RICE UNIVERSITY

Image Compression using Multiscale Geometric Edge Models

by

Michael B. Wakin

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

APPROVED, THESIS COMMITTEE:

Richard Baraniuk, Chair
Professor of Electrical and Computer Engineering

Robert Nowak
Assistant Professor of Electrical and Computer Engineering

Michael Orchard
Professor of Electrical and Computer Engineerng

Houston, Texas

May, 2002

ABSTRACT

Image Compression using Multiscale Geometric Edge Models

by

Michael B. Wakin

Edges are of particular interest for image compression, as they communicate important information, contribute large amounts of high-frequency energy, and can generally be described with few parameters. Many of today's most competitive coders rely on wavelets to transform and compress the image, but modeling the joint behavior of wavelet coefficients along an edge presents a distinct challenge. In this thesis, we examine techniques for exploiting the simple geometric structure which captures edge information. Using a multiscale wedgelet decomposition, we present methods for extracting and compressing a cartoon sketch containing the significant edge information, and we discuss practical issues associated with coding the residual textures. Extending these techniques, we propose a rate-distortion optimal framework (based on the Space-Frequency Quantization algorithm) using wedgelets to capture geometric information and wavelets to describe the rest. At low bitrates, this method yields compressed images with sharper edges and lower mean-square error.

Acknowledgments

A majority of the work presented in this thesis was accomplished with the help of Justin Romberg and Hyeokho Choi. I greatly appreciate their assistance and their insight. I would like to thank the members of my committee, Dr. Robert Nowak and Dr. Michael Orchard, for helping to create an exceptional atmosphere for research in our department. Thanks also to my advisor, Dr. Richard Baraniuk, for his encouragement and for his remarkable enthusiasm. Finally, I would like to thank my wife, Laura, for her unending support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A successful technique for compression of natural images must efficiently treat the various types of commonly occurring features, namely: edges, smooth regions, and textures. Edges represent abrupt changes in intensity that persist along straight or curved contours. Smooth regions are characterized by slowly varying intensities; textures contain a collection of localized intensity changes.* Edges are of particular interest for compression, as they contribute significant information to the viewer, account for a great deal of energy in the frequency domain, and can generally be described with few parameters.

A majority of today's most efficient image compression algorithms [1–5] transform the image to the wavelet domain and then quantize the wavelet coefficients. Wavelets are indeed well-suited to represent smooth and textured regions of images, but wavelet-based descriptions of edges are highly inefficient. In the high-frequency subbands, which contain the largest number of wavelet coefficients, a majority of the energy is concentrated in regions surrounding the dominant edges of the image. Figures 1.1 and 1.2 demonstrate this effect using the Cameraman test image.

The use of wavelets in image processing arose from the success of wavelets when applied to 1-dimensional signals. Notably, wavelets offer efficient representations of point

*Most textures can be viewed as sets of many small edges separating smooth regions. Due to the practical difficulties of considering these edges explicitly, and because this thesis does not focus on texture analysis or synthesis, we reserve the general name "texture" for such complicated regions.

Figure 1.1 Original 256×256 Cameraman image.

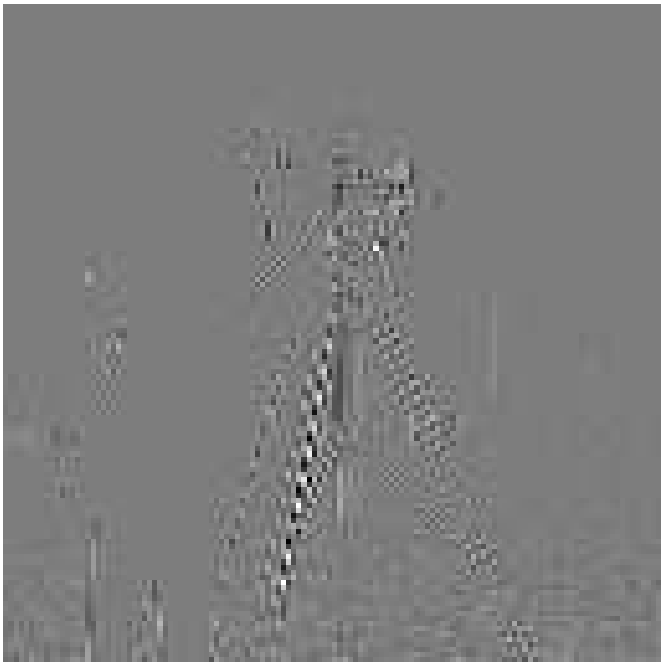


Figure 1.2 Vertical high-pass subband of Cameraman wavelet decomposition.

singularities in 1-d signals. This is because, as the scale j of the wavelet transform increases, the number of significant coefficients due to a singularity grows as $O(j)$. Nonlinear approximations can easily be made with such few coefficients. In 2-dimensional signals, wavelets again offer efficient descriptions of point singularities, but not for edges, which can be viewed as singularities along contours. In this case, the number of significant coefficients due to an edge singularity grows as $O(2^j)$. Indeed, all of these significant coefficients must be accurately transmitted to the decoder in order to faithfully reproduce the edges. Similar to the Gibbs phenomenon with Fourier coefficients of a 1-d step edge, disrupting the coherent alignment of these wavelet coefficients will introduce ringing into the image. The careful modeling of the dependence among these numbers becomes a critical issue in compression performance.

Edges are low-dimensional objects in images; they can generally be described quite accurately with very few parameters. For example, the simple image in Figure 1.3 shows a single straight edge separating regions of intensity 0 and 1. While a discrete version of this image may contain an arbitrarily large number of pixels (N^2 pixels for an $N \times N$ image), it can be precisely described by only two parameters: the orientation and location of the straight edge. Thus, the set of all images in this single-edge class forms a 2-dimensional manifold embedded in the space $\mathbb{R}^{N^2}$ of possible images. Accurate description of one of these images (in $\mathbb{R}^{N^2}$) merely requires pinpointing its location along the manifold. A thorough analysis of natural images [6] has confirmed that high-energy blocks of pixels typically occur very close to a low-dimensional edge manifold such as this.

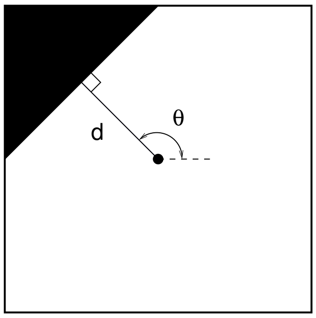


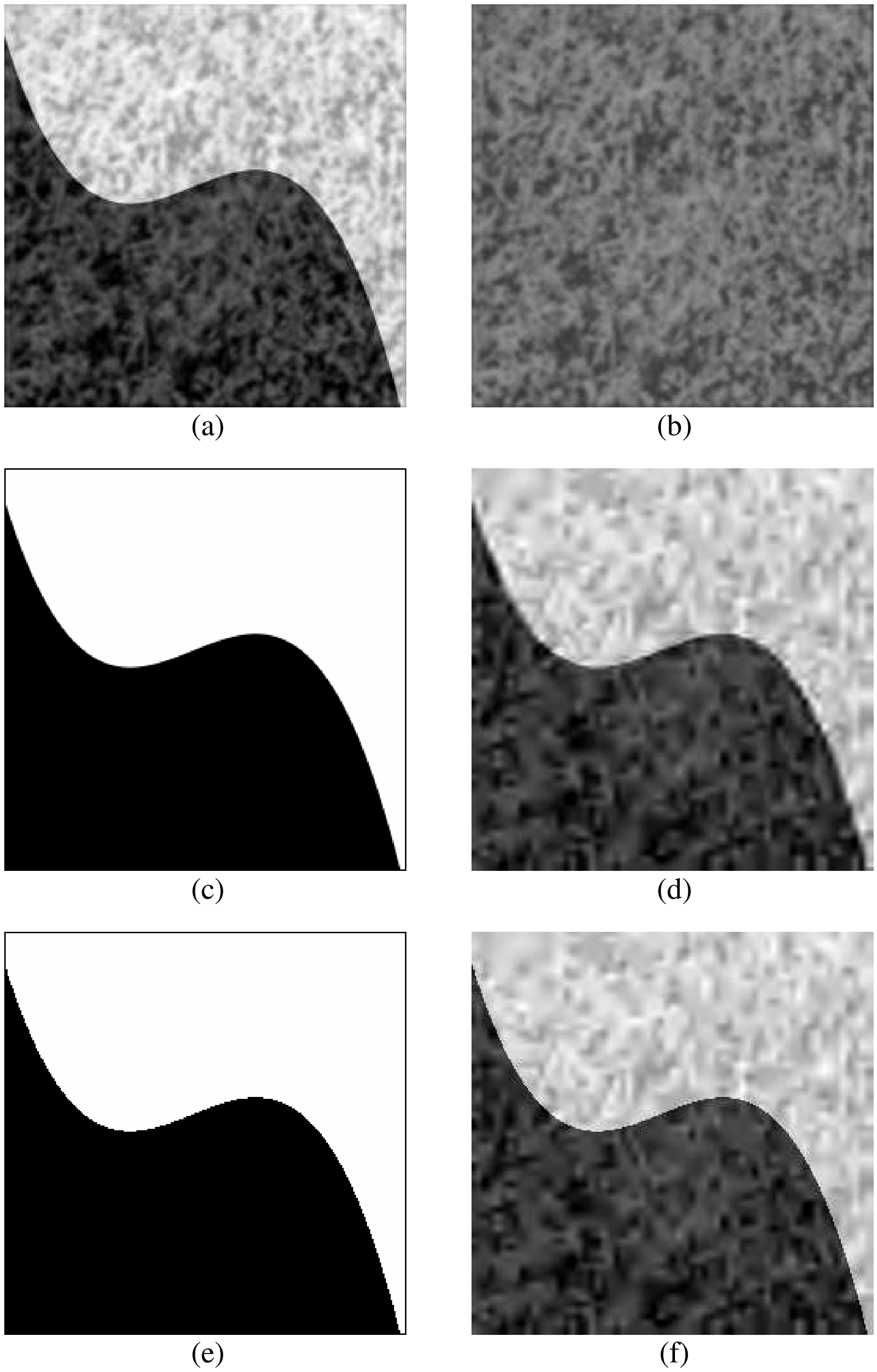
Figure 1.3 Single straight edge in an $N \times N$ image block.

