images

The continuous-domain analysis of the previous sections illustrates the fundamental principle underlying wedgeprints: we can construct a sparse, efficient geometric image representation that interfaces naturally with the standard wavelet representation for approximating piecewise smooth images. Using this principle as a guide, we now turn our attention to the more practical problem of compressing images defined on a discrete domain. In this regime, the optimality analysis of algorithms becomes much more difficult; instead, a typical and more pragmatic measure of performance is simply the *operational rate-distortion performance* on natural images [37].

We begin this section by implementing a discrete version of the proto-WSFQ algorithm in Section III. After identifying some limitations with this approach, we propose some practical modifications to the algorithm. In our discussion, we view a discrete image as a partition of the unit square $[0,1]^2$ into square pixels (with a constant intensity defined on each). All notations regarding dyadic squares, quadtrees, and wavelets follow naturally from their continuous-domain analogues.

A. Discrete Wedgeprints

To obtain wedgeprints for use in a discrete image coder, we first must choose a candidate wedgelet for each dyadic square. This is accomplished by searching a discrete dictionary of pixelized wedgelets for the best ℓ_2 fit to the image square. For a given node, we conduct this search on the square $S_{j,k}$ having sidelength exactly 2^{-j}; this corresponds to the "idealized" support of the wavelet basis function $\psi_{j,k}$ and facilitates fast com-

[8] In this example it suffices to initialize each tree-pruning at a scale J that meets (18), with a value R corresponding to the point where the curve (19) has slope $-\lambda$; in practice with a discrete $M \times M$ image, we merely set $J = \log_2 M$.

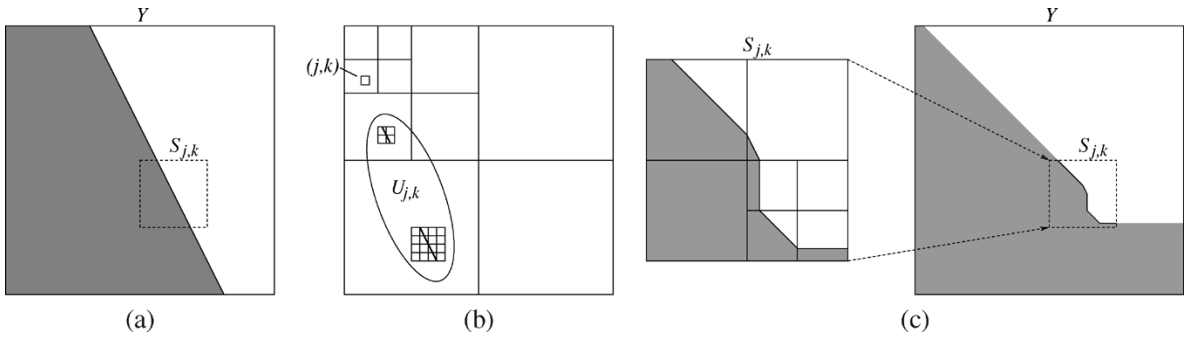


Fig. 5. Obtaining a discrete wedgeprint. (a) Create a temporary image Y, using linear extensions at the border of $S_{j,k}$. (b) Take the wavelet transform of Y, and extract the subtree of wavelet coefficients rooted at node (j, k) (choosing the horizontal wavelet subband in this example). (c) A similar procedure is used for projecting a wedgelet tiling.

Fig. 6. *Cameraman* test image coded at 0.169 bpp. (a) SFQ results, PSNR 26.37 dB. (b) Proto-WSFQ results, PSNR 26.45 dB. (c) WSFQ results using enhanced wedgeprints, PSNR 26.73 dB. White boxes indicate the idealized support of each wedgeprint used in coding.

putation.[9] Because some wavelet basis elements in $\mathcal{T}_{j,k}$ extend slightly beyond this square, however, we extend the wedgelet outside the square to infer the wavelet coefficient values. In practice, to minimize border artifacts, we use the simple procedure demonstrated in Fig. 5(a) and (b).

B. Proto-WSFQ Implementation

Using discrete wedgeprints, we implement the proto-WSFQ coder outlined in Section III. As a basis for this implementation, we use the discrete SFQ coder [6], which combines wavelets and zerotrees and uses a bottom-up tree-pruning optimization. In general, the extension of SFQ to accommodate wedgeprints is intuitive and straightforward; we list in Appendix A the relevant details of our modifications.

For our experiments, we implement both SFQ and proto-WSFQ using biorthogonal 10–18 wavelets [39] in MATLAB.[10] Fig. 6(a) and (b) shows the 256 × 256 *Cameraman* test image compressed using SFQ and proto-WSFQ at a bit rate of 0.169 bits per pixel (bpp). (In each case, we encode a four-level wavelet decomposition.)

We see in this example that 25 distinct wedgeprints are used in the proto-WSFQ coding [white boxes in Fig. 6(b) indicate the approximate spatial-domain support for each chosen wedgeprint]. In addition to providing a small improvement in visual quality in these regions (compared to the ringing artifacts of SFQ), the proto-WSFQ algorithm also provides a modest gain of 0.08 dB in PSNR.[11] For comparison purposes, compressing this image using the wavelet-based JPEG-2000 coder [19] yields a PSNR of 25.39 dB, well below the SFQ performance, and also introduces strong ringing artifacts.

C. Discussion

The previous experiment demonstrates a tangible yet modest gain for the proto-WSFQ algorithm; such performance is typical on discrete images. This marginal gain is likely due to the fact that natural images may not be well-modeled as C^2/C^2 functions: image curves may not be C^2 and may be corrupted by blurring and pixelization. In addition, the asymptotic optimality derived in Section III does not necessarily translate to efficient performance at low bit rates.

