NOTE TO USERS

Page(s) missing in number only; text follows. Page(s) were microfilmed as received.

ii

This reproduction is the best copy available.

UMI[®]

RICE UNIVERSITY
Multiscale Geometric Image Processing

by

Justin K. Romberg

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

APPROVED, THESIS COMMITTEE:

Richard G. Baraniuk , Chair
Professor of Electrical and Computer
Engineering

Robert D. Nowak
Associate Professor of Electrical and
Computer Engineering

Michael T. Orchard
Professor of Electrical and Computer
Engineering

Steven J. Cox
Professor of Computational and Applied
Mathematics

Ronald A. DeVore
Robert L. Sumwalt Professor of Mathematics
University of South Carolina

Houston, Texas

July, 2003

UMI Number: 3122530

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI Microform 3122530

Copyright 2004 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

Multiscale Geometric Image Processing

by

Justin K. Romberg

Since their introduction a little more than 10 years ago, wavelets have revolutionized image processing. Wavelet based algorithms define the state-of-the-art for applications including image coding (JPEG2000), restoration, and segmentation. Despite their success, wavelets have significant shortcomings in their treatment of edges. Wavelets do not parsimoniously capture even the simplest geometrical structure in images, and wavelet based processing algorithms often produce images with ringing around the edges.

As a first step towards accounting for this structure, we will show how to explicitly capture the geometric regularity of contours in cartoon images using the wedgelet representation and a multiscale geometry model. The wedgelet representation builds up an image out of simple piecewise constant functions with linear discontinuities. We will show how the geometry model, by putting a joint distribution on the orientations of the linear discontinuities, allows us to weigh several factors when choosing the wedgelet representation: the error between the representation and the original image, the parsimony of the representation, and whether the wedgelets in the representation form "natural" geometrical structures. We will analyze a simple wedgelet coder based on these principles, and show that it has optimal asymptotic performance for simple cartoon images.

Next, we turn our attention to piecewise smooth images; images that are smooth away from a smooth contour. Using a representation composed of wavelets and wedgeprints (wedgelets projected into the wavelet domain), we develop a quadtree based prototype coder whose rate-distortion performance is asymptotically near-optimal. We use these ideas to implement a full-scale image coder that outperforms JPEG-2000 both in peak signal to noise ratio (by 1-1.5dB at low bitrates) and visually.

Finally, we shift our focus to building a statistical image model directly in the wavelet domain. For applications other than compression, the approximate shift-invariance and directional selectivity of the slightly redundant complex wavelet transform make it particularly well-suited for modeling singularity structure. Around edges in images, complex wavelet coefficients behave very predictably, exhibiting dependencies that we will exploit using a hidden Markov tree model. We demonstrate the effectiveness of the complex wavelet model with several applications: image denoising, multiscale segmentation, and feature extraction.

Acknowledgments

I have truly enjoyed my time as a graduate student in the Electrical and Computer Engineering department at Rice University. First and foremost, I would like to thank my advisor Richard Baraniuk, whose energy is amazing and enthusiasm infectious. I also benefitted greatly from conversations with other faculty at Rice; in particular, Hyeokho Choi, Robert Nowak, Michael Orchard and Rolf Riedi. Hyeokho and Michael Wakin are collaborators for much of this thesis.

I would also like to thank the two people from outside the department on the thesis committee. Steve Cox taught a seminar on approximation theory at the request of several students from ECE; as thanks he had to serve on all of our committees. Ron DeVore has helped me to generalize the some of the theory in the thesis, and has provided many valuable comments.

The most valuable part of graduate school are the lunchtime and late night discussions/arguments on various topics (some research related, most not) with other students. I have had too many of these to remember them all, but I know that Ramesh Neelamani, Vinay Ribeiro, Felix Fernandes, Rutger van Spaendonck, Michael Wakin, Roger Claypoole, Matt Crouse, Rui Castro, Clay Scott, Yi Wan, Chris Rozelle, Rebecca Willet and Michael Lexa were involved in more than a few. I thank them for the experience.

I would like to give a special thanks to my lovely wife Mercedes, whose companionship and support I would not be the same without.

I would also like to thank the good men and women at Valhalla, for supplying the grease that keeps the wheels turning (as well as for something to do every Friday at 5 in

the afternoon).

For motivation, I would like to acknowledge the anonymous prospective graduate student who, when writing to a professor at the University of Illinois to enquire about research assistant openings, wrote "I heard that graduate school at Illinois is a kind of hell. I want to try it. We have a saying: 'Harsh fire makes strong warrior'." Indeed.

Finally, for reasons that will not be mentioned here, I would like to thank the 1999 Ohio State basketball team for facilitating a story that will likely follow me around the rest of my life.

Contents

Abstract	iii
Acknowledgments	v
List of Figures	x
List of Tables	1
1 Introduction	2
2 Wavelets and Approximation	10
2.1 Wavelets and quadtree models	11
2.2 Wavelets, nonlinear approximation, and asymptotic rate-distortion	13
2.2.1 Quadtree-based wavelet approximation	14
2.2.2 A Prototype Wavelet Coder	16
2.3 Quadtree Selection	17
3 Wedgelet Models and Algorithms	19
3.1 Introduction	19
3.2 Wedgelet Representations	22
3.2.1 Wedgelet Basics	22
3.2.2 Wedgelets and Approximation	24
3.3 Wedgelet Modeling and Inference	25
3.3.1 End point geometry model	29
3.3.2 Connected Wedgelets	31
3.3.3 Wedgelet Inference using the Geometry Model	32