Similar arguments can be made for general edges in images: along curved contours with varying degrees of sharpness, the pixel intensity values near edges are highly dependent and are constrained by the geometric structure of the edge. By "geometric structure", we specifically refer to the regularity of the underlying contour (such as a linear segment, or a contour which satisfies some smoothness criteria) which admits the potential for mathematical modeling. The low-dimensional dependency imposed on pixels in the spatial domain translates to the wavelet coefficients which describe those pixels (i.e., the wavelet coefficients whose basis functions overlap with the edge contour). Specifically, while many wavelet coefficients are typically required to fully describe an edge, a well-defined structure exists among these coefficients. Though well-defined, the dependency is quite complicated, and most coders fail to fully exploit it. The Estimation-Quantization (EQ) coder [3] models the increased variance of coefficients near an edge, but it does not explicitly model their values. The Least-Square EQ (LS-EQ) coder [7] attempts to predict actual wavelet coefficients from their neighboring coefficients, but its approach is limited to relatively straight edges, and it requires a large amount of training information to adapt to the local structure of an edge. Many other wavelet coders, such as the Embedded Zerotree

Figure 1.4 JPEG-2000 compression of Cameraman image using 0.24 bits per pixel (bpp). The strong ringing around edges is typical of wavelet-based coders at low bitrates.

Wavelet (EZW) coder [1], take no specific consideration of edges; they simply model the rough statistical behaviors of coefficients. The typical result of wavelet-based compression is shown in Figure 1.4: because the dependencies among wavelet coefficients are not properly exploited, quantization destroys the coherence which creates clean edges, and strong ringing artifacts are introduced near edges.

The failure of wavelets to efficiently describe edges, along with the inherent simplicity of edge structures and the success of wavelets in describing smooth and textured regions motivates a simple twofold approach to compression. Specifically, a *geometry-based* compression scheme could be developed which efficiently compresses edge information, while wavelets could be used to compress the remaining smooth and textured regions. Figure 1.5 illustrates the potential benefits of such an approach. Figure 1.5(a) shows a simple



(f)

Figure 1.5 (a) Synthetic 256×256 image consisting of (b) textures and (c) edge. (d) JPEG-2000 compression, 0.118 bpp, PSNR = 27.40dB. (e) Edge compressed with simple chain code, 0.012 bpp. (f) Compressed edge + JPEG-2000 compressed texture, total 0.114 bpp, PSNR = 27.98dB.

artificial image containing both texture (Figure 1.5(b)) and an edge (Figure 1.5(c)). With a target bitrate of 0.118 bpp, applying the JPEG-2000 * compression scheme to the original image yields the result in Figure 1.5(d). Taking explicit consideration of the contour, however, greater compression performance may be attained. Figure 1.5(f) shows the result of a two-stage approach: a simple chain code is used to compress the edge contour using 0.012 bpp, and JPEG-2000 is used to compress the texture with the remaining bits. Chain codes are one of many possible techniques for compression of geometric information; we introduce more advanced techniques in subsequent chapters. Clearly, better compression performance, in terms of visual ringing artifacts and PSNR, † can be obtained through geometry-based compression.

Proper implementation of a geometry-based compression scheme should satisfy the following practical conditions:

G1. *Parsimony:* The geometric dictionary should offer parsimonious descriptions of edges, with few coefficients, so that it yields clean, sharp approximation of edges, even at low bitrates.

G2. *Regularity:* The technique for geometric modeling must exploit the inherent structure of the edge; unlike wavelet-based compression, the ultimate compression performance should depend on the regularity of the underlying edge contour, not simply on its length.

---

*JPEG-2000 software, based on the JPEG-2000 Part-1 standard, was obtained from the JasPer Project home page: *http://www.ece.ubc.ca/~mdadams/jasper/*.

†Peak Signal-to-Noise Ratio (PSNR) represents a common numerical measure of distortion; assuming a maximum possible intensity of 255, $PSNR = 10 \log_{10} \frac{255^2}{MSE}$.

G3. *Geometric R/D Optimality:* The geometric compression should be optimized in terms of its rate-distortion (R/D) performance, with a distortion metric which reflects the ultimate image distortion.

G4. *Global R/D Optimality:* The geometry-based scheme must fit into a comprehensive framework for compression of natural images (with smooth and textured regions), and the overall rate-distortion performance should be optimized.

This thesis examines techniques for natural image compression using geometric edge modeling; techniques are discussed which meet each of the above criteria. Based upon Donoho's wedgelet dictionary [8], we propose a multiscale geometry-based framework for approximation and compression of edge information. We then address the practical and important issues of integrating such a scheme into a natural image coder. Finally, we propose a framework (using the Space-Frequency Quantization algorithm [4]) for natural image compression which exploits the geometric structure of edges, but optimizes its representations in a global rate-distortion sense.

Chapter 2 introduces the fundamental concepts of wedgelets and the relevant concepts of geometric modeling. Chapter 3 offers a first attempt at natural image compression using wedgelets to code a cartoon sketch of the image's edges. Chapter 4 extends these concepts to a globally-efficient compression framework, which builds upon the SFQ coder and uses wedgelets more effectively. Chapter 5 offers suggestions for improving these compression techniques, and Chapter 6 concludes.

Chapter 2

Wedgelets

We choose wedgelets as a tool for approximation and compression of edge information. Wedgelets approximate curved contours using an adaptive piecewise-linear representation, whose geometry can be easily understood. Furthermore, the wedgelet dictionary can be conveniently arranged to allow efficient compression.

Wedgelets were first introduced by Donoho [8]. To fit our purpose, we present here a slightly modified definition and parameterization of wedgelets.

2.1 The Wedgelet Dictionary

A *wedgelet* is a square, dyadic block of pixels containing a picture of a single straight edge. Figure 2.1 shows an example of a wedgelet. Each wedgelet is fully parameterized by five numbers:

d: (edge location) normal distance from the edge to block center

θ: (edge orientation) measure of the angle from horizontal to the normal line

m_1, m_2: (shading) grayscale intensity values on each side of the edge

N: (block size) number of pixels in one row/column of the wedgelet

Technically, the first four parameters describe an edge on a continuous domain. Obtaining the discrete (pixelized) version of a wedgelet requires a model. Motivated by simplicity,

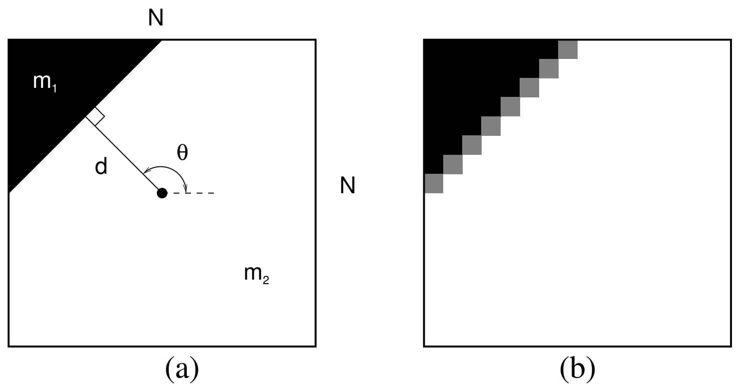


Figure 2.1 (a) Parameterization of a wedgelet on an $N \times N$ image block: orientation (θ), offset (d), and grayscale intensity values (m_1 and m_2). (b) Example of a wedgelet for pixelized image block, $N = 16$.

and based on a rough approximation of the point-spread function of a camera lens, we assume that each pixel contains the averaged intensity of the continuous function on that support. For a given $N \times N$ block of pixelized data, then, one can obtain an estimate of the best wedgelet fit – the set of parameters $\left(\widehat{\theta}, \widehat{d}, \widehat{m_1}, \widehat{m_2}\right)$ which describe a wedgelet with the closest mean-squared fit to the data. Section 2.2 discusses methods for estimating these parameters.

The *wedgelet dictionary* is the dyadically organized collection of all possible wedgelets. It also consists of dyadic constant blocks which are parameterized only by their mean m and their size N. As illustrated in Figure 2.2, contours in an image may be approximated by a *wedgelet decomposition*: a chain of wedgelets chosen from this dictionary. In general, long, straight edges are well-approximated using large wedgelets. A series of smaller wedgelets may be required to approximate curved segments of an edge. These wedgelets, though, clearly have a dependency which is related to the geometry of the edge contour.

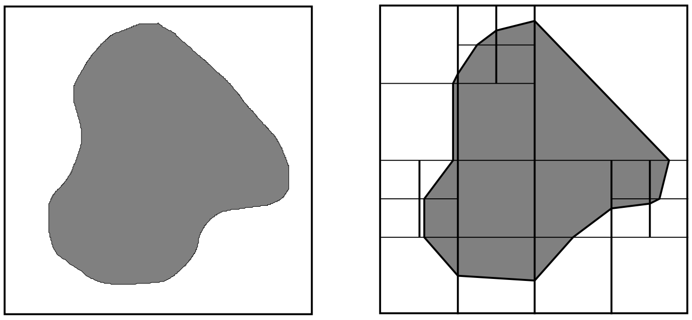


Figure 2.2 (a) Artificial image. (b) Wedgelet decomposition. Each dyadic block in the tiling is constant or contains a single straight edge.