We do observe, however, in such practical experiments that wedgeprints *are chosen* in the tree-pruning proto-WSFQ optimization. We take this as a validation of the fundamental

[9]At first glance, the complexity for this search is linear in the number of pixels and the size of the dictionary; however there exist techniques to amortize the cost across scales and also to obtain fast approximate solutions [38], [41].

[10]Using Daubechies 7–9 wavelets [15], we observe slightly lower performance for both SFQ and WSFQ, although the *relative* performance of WSFQ to SFQ is slightly better.

[11]Peak signal-to-noise ratio (PSNR) is a commonly used measure of distortion; assuming a maximum possible pixel intensity of 255, PSNR $:= 10\log_{10}(255^2/\text{mse})$.

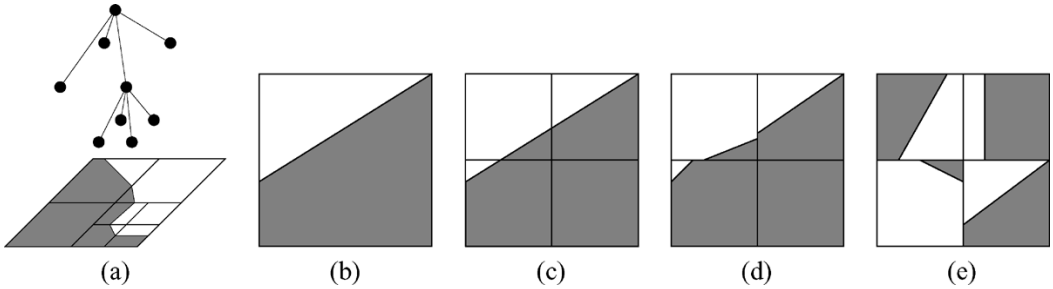


Fig. 7. (a) Wedgelet tiling can be interpreted as a pruned quadtree, where each node includes a set of wedgelet parameters and leaf nodes specify the pictured wedgelets. (b) Wedgelet on a dyadic square. (c) MGM predictions for the square's children, considered to be their most likely configuration. (d) A slightly less likely configuration for the children. (e) A significantly less likely configuration.

principle underlying wedgeprints, but we also see the results as an indication that such simple wedgeprints are not suited to the challenges of natural images. Because the proto-WSFQ framework is quite flexible and is designed to leverage the performance of the SFQ algorithm, one solution is clear: by designing wedgeprints better suited to the geometric features of natural images, we can expect to realize better practical coding gains.

In the remainder of this section, we propose a series of minor modifications designed to improve the practical performance of wedgeprints; compression results incorporating these changes into our WSFQ coder are presented in Section V. In general, these enhancements are not derived from rigorous mathematical models (which are elusive for real-world images); rather, they are the result of our own experimentation. For the sake of illustration, the proposed changes are intended as direct responses to the particular shortcomings of *wedgelet*-based "prints"; other possibilities are further discussed in the Conclusion.

D. Wedgelet Tilings

We begin by developing a more explicit geometrical primitive for wedgeprint projections. Consider a quadtree node (j,k). In Section II-A, we constructed a wedgeprint based on a single wedgelet on the square $S_{j,k}$. Of course, this is not the only available representation for the geometry on $S_{j,k}$. We propose to use not a single wedgelet but a *tiling* of smaller wedgelets to more precisely describe the geometry on $S_{j,k}$. As shown in Fig. 7(a), a dyadic wedgelet tiling can be interpreted as a pruned quadtree [28], where each node $(j',k') \in \mathcal{T}_{j,k}$ includes a set of wedgelet parameters describing the corresponding dyadic square $S_{j',k'} \subset S_{j,k}$. Leaf nodes of the pruned quadtree are used to assemble the picture of the wedgelet tiling, and dividing a leaf node into four children yields a finer approximation to a contour. Discrete wedgeprints can be obtained from wedgelet tilings using the procedure shown in Fig. 5(c).

Wedgelet tilings allow more explicit representations of geometry on dyadic squares; compared to a single coarse wedgelet, a multiscale tiling generally offers an approximation with lower distortion. To fully exploit these benefits in image compression, we must also be able to efficiently encode such a tiling. As demonstrated in Fig. 7, there exist strong correlations among wedgelet parameters in a given tiling. We have developed a multiscale geometry model (MGM) to capture these dependencies [31], [35]. The MGM is a simple Markov-1 model that defines a probability distribution on a wedgelet line parameter conditioned on the line parameter of its parent node. This model permits a straightforward top-down, predictive algorithm for encoding a wedgelet tiling and supports R-D optimized tree-pruning, similar to the algorithm discussed in Section III-C, for determining the optimal tiling. In exchange for the R-D benefits of wedgelet tilings, computational complexity is increased: the optimal tiling must be recomputed for each value of the R-D parameter λ, and because the choice of wedgelet at each node may also be R-D optimized (rather than simply using the best ℓ_2 fits), the complexity of each tree-pruning becomes quadratic in the wedgelet dictionary size (however, this can be greatly reduced by accepting a near-optimal solution). The MGM and its associated algorithms for discrete wedgelet processing are described in depth in [31] and [35].

E. Edge Profile Smoothness

As an additional practical extension to our geometric representation, we include a notion of smoothness across the profile of an edge. While a sharp edge makes an abrupt step transition from A to B in profile [see Fig. 2(b)], a smooth (or blurred) edge may take several pixels to transition from A to B.

We enhance our geometric representation by including a parameter $\sigma_{j,k}$ that specifies the profile smoothness of the wedgelet tiling on $S_{j,k}$. This parameter is used as an index into a discrete dictionary of smoothing filters having a variety of shapes and supports; after first constructing the approximation on $S_{j,k}$ using sharp wedgelets, we then apply the blurring filter indicated by $\sigma_{j,k}$. The coder reserves the option of operating with sharp wedgelets, but on occasion the blurring filter may provide a worthwhile reduction in distortion for a minimal cost in rate.

An iterative algorithm allows us to choose the parameter $\sigma_{j,k}$ for a given node. Optimal tree-pruning is difficult because the effects of the smoothing filter are not localized to the dyadic square of each wedgelet. Thus, the tree-pruning simply uses sharp wedgelets at each iteration; based on the resulting wedgelet tiling, we then search the smoothness dictionary for the smoothing filter that yields the closest match to the image square. The tree-pruning then repeats using sharp wedgelets, except that we adjust the Lagrangian penalty at fine scales to reflect the estimated smoothness for the underlying edge (when the edge is blurred, the accuracy of a fit from a sharp wedgelet becomes less important).

Fig. 8. (a) Synthetic piecewise smooth test image containing a smooth discontinuity: Images coded at 0.10 bits per pixel, (b) JPEG-2000, PSNR 30.93 dB, (c) SFQ, PSNR 32.92 dB, and (d) WSFQ algorithm combining wavelets and wedgeprints, PSNR 34.27 dB.

F. Texture Compression

Even with these geometric enhancements, wedgeprints do not offer a complete representation for all of the information contained in a wavelet subtree. In (13) and (14), we saw for the C^2/C^2 image model that within wedgeprint blocks, the geometric error (inside the "strip") dominated the error in the surrounding smooth regions. In practice, however, an image may contain texture adjacent to an edge that may not be well approximated by the piecewise constant wedgeprint model. In principle, on each dyadic square it would be desirable to encode only the geometry using a wedgelet tiling and encode the remaining information using an alternative representation. Of course, this separation is difficult, if not impossible, to implement properly.

We propose the following technique to capture the information neglected by the wedgelet tiling: following the encoding of a wedgeprint at a subtree node, we compute the resulting subtree of residual wavelet errors and encode this residual subtree using SFQ (the specific details of this modification are included in Appendix B). The ultimate coded subtree of wavelet coefficients can be viewed as its own wedgeprint, combining a geometric primitive with a set of additional wavelet corrections.

We note that the residual wavelet coefficients will contain two fundamental types of information: ridge-like geometric artifacts resulting from the limited precision of the wedgelet tiling, and texture-like features away from the geometry. While the residual SFQ approach will be faced with coding both types of information, it is free to neglect those features that it cannot efficiently encode. In fact, by using a ZT symbol at a subtree root node, the residual SFQ encoder may elect not to correct any residual errors.[12]

Ultimately, for any particular node, the two stages of this approach (first encoding a R-D optimized wedgelet tiling and then using SFQ for residual coding) may fail to yield the strictly optimal *joint* tiling/SFQ strategy for encoding that node. We stress, however, that the encoder will compare this tiling/SFQ option with the standard SFQ options (quantization and zerotree), and so wedgeprints will be selected for coding only when they improve the coder's ultimate R-D performance. These decisions are made locally, as opposed to a global two-stage approach that might encode geometry and then residual. As demonstrated in Section V, the fact that such tiling/SFQ nodes are chosen in the ultimate tree-pruned WSFQ configuration is confirmation that these wedgeprints are helpful in reducing the coding cost.

[12]When the coder does correct for residual errors, it risks the reintroduction of "ringing" artifacts due to wavelet quantization of the ridge-like features. In practice these are moderate compared to the original SFQ artifacts; see Section V for results.

V. Experimental Results

To highlight the performance of WSFQ using the wedgeprint enhancements in Section IV-D–IV-F, we first consider a synthetic 256 × 256 image consisting of a sharp Horizon-class image added to a lowpass-filtered Brodatz grass (D9) texture pattern.[13] Fig. 8 shows the uncompressed image as well as the compressed JPEG-2000, SFQ, and WSFQ versions at 0.10 bpp; for both SFQ and WSFQ we encode a four-level wavelet decomposition. As a simple example using a natural image, we also consider the 1024 × 1024 *Wet Paint* image (available at [18]). Fig. 9 shows the compression results at 0.010 bpp, using a five-level wavelet decomposition for SFQ and WSFQ. Table II contains a summary of the performance, listing PSNR values for each coder and detailing the number of quantized wavelets and wedgeprints.

As expected, in regions where WSFQ chooses wedgeprints, ringing artifacts are noticeably reduced compared to the JPEG-2000 and SFQ versions (see Fig. 1 for a close-up version of the *Wet Paint* results). Moreover, due to the strong, simple geometry of the artificial image, WSFQ offers a substantial gain in PSNR: 1.35 dB over the state-of-the-art SFQ technique and 3.34 dB over JPEG-2000.[14] For the *Wet Paint* example, WSFQ offers a considerable improvement of 0.45 dB over both JPEG-2000 and SFQ. We note also that WSFQ offers a practical advantage by encoding explicit geometric information about the image. This information may be useful in situations that benefit from a physical understanding of the scene.