A compression scheme based on the wedgelet representation requires a model which captures the dependency among neighboring wedgelet fits; we refer to this as "geometric modeling". Ultimately, geometric models may be developed for particular classes of curves, in order to fully exploit the regularity of those classes. In subsequent chapters, we introduce actual compression schemes using wedgelets.

2.2 Wedgelet Estimation

Obtaining an accurate wedgelet approximation of a contour requires a technique for estimating wedgelet parameters which fit the pixelized data. A standard criterion, which agrees with the commonly used distortion metric for image compression, is to seek the set of parameters which minimize the l^2 distance from the wedgelet approximation to an $N \times N$ block of pixel data.

The estimation problem may be interpreted as follows. The set of possible wedgelets (each described by four parameters) forms a nonlinear four-dimensional subspace in the space $\mathbb{R}^{N^2}$ of possible $N \times N$ pixel blocks. Finding the best wedgelet fit reduces to projecting the data onto this subspace.

Many possibilities exist for estimating these parameters. The key to practical implementation, however, is finding a fast algorithm. Fast but suboptimal solutions may be obtained using elements of detection theory [9] or by exploiting the phase relationships of the complex wavelet transform [10]. More accurate estimates may be obtained through an analysis of the block's Radon transform. Finally, by restricting the wedgelet dictionary to a carefully chosen discrete set of orientations and locations, the inner products of all wedgelets may be quickly computed [11]. The beamlet transform [12] yields another method of computing these inner products for a slightly different set of discrete wedgelets. For examples presented in this thesis, we use the Radon-domain technique for estimation of continuous wedgelet parameters. Chapter 5 discusses possible improvements to our compression scheme using a discrete wedgelet dictionary.

Chapter 3

Compression via Edge Cartoon

In this chapter, we present a preliminary compression scheme for natural images using wedgelets for geometric compression of edge information. This two-stage scheme (first presented in [13]) is based upon a simple image model:

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

In Figure 1.5, for example, the edge cartoon is illustrated in Figure 1.5(c), while the residual texture is shown in Figure 1.5(b). The edge cartoon contains the dominant edges of the image, separated by constant regions. By representing the image's primary geometric information, this cartoon is quite similar to the "raw primal sketch" described by Marr [14].

Figure 3.1 shows the basic operation of the two-stage process. In Section 3.1, we describe how to use wedgelets to estimate the dominant edges and code a sketch of the edge cartoon. The resulting scheme produces compressed images with clean, sharp edges at low bitrates. In Section 3.2, then, we discuss methods of using wavelets to compress the residual texture information. We address the practical issues which arise in compressing natural images; this helps to develop a more efficient compression framework in Chapter 4.

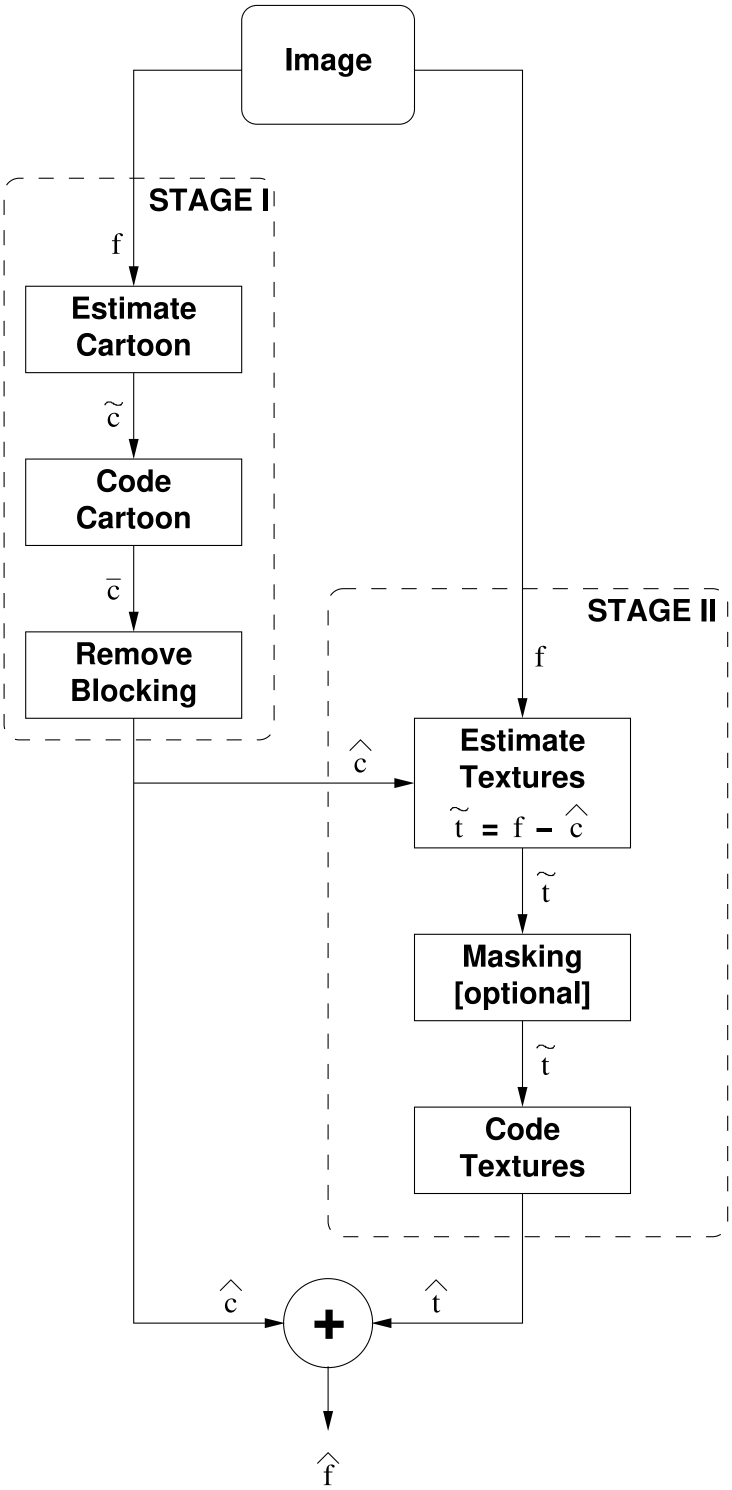


Figure 3.1 Two-stage cartoon/texture compression scheme. The removal of blocking artifacts is specific to our cartoon compression scheme (see Section 3.1). Masking is optional and is discussed in Section 3.2.2.

3.1 Stage I: Estimation and Compression of the Edge Cartoon

As described in Section 2.1, a wedgelet decomposition offers a piecewise-linear approximation to a contour. Similarly, an entire image may be approximated using a wedgelet decomposition. The resulting image resembles a "cartoon sketch" – it contains approximations of the image's dominant edges, and spaces between the edges are filled with constant values. Key to efficiently compressing such a sketch is a framework for choosing the proper wedgelet decomposition. Most importantly, the sizes of wedgelet blocks should be chosen intelligently. In our cartoon compression scheme, we begin with a full dyadic tree of wedgelets. Each node n of the tree is associated with the wedgelet parameters $\left(\theta^{(n)}, d^{(n)}, m_1^{(n)}, m_2^{(n)}\right)$ which give the best l^2 fit to the data in the corresponding image block. From this tree, and under our compression framework which is described below, we seek the wedgelet decomposition (or "tree-pruning") which minimizes the Lagrangian rate-distortion cost. Donoho's CART algorithm [8] provides a methodology for choosing the proper wedgelet decomposition. Given a compression scheme from which rates for each wedgelet may be computed, the CART algorithm yields the optimal tree-pruning in a rate-distortion sense. The picture $\widetilde{c}$ described by optimal the tree-pruning represents the Stage I estimation of the edge cartoon.

Figure 3.2 shows an example of a pruned quadtree, as returned by the CART algorithm. Leaf nodes are designated E (for those which contain an edge) or C (for those which are constant). Interior nodes are designated I. We code the pruned wedgelet tree using a top-down predictive framework. Every node has an associated set of wedgelet parameters,

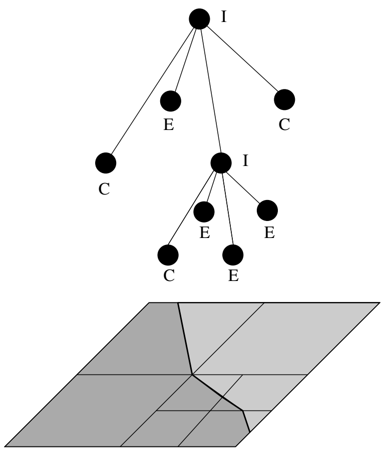
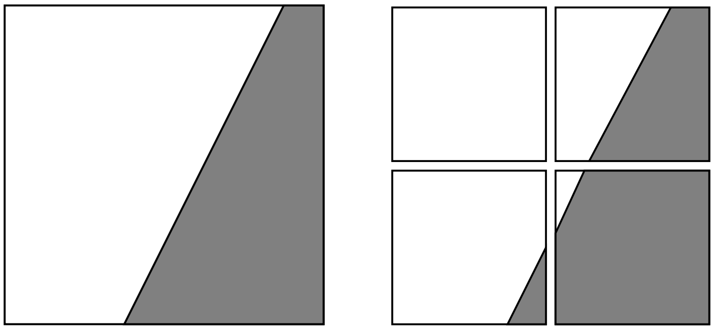


Figure 3.2 Pruned wedgelet quadtree. Interior nodes are labeled I, while leaf nodes are E (edge) and C (constant). The cartoon sketch is constructed according to the leaf nodes.

and these parameters are transmitted to the decoder, even though the cartoon sketch is constructed only from the leaf nodes. Wedgelet parameters of interior nodes are useful for predicting wedgelet parameters of their children.

The coder transmits all information in a single pass, starting with the root node of the wedgelet tree. Three types of information must be sent: (1) a symbol from $\{E, I, C\}$ denoting the state of each node, (2) edge parameters (d, θ) when appropriate, and (3) grayscale values (m) or (m_1, m_2). After designating a node E or C, no further information is coded regarding its descendants in the tree.

For a given node to be coded, we predict its edge parameters and grayscale values based on the previously coded parameters of its parent (necessarily an I node). This causality assures that the decoder can make the same predictions. We make the prediction based on a simple spatial intuition: the parent's wedgelet is divided dyadically to predict the wedgelets of its four children, as demonstrated in Figure 3.3. This provides a prediction for all edge parameters at the node; the coder transmits only the deviations from the true



(a) (b)

Figure 3.3 Multiscale wedgelet prediction. (a) Parent wedgelet. (b) Predicted children.

wedgelet parameters (the prediction errors).

We model each prediction error as a beta(p, q) random variable, with zero mean and symmetric distribution ($p = q$). Quantization is performed using a simple uniform quantizer with deadzone. For each variable to be coded, a beta shape parameter $\widehat{p}$ is estimated from previously coded instances of the same parameter, and a table lookup (using $\widehat{p}$ and the overall desired R/D slope λ) provides the optimal quantization step sizes. The coder easily computes the bin probabilities and uses them for arithmetic coding [15] of each quantization bin index.

After coding the pruned wedgelet tree, we translate it into the cartoon sketch. The wedgelets provide the locations of the edges in the cartoon, and grayscale values describe the shading in regions away from the edges. Figure 3.4 shows an example on the Cameraman image. Because neighboring constant blocks may have slightly different grayscale values, we smooth such regions to prevent the creation of phantom edge artifacts at the block boundaries. Letting $\bar{c}$ be the cartoon sketch with blocking artifacts, we define $\delta(x, y)$ to be the normal distance from pixel (x, y) to the nearest wedgelet-specified edge in $\bar{c}$. For each pixel (x, y) we define the set

$$V_{x,y} = \{(x', y') : \sqrt{(x - x')^2 + (y - y')^2} < \delta(x, y)\}.$$

which represents a circular * set of pixels centered at (x, y) that do not overlap any wedgelet-specified edge. Constructing the smoothed cartoon sketch $\widehat{c}$, we let

$$\widehat{c}(x, y) = \begin{cases} \bar{c}(x, y) & \text{if } \delta(x, y) \leq 1.5, \\ \frac{1}{|V_{x,y}|} \sum_{(x', y') \in V_{x,y}} \bar{c}(x', y') & \text{if } \delta(x, y) > 1.5 \end{cases}$$

for each pixel (x, y). As shown in Figure 3.5, this smoothing preserves all desired edges from the wedgelet sketch but removes blocking artifacts at the borders of wedgelet blocks. Because the smoothing algorithm is known to both the encoder and decoder, and because the smoothed version of the cartoon image better represents the desired features, we consider the smoothed version $\widehat{c}$ to be the actual coded representation of the cartoon sketch.

3.2 Stage II: Compression of Residual Textures

After transmitting the approximate wedgelet cartoon $\widehat{c}$, we wish to compress the textures of the image. In practical settings, however, the true collection of textures t is not available; it must be estimated. Because the texture compression occurs after the cartoon coding, a straightforward method of estimating the texture is simply to use the residual: $\widetilde{t} = f - \widehat{c}$.

*In practice, we use a square set $V_{x,y}$ for efficient computation.

Figure 3.4 Cartoon sketch $\bar{c}$ obtained from the coded wedgelet parameters, 0.12 bpp, PSNR = 22.28dB. Note the blocking artifacts at the wedgelet borders.

Figure 3.5 Final coded cartoon sketch $\widehat{c}$, PSNR = 22.52dB. Smoothing removes blocking artifacts while preserving the coded edges.

Unfortunately, errors in the wedgelet approximation and its lossy encoding will create ridge-like artifacts in the residual image $\widetilde{t}$. These artifacts do not correspond directly to natural features of the image; they are merely the difference between the true edges and the sketch. This section introduces some possible approaches for coding the residual and dealing with these artifacts.

3.2.1 Wavelets

Perhaps the most straightforward technique for residual compression is simply to code the entire residual $\widetilde{t}$ using a standard wavelet coder (such as a zerotree coder [1]). Unfortunately, wavelets are generally not well-suited to compression of the residual image. While a cartoon sketch may contain a majority of the edge information, ridge artifacts in the residual image may still contain a significant amount of energy. Because these long, thin artifacts have a geometrical structure (similar to that of the original edge contour), they present the same difficulties for wavelet compression as did the original edges.

Figure 3.6 shows the residual image $\widetilde{t}$ created when the coded cartoon sketch $\widehat{c}$ (Figure 3.5) is subtracted from the original image f. A great amount of geometrical structure remains along the edge contours described by the sketch. Compressing this residual image using the zerotree coder, and adding the result back to the coded cartoon sketch, we obtain the image shown in Figure 3.7. For comparison purposes, Figure 3.8 shows the result of applying the standard zerotree compression scheme to the original image, without using a cartoon sketch. By reducing the amount of geometrical information coded with wavelets,

Figure 3.6 Residual texture image $\widetilde{t} = f - \widehat{c}$.

our scheme reduces the visual ringing artifacts along the dominant edges. Because fewer bits were available to code textures, though, the ultimate PSNR was lower than the standard zerotree technique.

As demonstrated by the above experiment, for a natural image coder which uses geometrical coding, the residual information near coded edges presents particular challenges. In [16], Froment demonstrates the effectiveness of coding this type of residual using wavelet packets. We seek, however, an approach which is more closely tailored to the geometry of the contour. In the next section, we present a simple technique which avoids the challenge of compressing these ridge artifacts.

Figure 3.7 Wavelet-compressed residual image $\widehat{t}$ + coded cartoon sketch $\widehat{c}$, total 0.40 bpp, PSNR = 28.48dB.

Figure 3.8 Zerotree compression of Cameraman image, 0.40 bpp, PSNR = 28.98dB.

3.2.2 Wavelets with masking

In our two-stage compression scheme, we assume that the encoder transmits the cartoon sketch before transmitting the residual image. In particular, this means that the encoder has full knowledge of the *locations* of the edge approximations, and hence of the possible ridge artifacts. A simple method for avoiding the difficulty of wavelet-based compression of these artifacts is to "mask" the possible artifacts out of the texture image, setting $\widetilde{t} = 0$ near all coded cartoon edges (as in [16]). This removal of information (which may not be recovered by the decoder) has a significant impact on the PSNR. It also discards legitimate texture information close to edges, which cannot easily be separated from edge artifacts. The resulting residual image, however, may be compressed much more efficiently. In addition, the final coded representations for dominant edges will be given only by the geometric coder. This should result in clean, sharp edges, similar to those in the cartoon sketch.

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

Figure 3.9 shows the residual image after masking. Most ridge artifacts near wedgelet-

Figure 3.9 Residual image $\widetilde{t}$ after masking. Most ridge artifacts have been removed.

coded edges have been removed. Compressing this residual image using the zerotree coder, and adding the result back to the coded cartoon sketch, we obtain the image shown in Figure 3.10. Compared to the technique without masking, or to the standard zerotree technique, this process results in much less ringing around the dominant edges of the image. The residual has been made much easier to compress with wavelets; away from edges, this scheme yields the highest PSNR. Unfortunately, masking destroys a large amount of information in the image that cannot be retrieved. The gain in residual coding efficiency is overcome by this loss of information; the result is a lower overall PSNR.

A possible interpretation of this approach is that it moves most of the texture compression errors into regions very close to edges, in order to compress the remaining features more accurately. Such an approach has a physiological motivation: the human visual sys-

Figure 3.10 Wavelet-compressed masked residual image $\widehat{t}$ + coded cartoon sketch $\widehat{c}$, total 0.40 bpp, PSNR = 27.35dB. Ringing artifacts are significantly reduced, but the PSNR drops as well.

tem has a spatial masking effect which can cause decreased sensitivity to errors very close to edges. Indeed, the visual masking effect has been exploited in previous compression techniques, such as the video coding algorithm described in [17]. We desire, though, to exploit geometric modeling to the extent that it actually improves PSNR, not just perceptual quality.

3.2.3 Additional approaches

While the above approach focused on exploiting geometrical information available to the encoder, it is also possible to exploit information which is available to the decoder. Since the edge cartoon is completely transmitted before the residual is compressed, the decoder also has full knowledge of the locations of the edge approximations. Thus, the decoder has a rough knowledge of the nature and the location of the edge artifacts: they are tall, thin ridges that exist along the edge contours, possibly alternating between positive and negative as the contour crosses the wedgelet cartoon boundary. Such information could potentially be exploited in a residual compression scheme. “Bandelets” [18] offer one approach which adapts its compression scheme to represent residual information near edges, but they have yet to yield a viable compression algorithm. Additional techniques for residual compression are a possible topic of future research.

3.3 Improving the Compression Scheme

The wedgelet-based cartoon compression scheme, combined with the tapered masking scheme for wavelet compression of the residual image, results in a compressed image with fewer ringing artifacts but lower PSNR than a standard wavelet compression of the image. Indeed, the residual image is actually compressed more efficiently, but too much information has been discarded to overcome the drop in PSNR. Ultimately, we wish to exploit geometric modeling to attain improvements in visual quality and PSNR.

While the components of this two-stage cartoon/texture scheme satisfy the practical criteria **G1–G3**, the fundamental flaw is the lack of global rate-distortion optimality (**G4**). The wedgelet decomposition in Stage I has been optimized only locally. The consideration of placing wedgelets is made without knowledge of *any* residual compression scheme to follow. The resulting wedgelet placements often create residual artifacts which are difficult to code. To achieve competitive PSNR performance, wedgelets should be placed only when they actually *improve* the overall rate-distortion performance of the coder. As has been demonstrated, visual performance should also improve anywhere these wedgelets are used.