Fig. 10 compares the performance of SFQ and WSFQ against JPEG-2000 for the standard *Peppers*, *Cameraman*, and *Lena* test images. The WSFQ result for *Cameraman* is also shown in Fig. 6(c). In general, both algorithms outperform JPEG-2000 at most rates; for images containing strong geometric features, WSFQ typically outperforms SFQ by 0.2–0.3 dB at low rates. At higher rates, however, WSFQ has diminishing performance gains relative to SFQ. This behavior has several possible causes. At these high rates, for example, it is essential to accurately

[13]The USC-SIPI Image Database. sipi.usc.edu/database/

[14]By decreasing the energy of the texture, we observe gains up to several decibels above standard SFQ.

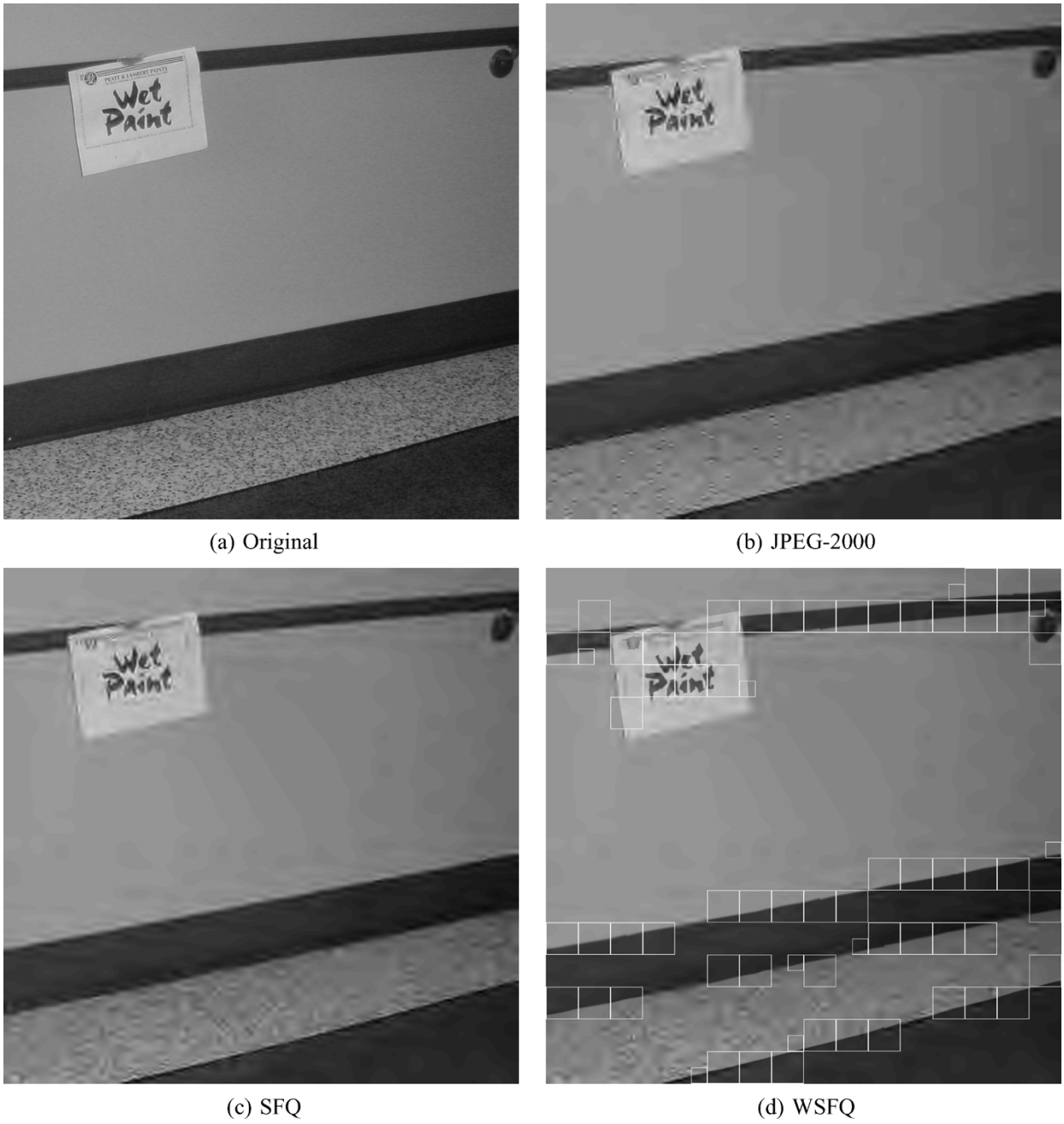


Fig. 9. (a) *Wet Paint* test image. Images coded at 0.010 bpp: (b) JPEG-2000, PSNR 29.77 dB, (c) SFQ, PSNR 29.77 dB, and (d) WSFQ, PSNR 30.22 dB. White boxes indicate the idealized support of each selected wedgeprint. See Fig. 1 for a close-up zoom.

TABLE II
PERFORMANCE OF WSFQ CODER COMPARED TO JPEG-2000 AND SFQ

	Synthetic, 0.10 bpp	*Wet Paint,* 0.010 bpp
JPEG-2000 PSNR	30.93dB	29.77dB
SFQ PSNR	32.92dB	29.77dB
Number of quantized wavelets	2032	3448
WSFQ PSNR	34.27dB	30.22dB
Number of quantized wavelets	1592	1332
Distinct wedgeprints	17	67
Number of quantized residual wavelets	236	300

encode textures very near edges. In addition, wedgelet tilings may not offer high enough precision to code natural instances of geometry at high rates. Similar challenges are presented by images that contain softer edges with surrounding textures, such as *Lena*, for which WSFQ gains are more modest (below 0.15 dB) relative to SFQ. Such considerations can be incorporated into future implementations of WSFQ, as discussed in the Conclusion.