Achieving global rate-distortion optimality requires sharing information between the geometry-based coder and the residual coder. The two methods may actually be combined into a “single-stage” coder which never explicitly creates a cartoon sketch. Chapter 4 introduces one such scheme which uses wedgelets and wavelets in a framework which is globally rate-distortion efficient.

Chapter 4

W-SFQ: Geometric Modeling with Rate-Distortion Optimization

Chapter 3 illustrated the potential effectiveness of geometric modeling and compression of edge contours. It also demonstrated the need for a natural image coder to wisely apply its geometric techniques in a rate-distortion sense (criterion **G4**).

This chapter introduces a method which uses a simple wedgelet-based geometric representation in a framework which allows for global rate-distortion optimization. In particular, wedgelets are used only when they actually increase the final rate-distortion performance of the coder. They key to our proposed method is the optimization framework of the Space-Frequency Quantization (SFQ) coder [4]; the basics of the SFQ coder are explained in Section 4.1. In Section 4.2, we expand our wedgelet dictionary and propose a method for incorporating wedgelets into SFQ. Section 4.3 describes the details of our wedgelet-modified SFQ (W-SFQ) implementation. Finally, Section 4.4 shows that this simple implementation improves upon standard SFQ both in terms of visual quality and in terms of PSNR.

4.1 The SFQ Compression Framework

SFQ is a powerful wavelet-based compression scheme; it outperforms JPEG-2000 at most bitrates. It also involves a quadtree-structured optimization which accommodates our wedgelet representations. A full description of the SFQ coder is given in [4]. The relevant details are repeated here in order to fully explain W-SFQ in subsequent sections.

4.1.1 SFQ fundamentals

The SFQ coder is based on a zerotree quantization framework. The dyadic quadtree of wavelet coefficients is transmitted in a single pass from the top down, and each directional subband is treated independently.* Each node n_i includes a binary map symbol. A 0 symbol indicates a zerotree: all of the descendants of node n_i are quantized to zero. A 1 symbol indicates that the node's four children are significant: their quantization bins are coded along with an additional map symbol for each. Thus, the quantization scheme for a given wavelet coefficient is actually specified by the map symbol of its parent (or a higher ancestor, in the case of a zerotree); the map symbol transmitted at a given node refers only to the quantization of wavelet coefficients descending from that node. All significant wavelet coefficients are quantized uniformly by a common scalar quantizer; the quantization stepsize q is optimized for the target bitrate.

A tree-pruning operation optimizes the placement of zerotree symbols by weighing the rate and distortion costs of each decision. The pruning starts at the bottom of the tree and proceeds upwards. In the beginning, it is assumed that all coefficients are significant, and decisions must be made regarding whether to group them into zerotrees. The coder uses several bottom-up iterations until the tree-pruning converges; convergence is guaranteed because the number of zerotrees can only increase in each iteration. At the beginning of each iteration, the coder estimates the probability density $p(\widehat{w})$ of the collection of signif-

*Scaling coefficients are coded separately; one approach is mentioned in Section 4.4.

icant coefficients; this yields an estimate of the entropy (and hence coding cost) of each quantized coefficient. Ultimately, adaptive arithmetic coding [15] is used for transmission of these quantization bin indices. The SFQ tree-pruning produces a near-optimal configuration of zerotrees; finding the truly optimal configuration would require computationally demanding techniques such as an exhaustive search.

Before describing the tree-pruning, we introduce some notation. Let w_i be the wavelet coefficient at node n_i, and let $\widehat{w}_i$ denote the coefficient quantized by stepsize q. The set of the four children of node n_i is denoted C_i, and the subtree of descendants of node n_i is denoted U_i (note that this does not include node n_i).

Optimization in the SFQ framework begins with Phase I, where the tree is iteratively pruned based on the rate and distortion costs of quantization. Phase I ignores the bits required to transmit map symbols; in Phase II the tree-pruning is adjusted to account for these costs.

4.1.2 SFQ tree-pruning: Phase I

In each iteration of the Phase I optimization, those nodes currently labeled significant are examined (those already in zerotrees will remain in zerotrees). The coder has two options at each such node: create a zerotree (symbol 0) or maintain the significance (symbol 1). Each option requires a certain number of bits and results in a certain distortion relative to the true wavelet coefficients. The first option, zerotree quantization of the subtree beginning with node n_i, requires $R_i^{(0)} = 0$ bits because no information is transmitted besides the map symbol. This option results in distortion

$$D_i^{(0)} = \sum_{n_j \in U_i} {w_j}^2.$$

The second option is to send a significance symbol for n_i, as well as the quantization bins corresponding to $\widehat{w_j}, n_j \in C_i$. Note that for this option, given the bottom-up framework, we must consider the (previously determined) rate and distortion costs of nodes in C_i as well. Thus

$$R_i^{(1)} = \sum_{n_j \in C_i} - \log_2 \left[p(\widehat{w_j}) \right] + \sum_{n_j \in C_i} R_j.$$

This option results in distortion

$$D_i^{(1)} = \sum_{n_j \in C_i} (w_j - \widehat{w_j})^2 + \sum_{n_j \in C_i} D_j.$$

The decision between the two options is made to minimize the Lagrangian cost $J_i = D_i + \lambda R_i$ where λ is the optimization parameter controlling the tradeoff between rate and distortion.

4.1.3 SFQ tree-pruning: Phase II

In Phase II of the SFQ optimization, the tree-pruning is adjusted to account for the costs of transmitting map symbols. These costs are obtained by considering the method through which the symbols will be transmitted. Specifically, in the top-down transmission of the quadtree, map symbols are predicted based on the variance of local, causal quantized wavelet coefficients. High variances indicate a likelihood of a significant symbol. Low variances indicate a likelihood of a zerotree symbol. Occasionally the rate-distortion performance of a node may be improved by switching its symbol if the gain in map symbol rate exceeds the loss in “Phase I” rate-distortion efficiency. In particular, let $R^{(m)}_{i,\mathrm{map}}$ denote the rate required to transmit map symbol m. Define

$$\Delta R_{i,\mathrm{map}} = (R^{(y)}_{i,\mathrm{map}}) - (R^{(x)}_{i,\mathrm{map}})$$

and

$$\Delta J_{i,\mathrm{data}} = J^{(x)}_i - J^{(y)}_i = (D^{(x)}_i + \lambda R^{(x)}_i) - (D^{(y)}_i + \lambda R^{(y)}_i)$$

where x and y represent the losing and winning options from Phase I, respectively. Thus, $\Delta R_{i,\mathrm{map}}$ denotes the gain in map rate and $\Delta J_{i,\mathrm{data}}$ reflects the loss in rate-distortion efficiency by choosing the losing option from Phase I. In Phase II, the map symbol at node n_i is switched from y to x if $\lambda \Delta R_{i,\mathrm{map}} > \Delta J_{i,\mathrm{data}}$.

4.2 Wedgelets for W-SFQ

Wedgelets provide a convenient geometric tool for improving the SFQ coder. Because the wedgelet dictionary is dyadically organized, each wedgelet can be used to describe a subtree of wavelet coefficients. Thus, transmitting a wedgelet becomes a third option to be considered in the SFQ pruning. Like the zerotree option, many wavelet coefficients can be described by transmitting very little information. In practice, each wedgelet which is sent eliminates the cost of independently quantizing a number of significant coefficients. Moreover, using a wedgelet automatically assures *coherency* among the wavelet coefficients in the subtree, so ringing artifacts should be reduced. The tree-pruning ensures, though, that wedgelets are included only when R/D efficient.

In Section 4.2.1, we adapt the wedgelet dictionary to improve its descriptions, and to ease its integration into our W-SFQ framework. In Section 4.2.2, we explain how to transmit individual wedgelets and describe how these may be translated into subtrees of wavelet coefficients.

4.2.1 Wedgelet parameterization

As presented in Chapter 2, each wedgelet represents a picture of a *sharp* edge passing through a block. The transition of intensities from m_1 to m_2 is abrupt. Natural images, however, also contain *smooth* edges; transitions from dark to light may take up to 10 pixels or more, depending on the scene. We find it useful, then, to extend our dictionary to accommodate a representation for smoothness. The notion of smoothness is orthogonal to the parameters (θ, d, m_1, m_2) already defined; these parameters need not be altered to accommodate smoothness.

A simple model for smoothness assumes a typical shape of the edge profile. One possibility, for example, would be a linear transition of intensities from m_1 to m_2, where smoothness could be fully parameterized by a single variable s, the width of the transition. Based on heuristic analysis, we desire a smooth transition of intensities. Retaining

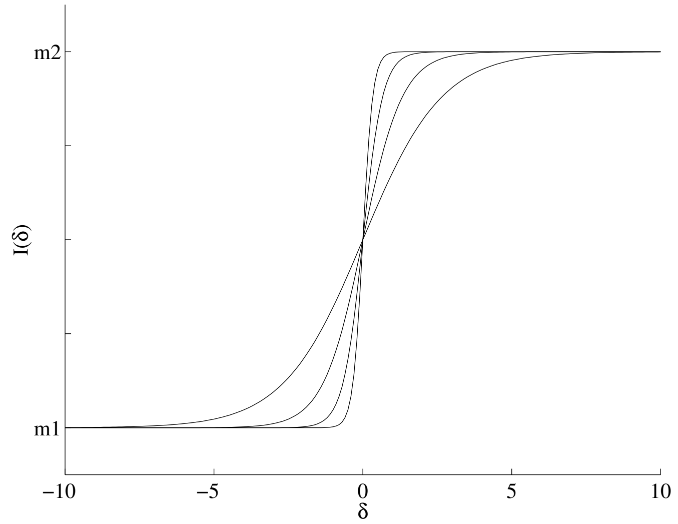


Figure 4.1 Edge profile function $I(\delta(x, y))$ for $s = 2, 4, 8, 16$.

the single-parameter description of smoothness, we choose a sigmoid profile function. Let $\delta(x, y)$ denote the signed normal distance from pixel (x, y) to the sharp edge, with the convention that $\delta < 0$ on the "dark side" of the edge. Let s equal the desired transition width, in pixels. We denote the smoothed intensity at pixel (x, y) as

$$I(x, y) = I(\delta(x, y)) = \frac{m_2 - m_1}{1 + e^{-\frac{6\delta}{s}}} + m_1.$$

When $\delta = s$, the profile function has attained over 99% of its transition. Examples of this profile function are shown in Figure 4.1 for a range of s values.

The addition of an additional wedgelet parameter adds complexity to the search for the best set of parameters $\left(\widehat{\theta}, \widehat{d}, \widehat{m_1}, \widehat{m_2}, \widehat{s}\right)$ to describe a given image block. Because smoothness is unrelated to the other parameters, though, reasonable accuracy may be achieved by first assuming an abrupt transition and estimating $\left(\widehat{\theta}, \widehat{d}, \widehat{m_1}, \widehat{m_2}\right)$. Fixing those parameters, an estimate for $\widehat{s}$ may be obtained with a search over possible s values.

Motivated by our specific application of wedgelets, we make a small adjustment to the parameterization which is transmitted to the decoder. In implementing W-SFQ, we intend to infer wavelet coefficients from transmitted wedgelets. These wavelet coefficients depend only the *relative* pixel intensities, though, so it is not necessary to transmit both of the intensity parameters. Instead, we may transmit only $h = m_2 - m_1$. This approach is used in W-SFQ.

4.2.2 Transmitting wedgelets

When we wish to transmit a wedgelet block, we use a rate-distortion coding framework. This operates under the same R/D parameter λ used in Section 4.1. Each of the three wedgelet parameters (d, θ, h) is quantized separately; the quantization stepsizes are chosen to ensure the correct operating point on the R/D curve. To perform such an optimization, the influence of each parameter's distortion on the squared-error image distortion must be estimated. This estimation is performed at both the encoder and decoder, based on simple assumptions and on previously coded parameters. The height parameter h, for example, is coded first. For large values of h, errors in transmitting d and θ will create significant distortion in the coded wedgelet block; more bits should be used to quantize these parameters. Lacking a clear relation between distortion of the smoothness parameter s and the image distortion, we choose a discrete set of possible s values instead of quantizing a continuous parameter. The range of possible s is adjusted based on the size of the current wedgelet.

Adaptive arithmetic coding is used to transmit the indices of the smoothness parameters. We denote by R_{W_i} the rate required to code all of the wedgelet parameters for node n_i.

Once coded for a node n_i, a wedgelet may be used to predict the wavelet coefficients at all descendants in U_i. This is due to the approximate support of each wavelet basis function within its corresponding dyadic block. Thus, one way to obtain a prediction for these coefficients is to create an image containing the coded wedgelet at the appropriate location, take the wavelet transform, and extract the appropriate coefficients. For each $n_j \in U_i$, we denote the predicted wavelet coefficient as $w_j^{W_i}$. Note that this method may actually be used to predict wavelet coefficients in all three subbands; at this time, however, we treat each subband independently.

The next section describes the steps necessary to add a wedgelet symbol to the SFQ tree-pruning. A similar technique was first presented in [19].

4.3 W-SFQ Tree-Pruning

With an established strategy for coding wedgelet parameters and for predicting the corresponding wavelet coefficients, the addition of wedgelet symbols to the SFQ tree-pruning is straightforward. When a node n_i is transmitted with a wedgelet (symbol 2), the wedgelet parameters are used to compute the values of all coefficients in the subtree U_i. The tree is considered pruned at that point, and no further information is transmitted for any of the node's descendants.

Phase I of the SFQ coder ignored the possible rate cost of transmitting map symbols.

Ultimately, the cost of 0 and 1 symbols would not differ by a great amount, so the pruning could safely neglect the costs (until Phase II). We expect, however, many fewer wedgelet symbols to be transmitted, since each one represents so many possibly significant coefficients. We find it useful, then, to consider in Phase I of W-SFQ a rough estimate of the probability (and hence rate cost) of sending symbol 2. We denote the probability of a wedgelet map symbol as P_2 and choose a suitably low value. It follows that the Phase I rate cost for the wedgelet option is given by

$$R_{i(2)} = -\log_2 [P_2] + R_{W_i}$$

and the resulting distortion is simply

$$D_{i(2)} = \sum_{n_j \in U_i} \left(w_j - w_j^{W_i}\right)^2.$$

These costs are weighed against the costs of the other two symbols; the option with the lowest Lagrangian cost is chosen.

As before, the tree-pruning decisions are made at nodes previously designated as significant (symbol 1); convergence is guaranteed because the number of significant coefficients may only decrease. At the beginning of each iteration, the probability estimate is still made only from those coefficients designated significant. Note an important difference, however: coefficients labelled part of a zerotree may later be included in a larger wedgelet tree, and vice-versa.

Phase II of W-SFQ again adjusts the tree-pruning to account for the costs of sending the map symbols. The predictive scheme for transmitting the symbols is similar to that of SFQ: low variance of neighboring coefficients indicates a likelihood of a zerotree symbol, while high variance indicates a likelihood of a significant *or* wedgelet symbol. Thus, symbols 1 and 2 are grouped together into a "significant cluster", and an extra symbol is transmitted when it is necessary to distinguish between the two. Phase II optimization then proceeds as in SFQ, optimizing the choices between zerotrees and significant clusters.

Just as the SFQ tree-pruning finds a near-optimal configuration of zerotrees, we believe our W-SFQ tree-pruning will find a near-optimal configuration of zerotrees and wedgelets. Ideally, our W-SFQ implementation would be a true superset of the SFQ coder; that is, every decision made with three symbols should do no worse than the corresponding two-symbol decision from SFQ. We believe, however, that the placement of wedgelet symbols, which prunes a number of coefficients with large magnitude from the significance collection, invalidates the guarantee that the W-SFQ will outperform the SFQ coder. The density of significant coefficients will not necessarily become "flatter" with each pruning; this trend was justification in the original SFQ coder for the one-directional pruning of the tree. We refer to the next section for evidence of the efficiency of the W-SFQ strategy.

4.4 Results

We implement the SFQ and W-SFQ algorithms using MATLAB. For each image, we perform a 4-level wavelet decomposition using biorthogonal 10-18 wavelets. The SFQ optimization is designed to compress wavelet coefficients; any efficient technique may be used to separately compress the scaling coefficients. For both SFQ and W-SFQ, we compress the scaling coefficients in a raster scan, predicting each coefficient from its quantized causal neighbors. The prediction errors are quantized and transmitted, with quantization optimized for a Generalized Gaussian Distribution (GGD).

Figure 4.2 shows the SFQ compression of the Cameraman image. At a bitrate of 0.146 bpp, a PSNR of 25.84dB is attained. The tree-pruned segmentation is shown in Figure 4.3. Black coefficients belong to zerotrees while gray coefficients are significant. The tree-pruning chooses to quantize a total of 3020 significant coefficients. As expected, most of the significant coefficients in the high-frequency subbands occur along edges.

At the same bitrate, Figure 4.4 shows the Cameraman image compressed using W-SFQ. A PSNR of 25.94dB was attained, an improvement of 0.10dB over the standard SFQ technique. Figure 4.5 shows the tree-pruned segmentation; white coefficients are described by wedgelets. In this case, a number of previously significant coefficients are now described by wedgelets. The tree-pruning quantizes only 2724 significant coefficients, while transmitting 36 wedgelets at various scales and location. A total of 1908 wavelet coefficients (small and large) are described in these wedgelet subtrees. Ringing artifacts in the wedgelet-pruned regions are noticeably reduced, compared to the SFQ result.

For this example, Table 4.1 shows the bit allocation for SFQ and W-SFQ. Because W-SFQ involves an additional map symbol, we incur a small increase in the coding cost for map symbols. The cost for quantization of significant coefficients is reduced, however,

Figure 4.2 SFQ compression of Cameraman image, 0.146 bpp, PSNR = 25.84dB.

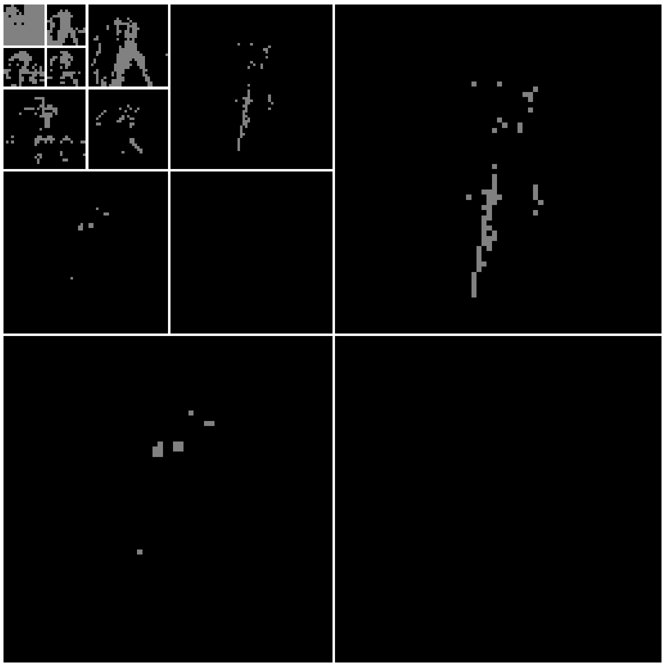


Figure 4.3 SFQ segmentation of Cameraman image. Significant coefficients are gray; zerotree coefficients are black.