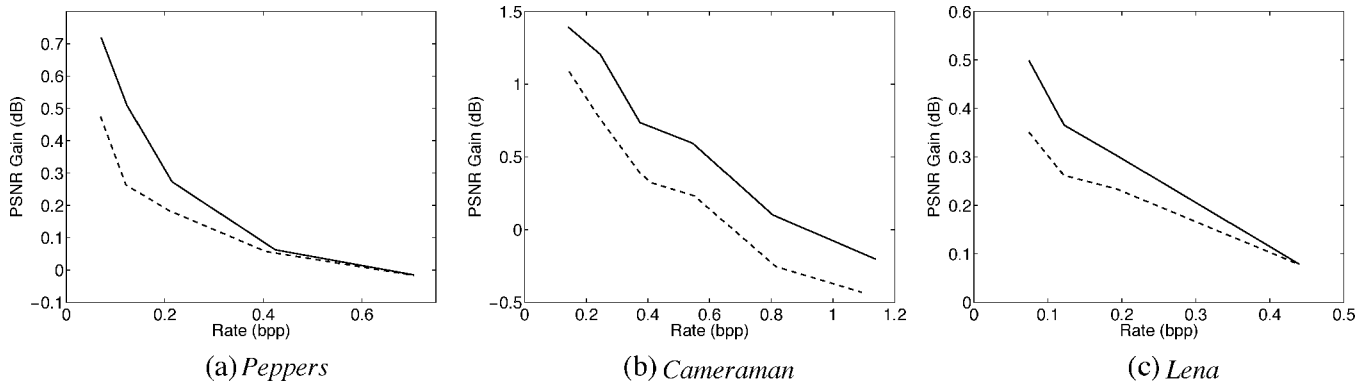


Fig. 10. Performance improvement of WSFQ (solid line) and SFQ (dashed line) relative to JPEG-2000 compression for (a) *Peppers*, (b) *Cameraman*, and (c) *Lena* test images.

VI. Conclusion

Wedgeprints provide a flexible, adaptive, and powerful tool for geometric image modeling, approximation, and compression. Viewed as an atomic representation, wedgeprints allow sparse representation of piecewise smooth images. Viewed as a wavelet-domain modeling framework, our work highlights an important misconception regarding transform-domain image compression. In particular, *sparsity* is not essential for efficient compression of transform coefficients. Indeed, all complete transforms provide the same information—it is simply the *accessibility of convenient models* (of which sparsity is one) that enables efficient compression. In our case, despite the lack of wavelet sparsity, the quadtree-structured organization of the 2-D wavelet transform, combined with the localization of the wavelet basis functions, was sufficient to develop an effective wavelet-domain model that allowed efficient compression.

Our approach leads to many natural extensions. For example, geometric primitives may be further extended to capture more sophisticated instances of geometry. More important, the concept of wedgeprints may be generalized beyond geometry to other types of image features. In fact, any efficient localized image description may be projected to a tree-structured wavelet-domain representation in similar fashion; the process of projection allows the critical information to be encoded in the more convenient domain. As we have done with WSFQ, the two-symbol SFQ tree-pruning algorithm may be easily extended to incorporate a multitude of options for each node. The primary obstacles of implementing such extensions are the increase in computational complexity, the added bit rate required to specify each symbol, and the support of the wavelet basis functions (as encountered in Section IV-A). In the future, we hope to examine these issues further while extending WSFQ.

Appendix A
Wedgelet-SFQ Implementation Details

We present here the relevant implementation details of the WSFQ coding algorithm and optimization scheme. Many of these details follow naturally from the SFQ algorithm [6].

The relevant innovations in this paper concern the encoding of wavelet coefficients; any standard technique may be used to additionally encode the scaling coefficients. We compress the scaling coefficients in a raster scan, predicting each coefficient from its quantized causal neighbors. We quantize and encode the prediction errors, using a quantization stepsize optimized for a generalized Gaussian distribution [7].

A. WSFQ Quantization

WSFQ encodes each wavelet quadtree $\mathcal{Q}^b$ in a single pass from the top down using a strategy similar to the prototype coder of Section III. For a small savings in bit rate, we make the following adjustment: each map symbol indicates the quantization strategy only for *descendants* of that node and not for the wavelet coefficient at the node itself (the quantization scheme for a given wavelet coefficient is actually specified at one of its ancestors). To be precise, the labels $\{\texttt{SQ}, \texttt{ZT}, \texttt{WP}\}$ correspond to the following actions.

1) `SQ` [scalar quantization]: Each child $w_{j',k'} \in \mathcal{C}_{j,k}$ is quantized according to a pre-determined uniform stepsize Δ. Quantization bin indices are encoded using adaptive arithmetic coding [40]. An additional map symbol is subsequently encoded at each child node to describe the quantization of its descendants.
2) `ZT` [zerotree]: The descendants $\mathcal{U}_{j,k}$ are set to zero. As such, the encoder does not encode labels for any node $(j',k') \in \mathcal{U}_{j,k}$.
3) `WP` [wedgeprint]: The encoder specifies a wedgelet for the square $S_{j,k}$ and uses the resulting wedgeprint to represent the descendant wavelet coefficients in $\mathcal{U}_{j,k}$ (see Fig. 5).[15] The encoder does not encode labels for any node $(j',k') \in \mathcal{U}_{j,k}$.