Figure 4.4 W-SFQ compression of Cameraman image, 0.146 bpp, PSNR = 25.94dB.

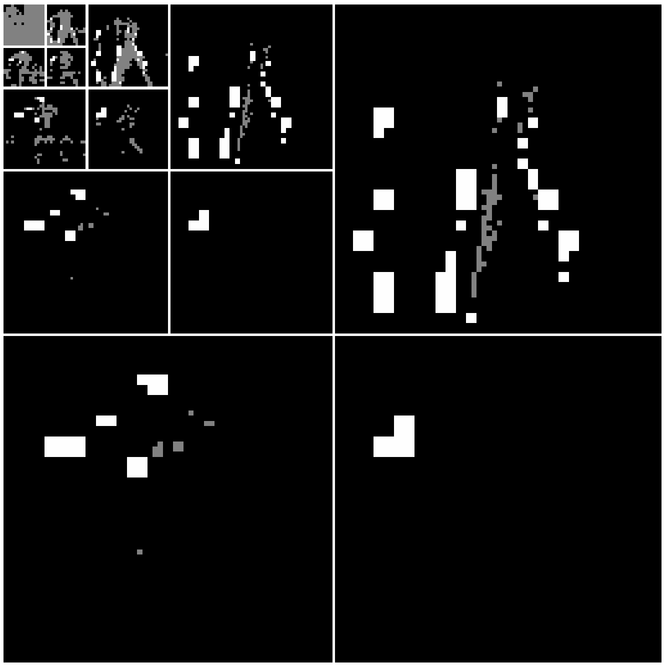


Figure 4.5 W-SFQ segmentation of Cameraman image. Wedgelet subtrees are white; significant coefficients are gray; zerotree coefficients are black.

Table 4.1 Bit allocation for SFQ and W-SFQ operating at 0.146 bpp.

Bit Allocation	SFQ	W-SFQ
Scaling coefficients	13%	13%
Map symbols	13%	14%
Significant coefficients	74%	67%
Wedgelet parameters	–	6%

and the remaining bits are used to transmit the wedgelet information. Figure 4.6 shows an example of a block which is compressed with a wedgelet. In this case, the vertical subtree of wavelet coefficients is compressed 50% more efficiently with a wedgelet than with a significant symbol.

Table 4.2 shows compression results for SFQ and W-SFQ at a variety of bitrates for the Cameraman and Lenna test images. In most cases, W-SFQ offers a higher PSNR than SFQ, with gains up to 0.10dB. Larger improvements tend to occur at lower bitrates. This is due to the fact that, at a given node, the wedgelet option generally introduces a relatively large distortion at a relatively small coding rate. The wedgelet option is most attractive when the rate cost is weighted heavily, corresponding to large values of λ, which yield low bitrates.

W-SFQ also performs better on the Cameraman image than on the Lenna image. The Lenna image involves more complicated textures and curved edges. In the tree-pruning, therefore, fewer wavelet coefficients are described with wedgelets. In one case, the addition of wedgelets actually decreases the PSNR by 0.01dB, a reminder that W-SFQ does not strictly guarantee better performance due to its impact on the histogram of significant coefficients. All of these results were obtained using a very simple implementation of ge-

(a) (b)

Figure 4.6 (a) Cameraman image. (b) 16×16 block representing a subtree in the vertical subbands. In the W-SFQ pruning at 0.146 bpp, a wedgelet is chosen to compress this subtree. At the parent node of the subtree, symbol 1 has a Lagrangian cost of 83600 while symbol 2 has a cost of 41700.

ometric coding. Nonetheless, wedgelets were still selected in some cases as yielding the most efficient description of a subtree. Possible improvements to this approach (using more sophisticated geometric modeling) are discussed in the next chapter. As wedgelet compression is made even more efficient, many more wedgelet descriptions should be selected in the W-SFQ tree-pruning, and the overall performance for both images should only increase.

Table 4.2 W-SFQ compression performance. Gain in dB reflects W-SFQ improvement over standard SFQ. JPEG-2000 figures are included for reference and are approximate, based on interpolation from available (R, D) points. Image sizes: Cameraman 256×256, Lenna 512×512.

Image	Rate (bpp)	**JPEG-2000** PSNR (dB)	**SFQ** PSNR (dB)	**W-SFQ** PSNR (dB)	Gain (dB)
camera	0.077	22.2	23.63	23.66	0.03
camera	0.146	24.8	25.84	25.94	0.10
camera	0.247	27.1	27.91	27.96	0.05
camera	0.381	29.5	29.81	29.85	0.04
camera	0.591	32.1	32.27	32.28	0.01
camera	0.887	35.4	35.21	35.21	0.00
lenna	0.075	28.8	29.14	29.13	(0.01)
lenna	0.139	31.4	31.68	31.69	0.01
lenna	0.341	35.5	35.58	35.58	0.00

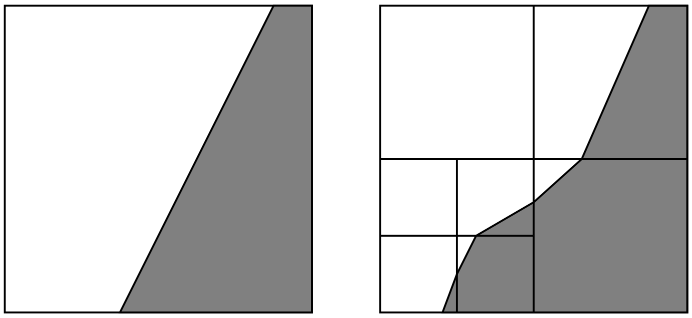
Chapter 5

Extensions

As demonstrated in Chapter 4, the careful use of wedgelets in image compression can result in improved visual quality with an increase in PSNR. In the W-SFQ examples, however, relatively few wedgelets were actually chosen in the tree-pruning (when compared with the number of wedgelets in a cartoon sketch). The key to having more wedgelets chosen (and hence improving the W-SFQ performance) is obvious: wedgelets should be cheaper to code.

In the preliminary implementation of W-SFQ, wedgelets were exploited for their simplicity, but little was done to exploit their geometry to code them efficiently. Essentially, because wedgelets were not omnipresent as in the coded cartoon sketch, they were coded independently. Criterion **G3** was not met in this simple implementation.

This chapter offers possible improvements for W-SFQ by using *groups* of wedgelets to actually trace a contour, making the wedgelets cheaper to code. Other practical issues are also discussed regarding wedgelets and W-SFQ. It is believed that by improving the geometric application and compression of wedgelets, W-SFQ can satisfy **G3**, and its performance can be significantly improved.



(a) (b)

Figure 5.1 Curve trees offer higher precision on a given block. (a) Single coarse wedgelet. (b) Curve tree consisting of smaller wedgelets.

5.1 Advanced Geometric Modeling

By using geometric modeling, nearby wedgelets may be more efficiently compressed jointly than independently. Instead of considering wedgelets one-at-a-time in the W-SFQ pruning, we propose collections of *curve trees*: local cartoon sketches composed of wedgelets describing a contour (see Figure 5.1). Compared to a single coarse wedgelet, a curve tree will more accurately describe a given image block, and geometric modeling may be used to efficiently compress the wedgelets belonging to the curve tree. Chapter 3 provided one possible method for compressing a cartoon sketch; this method is easily applied to a curve tree. We mention briefly, though, some possible improvements to geometric modeling which should make compression of cartoon sketches and curve trees more efficient.

In [11], a *discrete* wedgelet dictionary was introduced which allows for fast computation of wedgelet approximations. For each dyadic block, the number of allowable wedgelets was restricted to some constant M, with parameters d and θ distributed approximately uniformly (see Figure 5.2).

These discrete wedgelets may allow for more efficient top-down compression. Given a

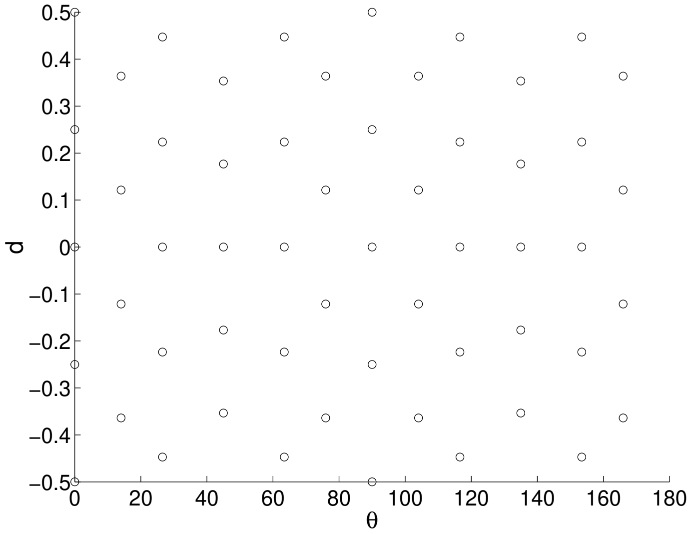
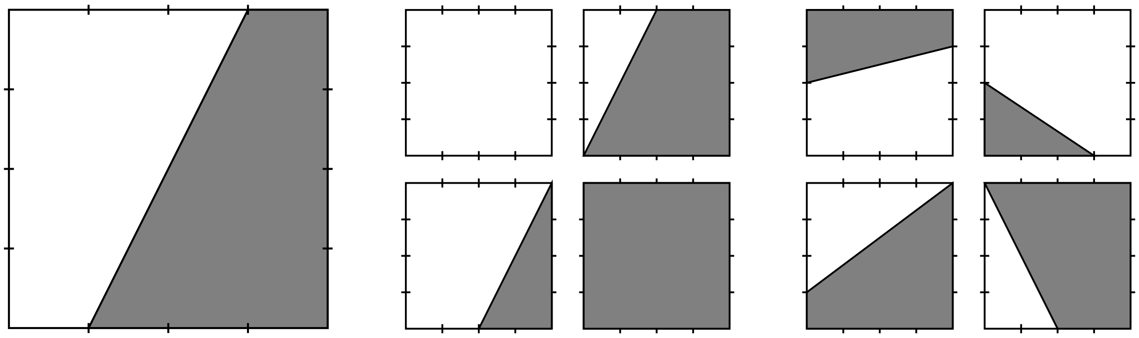


Figure 5.2 Constellation of discrete wedgelet parameters (θ, d) for $M = 56$.

best wedgelet fit for a block, only M^4 possible combinations of wedgelet fits exist for its four children (see Figure 5.3). The transition probabilities from parent to children may be modeled jointly in a $M \times M^4$ matrix, or each transition may be modeled independently in a $M \times M$ matrix. These transition matrices (geometric models) may be optimized for a particular class of curves, or they may be determined experimentally. We propose a simple method to construct an $M \times M$ transition matrix which performs well in practical tests.