At the coarsest wavelet scale j_0, we tie together small collections of quadtree root nodes and encode a symbol describing their collective quantization.

B. WSFQ Tree-Pruning

Before encoding the map symbols, the WSFQ coder must determine the proper quantization strategy. This strategy is a combination of two primary factors: the quantization stepsize Δ and the configuration of map symbols. As in Section III-C, we fix a value for the Lagrangian parameter λ and seek the strategy that minimizes the overall R-D cost $\Gamma := D + \lambda R$.

[15]Up to three subbands may request a wedgelet on the same square. In such a case, the wedgelet is encoded only once (to be used for wedgeprints in multiple subbands), but for simplicity, we do not reduce the anticipated cost $R_{\rho_{j,k}}$ in each subband.

To minimize Γ for a fixed λ, we consider several possible values for Δ and determine the optimal map configuration for each scenario. For a fixed value of Δ, we use a method similar to the tree-pruning algorithm of Section III-C. However, due to the entropy coding of quantization bins as well as the predictive coding technique for map symbols, the rate costs in our scheme are not strictly localized to subtrees. As discussed below, we use a generalization of the two-phase SFQ tree-pruning algorithm to obtain a configuration of map symbols that is *near-optimal* in terms of R-D cost: 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 map symbols. After pruning for several values of Δ, we select as our strategy the value of Δ (and the corresponding configuration of map symbols) that minimizes Γ.

Phase I (Quantization Costs): Phase I starts pruning from the bottom of the tree and proceeds upward. The algorithm initially assumes that all coefficients are scalar quantized and must make decisions regarding whether to insert zerotree or wedgeprint symbols. 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" wavelet coefficients (those to be quantized). This yields an estimate of the entropy (and hence rate cost) of each scalar quantization.

During each iteration of the Phase I optimization, only nodes currently labeled significant are examined. The coder has three options at each such node (j,k): maintain the scalar quantization (symbol SQ), create a zerotree (ZT), or create a wedgeprint (WP). The coder chooses the option that minimizes the total R-D impact $\Gamma_{j,k}$ on the subtree $\mathcal{T}_{j,k}$.

The per-symbol costs at node (j,k) are assigned as follows. For symbols SQ and ZT, Phase I ignores the rate required to encode each map symbol

$$\begin{aligned} R_{j,k}^{\mathtt{SQ}} &= \sum_{(j',k')\in\mathcal{C}_{j,k}} -\log_2\left[p(\widetilde{w}_{j',k'})\right] + R_{j',k'}^{*} \\ D_{j,k}^{\mathtt{SQ}} &= \sum_{(j',k')\in\mathcal{C}_{j,k}} (w_{j',k'}-\widetilde{w}_{j',k'})^2 + D_{j',k'}^{*} \\ R_{j,k}^{\mathtt{ZT}} &= 0 \\ D_{j,k}^{\mathtt{ZT}} &= \sum_{(j',k')\in\mathcal{U}_{j,k}} w_{j',k'}^2. \end{aligned}$$

Unlike the cases SQ and ZT, which we expect to be relatively common, we expect relatively few symbols WP to be encoded, since each wedgeprint represents many possibly significant coefficients. Each symbol WP, therefore, requires a nontrivial number of extra bits to encode. We find it useful, then, to add a rough estimate of this added cost to the wedgeprint rate $R_{\rho_{j,k}}$ before assigning the Phase I costs

$$\begin{aligned} R_{j,k}^{\mathtt{WP}} &= R_{\rho_{j,k}}, \\ D_{j,k}^{\mathtt{WP}} &= \sum_{(j',k')\in\mathcal{U}_{j,k}} (w_{j',k'}-\widehat{w}_{j',k'})^2. \end{aligned} \tag{21}$$

As in Section III-C, the optimization proceeds bottom-up. Once the top of the tree is reached, the process repeats bottom-up if any significant map symbols have changed. Convergence is guaranteed because the number of significant coefficients can only decrease. Despite the complications presented by the entropy coding, we believe (as in [6]) that the Phase I tree-pruning algorithm converges to a configuration of map symbols that is near-optimal in terms of R-D cost.

Phase II (Map Symbol Costs): Phase II adjusts the tree-pruning to better account for the specific costs of encoding map symbols. These costs are estimated 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 among nearby coefficients indicate the likelihood of symbol ZT, while high variances indicate the likelihood of symbols SQ and WP. We encode whether a particular symbol is ZT according to this expected behavior, and we then distinguish between symbols SQ and WP 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 ZT. A switch is made if the savings in map symbol rate exceeds the loss in Phase I R-D efficiency. This Phase II implementation is a relatively straightforward extension of the SFQ version [6].

APPENDIX B
RESIDUAL TEXTURE COMPRESSION

As described in Section IV-F, we implement standard SFQ compression on each residual subtree resulting from a wedgeprint. That is, encoding a symbol WP at node (j,k) involves the following four steps:

1) encode the wedgelet tiling on $S_{j,k}$ and compute the wedgeprint $\rho_{j,k}$;
2) compute the residual error subtree $\mathcal{E}_{j,k} = \{e_{j',k'} : (j',k') \in \mathcal{U}_{j,k}\}$ where $e_{j',k'} := w_{j',k'} - \widehat{w}_{j',k'}$;
3) prune and encode the subtree $\mathcal{E}_{j,k}$ using symbols SQ and ZT of SFQ with the same parameter λ, quantization stepsize Δ, and probability model p used elsewhere;
4) add the quantized residual $\widetilde{\mathcal{E}}_{j,k}$ to the wedgeprint coefficients for the final encoded values.

We adjust the costs (21) before assigning the Phase I symbol at node (j,k). Because these costs now may vary with the changing probability model p, we do allow the tree-pruning algorithm to analyze WP nodes on subsequent iterations, but to ensure convergence we only allow a WP symbol to change to ZT.

ACKNOWLEDGMENT

While this paper was in press, Hyeokho Choi passed away. We will forever remember his broad vision, his keen insights, and our lively discussions. His legacy will live on through his many contributions to the signal processing community. The authors would like to thank A. Cohen, R. DeVore, and M. Orchard for many helpful discussions and the anonymous reviewers for helpful comments.

REFERENCES

[1] D. L. Donoho, "Denoising by soft-thresholding," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 613–627, May 1995.

[2] H. Choi and R. G. Baraniuk, "Multiscale image segmentation using wavelet-domain hidden Markov models," *IEEE Trans. Image Process.*, vol. 10, no. 9, pp. 1309–1321, Sep. 2001.

[3] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 7, pp. 710–732, Jul. 1992.

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

[5] 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.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.

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

[7] S. LoPresto, K. Ramchandran, and M. T. Orchard, "Image coding based on mixture modeling of wavelet coefficients and a fast estimation-quantization framework," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, Mar. 1997, pp. 221–230.

[8] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards, and Practice*. Boston, MA: Kluwer, 2002.

[9] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed. San Diego, CA: Academic, 1999.

[10] R. A. DeVore, B. Jawerth, and B. J. Lucier, "Image compression through wavelet transform coding," *IEEE Trans. Inf. Theory*, vol. 38, no. 2, pp. 719–746, Mar. 1992.

[11] D. L. Donoho, "Unconditional bases are optimal bases for data compression and for statistical estimation," *J. Appl. comput. Harm. Anal.*, vol. 1, no. 1, pp. 100–115, Dec. 1993.

[12] A. Cohen, W. Dahmen, I. Daubechies, and R. DeVore, "Tree approximation and optimal encoding," *J. Appl. comput. Harm. Anal.*, vol. 11, pp. 192–226, 2001.

[13] A. Cohen, I. Daubechies, O. G. Guleryuz, and M. T. Orchard, "On the importance of combining wavelet-based nonlinear approximation with coding strategies," *IEEE Trans. Inf. Theory*, vol. 48, no. 7, pp. 1895–1921, Jul. 2002.

[14] 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. Nashville, TN: Vanderbilt Univ. Press, 1999.

[15] I. Daubechies, *Ten Lectures on Wavelets*. New York: SIAM, 1992.

[16] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

[17] D. Marr, *Vision*. San Francisco, CA: Freeman, 1982.

[18] Standard Test Images. [Online] Available: www.dsp.rice.edu/~wakin/images/

[19] M. D. Adams. The JasPer Project Home Page. [Online] Available: www.ece.uvic.ca/~mdadams/jasper/

[20] 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. Norwell, MA: Kluwer, 2001.

[21] L. Peotta, L. Granai, and P. Vanderghenyst, "Very low bit rate image coding using redundant dictionaries," in *Proc. Wavelets X, SPIE's 48th Annu. Mtg.*, vol. 5207, San Diego, CA, 2003, pp. 228–229.

[22] R. Shukla, P. L. Dragotti, M. Do, and M. Vetterli, "Rate-distortion optimized tree-structured compression algorithms for piecewise polynomial images," *IEEE Trans. Image Process.*, vol. 14, no. 3, pp. 343–359, Mar. 2005.

[23] E. Candès and D. L. Donoho, "New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities," *commun. Pure Appl. Math.*, vol. 57, pp. 219–266, 2004.

[24] M. N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091–2016, Dec. 2005.

[25] E. Le Pennec and S. Mallat, "Sparse geometric image representations with bandelets," *IEEE Trans. Image. Process.*, vol. 14, no. 4, pp. 423–438, Apr. 2005.

[26] F. Arandiga, A. Cohen, M. Doblas, R. Donat, and B. Matei, "Sparse representations of images by edge adapted nonlinear multiscale transforms," in *Proc. IEEE Int. Conf. Image Processing*, Barcelona, Spain, Sep. 2003, pp. 701–704.

[27] P. L. Dragotti and M. Vetterli, "Wavelet footprints: theory, algorithms and applications," *IEEE Trans. Signal Process.*, vol. 51, no. 5, May 2003.

[28] D. L. Donoho, "Wedgelets: Nearly-minimax estimation of edges," *Ann. Stat.*, vol. 27, pp. 859–897, 1999.

[29] D. L. Donoho and X. Huo, "Beamlets and multiscale image analysis," in *Multiscale and Multiresolution Methods*, T. J. Barth, T. Chan, and R. Haimes, Eds. New York: Springer, 2002, vol. 20, Springer Lecture Notes in Computer Science, pp. 149–196.

[30] R. Willett and R. Nowak, "Platelets: A multiscale approach for recovering edges and surfaces in photon-limited medical imaging," *IEEE Trans. Med. Imag.*, vol. 22, no. 3, pp. 332–350, Mar. 2003.

[31] J. K. Romberg, M. B. Wakin, and R. G. Baraniuk, "Multiscale Geometric Image Processing," *Proc. SPIE Visual Commun. Image Process.*, vol. 5150, pp. 1265–1272, 2003.