As described in Section 3.1, a best wedgelet fit for a block may be predicted from the best wedgelet fit on the block's parent. Let $K \in \{1, \ldots, M\}$ be the index of the parent wedgelet, and let $\widehat{k} \in \{1, \ldots, M\}$ be the predicted index of the child wedgelet. We would like to construct a likelihood function on the set of M possible wedgelets such that the predicted wedgelet corresponds to the most likely.* For each wedgelet $k \in \{1, \ldots, M\}$ we compute $\gamma(k, \widehat{k})$, the l^2 distance from wedgelet k to wedgelet $\widehat{k}$. We create a smooth,

*This is a reasonable assumption when the contour is relatively straight.



(a) (b) (c)

Figure 5.3 Discrete wedgelets have a finite number of possible transitions. (a) Parent wedgelet. (b) A likely refinement. (c) A less likely refinement.

decreasing function $\mu(\gamma)$, and we approximate the likelihood of wedgelet k as $P(k) = \mu(\gamma(k, \widehat{k}))$. In practice, a form such as $\mu(\gamma) = e^{-\alpha\gamma}$ works well for some small constant $\alpha \ll 1$. Thus, wedgelets which are close to the wedgelet predicted by the parent are most likely. After normalization, the values $P(k)$, $k \in \{1, \dots, M\}$ yield row K of the $M \times M$ transition matrix.

Even though these discrete wedgelets are less descriptive (since they lack high resolution on d and θ), they require very few bits to code and may therefore improve rate-distortion performance. Because the transition matrix captures all of the probabilistic modeling required for this wedgelet compression scheme, it allows a convenient framework in which to model geometry. With discrete wedgelets, it is also possible to require *connectivity* among the wedgelets in a curve tree, in order to produce visually pleasing results. Future work should focus on specific smoothness classes of curves and attempt to derive the optimal probability models for the wedgelet transitions.

Finally, the rate-distortion performance of a wedgelet compression scheme can be improved by considering alternative wedgelet fits at each block. Thus far, our techniques at each node have focused on transmitting the wedgelet with the lowest distortion relative to the data. On occasion, another wedgelet might require fewer bits to transmit and have little impact on distortion; the overall rate-distortion cost might be lower for this alternate wedgelet. Of course, this wedgelet impacts the prediction of its children. Tree-pruning with these options becomes more complicated, and it remains a topic of current research.

5.2 Application to Natural Images

The placement of a wedgelet or curve tree in a W-SFQ tree-pruning approximates a block as two constant regions separated by a contour. Many images, however, contain textures close to edges which are not well approximated by constants. Ideally, a scheme such as W-SFQ would imply geometric information only local to the contour, without restricting the rest of the block.

One possibility for accomplishing this goal is to implement a scheme similar to the masking introduced in Section 3.2.2. In this case, a given wedgelet or curve tree is used only to predict those wavelet coefficients with basis functions sufficiently close to the coded contour. Within a wedgelet or curve tree description of a dyadic block, certain subtrees of wavelet coefficients (away from the contour) will remain which are not described by the geometric information. These remaining subtrees may actually be compressed with wavelet-based techniques, with methods already determined by the bottom-up W-SFQ tree-pruning. Figure 5.4 illustrates the process. This system is similar to the previously discussed masking technique. In this case, though, masks are constructed dyadically, and we retain rate-

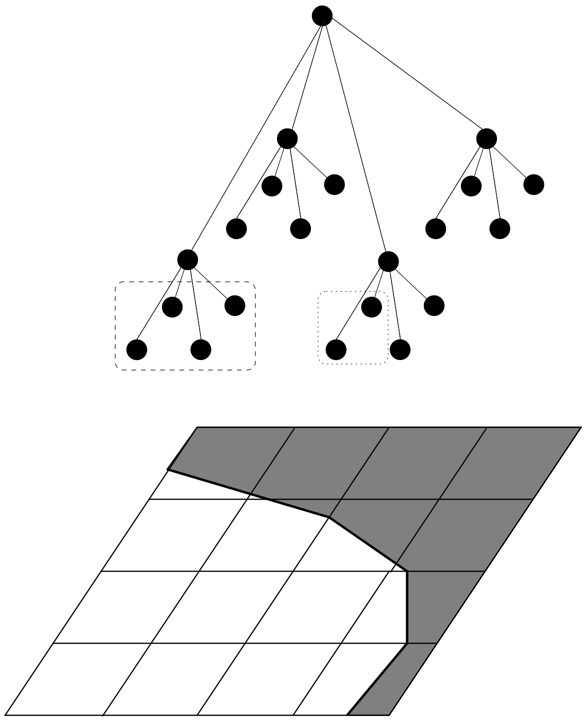


Figure 5.4 Curve tree with its associated wavelet coefficients. Coefficients near the curve (including those enclosed by the dotted line) are implied by its location. Coefficients away from the curve, such as the subtree enclosed by the dashed line, could be described by the SFQ-preferred method (either scalar quantization or zerotree).

distortion optimal compression schemes for the remaining wavelet coefficients. Future work should include an implementation such as the one described above, which should incorporate more wedgelets and curve trees into regions containing textures and edges.

Chapter 6

Conclusions

Natural images contain edges, simple structures that are not efficiently described by wavelets. Geometric modeling captures the inherent simplicity of these edges. The result is a more efficient representation, along with compression schemes which yield visually pleasing approximations at low bitrates. In order to improve rate-distortion compression performance, though, geometric modeling must be applied carefully. Specifically, the use of geometric modeling must leave residual information which may also be efficiently compressed.

We have proposed a scheme which exploits the simple geometric structure of pixels near edges, and which allows wavelets to efficiently represent those regions away from edges. While future work remains to improve many aspects of our algorithm, we have proposed schemes which satisfy the four practical criteria for geometry-based compression. The wedgelet dictionary offers parsimonious descriptions of edges, with clean, sharp approximations at low bitrates, and its performance depends on the regularity of the contour. The CART algorithm provides a technique for optimizing wedgelet approximations of contours. The key to improving visual quality and PSNR, however, rests in the integration of geometric modeling into a comprehensive compression framework. SFQ optimization provides such a framework where wedgelets can be used to improve visual quality and increase PSNR.

References

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

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

3. S. LoPresto, K. Ramchandran, and M. T. Orchard, "Image coding based on mixture modeling of wavelet coefficients and a fast estimation-quantization framework," in *Proceedings, IEEE Data Compression Conference – DCC '97*, Snowbird, Utah, March 1997, pp. 221–230.

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

5. A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Proc. Magazine*, vol. 18, no. 5, pp. 36–58, Sept. 2001.

6. A. B. Lee, K. S. Pedersen, and D. Mumford, "The complex statistics of high-contrast patches in natural images," in *WWW Proc. Second International IEEE Workshop on Statistical and Computational Theories of Vision*, Vancouver, Canada, 2001.

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

8. D. L. Donoho, "Wedgelets: Nearly-minimax estimation of edges," *Annals of Stat.*, vol. 27, pp. 859–897, 1999.

9. M. B. Wakin, "Edge characteristics in wavelet-based image coding," ELEC 599 project report, Rice University, April 2001.

10. J. K. Romberg, H. Choi, and R. G. Baraniuk, "Multiscale edge grammars for complex wavelet transforms," in *IEEE Int. Conf. on Image Proc. – ICIP '01*, Thessaloniki, Greece, Oct. 2001.

11. J. K. Romberg, M. B. Wakin, and R. G. Baraniuk, "Multiscale wedgelet image analysis: fast decompositions and modeling," in *IEEE Int. Conf. on Image Proc. – ICIP '02*, 2002.

12. D. L. Donoho and X. Huo, "Beamlet pyramids: a new form of multiresolution analysis, suited for extracting lines, curves, and objects from very noisy image data," in *Proceedings of SPIE*, July 2000, vol. 4119.

13. M. B. Wakin, J. K. Romberg, H. Choi, and R. G. Baraniuk, “Image compression using an efficient edge cartoon + texture model,” in *Proceedings, IEEE Data Compression Conference – DCC ’02*, Snowbird, Utah, April 2002, pp. 43–52.

14. D. Marr, *Vision*, W. H. Freeman and Company, San Francisco, CA, USA, 1982.

15. I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression,” *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, June 1987.

16. J. Froment, “Image compression through level lines and wavelet packets,” in *Wavelets in Signal and Image Analysis*, A. A. Petrosian and F. G. Meyer, Eds. Kluwer Academic, 2001.

17. B. Girod, “The information theoretical significance of spatial and temporal masking in video signals,” in *Proc. SPIE Human Vision, Visual Processing, and Digital Display*, 1989, vol. 1077, pp. 178–187.

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

19. M. B. Wakin, J. K. Romberg, H. Choi, and R. G. Baraniuk, “Rate-distortion optimized image compression using wedgelets,” in *IEEE Int. Conf. on Image Proc. – ICIP ’02*, 2002.