[32] V. Chandrasekaran, M. B. Wakin, D. Baron, and R. Baraniuk, "Compression of higher dimensional functions containing smooth discontinuities," in *Proc. Conf. Inf. Sci. Syst.*, Princeton, NJ, Mar. 2004.

[33] P. L. Dragotti and M. Vetterli, "Footprints and edgeprints for image denoising and compression," in *Proc. IEEE Int. Conf. Image Processing*, Thessaloniki, Greece, Oct. 2001, pp. 237–240.

[34] R. A. DeVore, "Nonlinear approximation," *Acta Numer.*, vol. 7, pp. 51–150, 1998.

[35] J. K. Romberg, "Multiscale geometric image processing," Ph.D. dissertation, Dept. Elect. Comput. Eng., Rice Univ., Houston, TX, 2003.

[36] M. N. Do, P. L. Dragotti, R. Shukla, and M. Vetterli, "On the compression of two-dimensional piecewise smooth functions," in *IEEE Int. Conf. Image Proc.*, vol. 1, Thessaloniki, Greece, Oct. 2001, pp. 14–17.

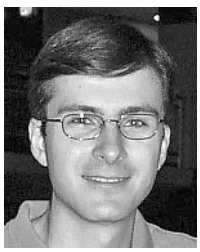
[37] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, Nov. 1998.

[38] M. B. Wakin, D. L. Donoho, H. Choi, and R. G. Baraniuk, "High-resolution navigation on nondifferentiable image manifolds," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 5, Philadelphia, PA, Mar. 2005, pp. 1073–1076.

[39] M. Tsai, J. Villasenor, and F. Chen, "Stack-run image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 5, pp. 519–521, Oct. 1996.

[40] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," *commun. ACM*, vol. 30, no. 6, pp. 520–540, Jun 1987.

[41] J. K. Romberg, M. B. Wakin, and R. G. Baraniuk, "Multiscale wedgelet image analysis: Fast decompositions and modeling," in *Proc. IEEE Int. Conf. Image Process.*, vol. 3, Sep. 2002, pp. 585–588.



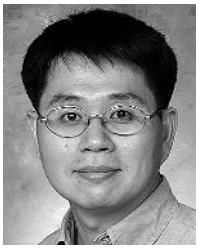
Michael B. Wakin (S'01) received the B.S. degree in electrical and computer engineering and the B.A. degree in mathematics in 2000 (summa cum laude), and the M.S. degree in electrical and computer engineering in 2002, all from Rice University, Houston, TX.

He is currently a graduate student in the Electrical and Computer Engineering Department, Rice University, and has held visiting positions at Baylor College of Medicine in 2003 and the UCLA Institute for Pure and Applied Mathematics in 2004. His research interests include multiresolution signal and image processing.

Mr. Wakin received a Texas Instruments/Nokia Distinguished Graduate Fellowship from Rice University in 2000 and a National Science Foundation Graduate Research Fellowship in 2001.

Justin K. Romberg (M'03) received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from Rice University, Houston, TX, in 1997, 1999, and 2003, respectively.

He spent the summer 2000 at the Xerox Palo Alto Research Center, Palo Alto, CA, and the fall 2003 at the Laboratoire Jacques-Louis Lyons, Paris, France. He is currently a Postdoctoral Scholar in applied and computational mathematics at the California Institute of Technology, Pasadena.



Hyeokho Choi (SM'04) received the B.S. degree in control and instrumentation engineering from Seoul National University Seoul, Korea, in 1991 and the M.S. and Ph.D degrees in electrical engineering from the University of Illinois, Urbana-Champaign, in 1993 and 1998, respectively.

In 1998, he joined Rice University, Houston, TX, as a Research Associate. From 2000 to 2005, he was a Research Professor at Rice University. In Fall 2005, he joined North Carolina State University, Raleigh, as an Assistant Professor in the Electrical and Computer Engineering Department. His research interests were in the area of signal and image processing and computer vision.

Dr. Choi received the Presidential Honor at Seoul National University at graduation in 1991 and the Texas Instruments Postdoctoral Fellowship in 1998. Dr. Choi was an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING.

Richard G. Baraniuk (F'02) received the B.Sc. degree from the University of Manitoba, Winnipeg, MB, Canada, in 1987, the M.Sc. degree from the University of Wisconsin, Madison, in 1988, and the Ph.D. degree from the University of Illinois, Urbana-Champaign, in 1992, all in electrical engineering.

After spending 1992–1993 at the Ecole Normale Supérieure, Lyon, France, he joined Rice University, Houston, TX, where he is currently the Victor E. Cameron Professor of Electrical and Computer Engineerng and Founder of the Connexions Project. He spent sabbaticals at Ecole Nationale Supérieure de Télécommunications, Paris, in 2001 and Ecole Fédérale Polytechnique de Lausanne, Switzerland, in 2002. His research interests in signal and image processing include wavelets and multiscale analysis, statistical modeling, and sensor networks. He received a NATO postdoctoral fellowship from NSERC in 1992.

Dr. Baraniuk received the National Young Investigator award from NSF in 1994, a Young Investigator Award from ONR in 1995, the Rosenbaum Fellowship from the Isaac Newton Institute of Cambridge University in 1998, the C. Holmes MacDonald National Outstanding Teaching Award from Eta Kappa Nu in 1999, the Charles Duncan Junior Faculty Achievement Award from Rice in 2000, the ECE Young Alumni Achievement Award from the University of Illinois in 2000, and the George R. Brown Award for Superior Teaching at Rice in 2001 and 2003.