3.4	Wedgelet Coding Asymptotics	35
3.4.1	Contour Localization	35
3.4.2	Quantizing Orientations	37
3.4.3	Parent-child localization	39
3.4.4	Geometry Modeling and Rate-Distortion Performance	40
3.5	Summary	43
4	Approximation and Compression of	
	Piecewise Smooth Images	44
4.1	Representing piecewise smooth images with wavelets and wedgeprints . . .	45
4.1.1	Quadtree based wavelet approximation	47
4.1.2	Wedgeprint approximation	50
4.1.3	From approximation to rate-distortion	54
4.2	Selecting the wavelet-wedgeprint representation	59
4.3	Summary	60
5	A Geometric Wavelet-Based Image Coder	62
5.1	Rate-distortion optimal tree pruning	63
5.2	Practical considerations	66
5.3	WSFQ compression results	67
6	Statistical Models for Complex Wavelets	71
6.1	Introduction	71
6.2	The wavelet transform and shift-invariance	76
6.3	Complex Wavelets	78

	ix
6.3.1 The dual-tree complex wavelet transform	80
6.3.2 The CWT and shift-invariance	81
6.3.3 The CWT and directional selectivity	83
6.4 The Complex Wavelet HMT Model	85
6.4.1 The Wavelet-domain hidden Markov tree	85
6.4.2 Complex wavelet HMT	87
6.5 Applications	90
6.5.1 Image denoising	90
6.5.2 Multiscale Texture Classification	92
6.6 A 1D coherent CHMT model	95
6.7 A Geometrical CHMT	101
6.8 Summary	105
7 Discussion and Conclusions	107
References	110

List of Figures

- 2.1 (a) Wavelet transform of the “cameraman” test image; wavelet coefficients with large magnitude are black, while those with small magnitude are white. (b) Each wavelet coefficient analyzes a local region in the image. The region is broken into four subregions at the next finest scale. This leads to (c) a natural quadtree organization of the wavelet coefficients in each subband. The nodes of the quadtree are indexed using the same parameters (j, k) as the wavelet coefficients: the scale j denotes the level of the quadtree, while the location k denotes the position inside the level. 12
- 3.1 (a) A wedgelet in a dyadic block is parameterized by an angle θ , an offset r , and heights h_1, h_2 . (b) Example of a “simple cartoon image”, (c) The wedgelet representation divides the domain of the image into dyadic squares, using a piecewise constant function in each square to approximate the image. 20
- 3.2 A tickmark dictionary contains wedgelets with line segments that intersect the dyadic square at one of the M equally spaced ticks on each side. Above, we have $M = 4$ for a total of 96 unique line segments (the size of the wedgelet dictionary is 97 with the null wedgelet). 24
- 3.3 Two sets of wedgelet orientations for the same partition. We want a geometrical model that favors “geometrically coherent” joint sets of orientations like those in (a) over incoherent sets like those in (b). 25

- 3.4 Just like (a) grayscale values are refined as we project onto finer scaling spaces, (b) geometry can be refined by using a denser dyadic partition. . . . 26
- 3.5 (a) Child tickmarks (denoted with lines, $M = 4$ here) are collected in pairs to form the parent tickmarks (denoted by ovals). (b) For a parent orientation with end points v_1, v_2 , there are a number of paths which we can use to refine the geometry, one of which is shown in (c). (d) The path corresponds to 64 possible sets of child orientations; we can choose one of 4 orientations in each of the three child quadrants that the supertick path passes through. 30
- 3.6 (a) For superticks bordering the same quadrant, there is only one path. (b) For superticks bordering on adjacent quadrants, there are paths passing through two and four quadrants. (c) For superticks bordering on diagonal quadrants, there are paths passing through three quadrants in both the clockwise and counterclockwise directions. 31
- 3.7 Wedgelet inference on a segment of the “cameraman” image. Using the multiscale geometry model, the orientations on the right take 162 bits to code. Coding them using an independent tree model requires 307 bits. . . 34
- 3.8 The smoothness of the contour β allows it to be locally approximated by a straight line around every dyadic square through which it passes. Specifically, for a dyadic square at scale j , there is a line segment ℓ (with orientation (r, θ)) of length $\sqrt{2} \cdot 2^{-j}$ such that every point on ℓ is within $\omega_j \sim 2^{-2j}$ of β . We can draw a strip of width ω_j and length $\sqrt{2} \cdot 2^{-j}$ around ℓ that locally contains β (β enters and leaves through the ends of the strip). 37

4.1	(a) “Cameraman” image, (b) image segment, (c) horizon class image playing the role of H_β in (4.1).	46
4.2	Since the contour β is smooth inside of dyadic square $S_{j,k}$ with sidelength $C_{\text{supp}}2^{-j}$ it can be bounded by a strip of width $\sim 2^{-2j}$ around a line ℓ . . .	51
4.3	Illustration of a wedgeprint — a wedgelet projected into the wavelet domain	52
4.4	(a) Wavelet pruning of “Cameraman” ($N=7359$, $\text{PSNR}=28.77\text{dB}$), (b) Wavelet-wedgeprint pruning ($N=5158$, $\text{PSNR}=30.27\text{dB}$).	60
5.1	(a) Synthetic test image, (b) “Wet paint” test image, and (c) portion of “peppers” test image used in compression experiments.	68
5.2	Performance improvement of WSFQ (solid) and SFQ (dashed) relative to JPEG-2000 compression for (a) “Peppers” image and (b) “Cameraman” image.	68
5.3	(a) Synthetic image coded using SFQ. (b) Multiscale wavelet-domain segmentation from SFQ tree-pruning. Zerotrees are represented in black; significant coefficients are gray. (c) Image coded using WSFQ. (d) WSFQ segmentation. Wedgeprints are represented in white; note that the three subbands used slightly different wedgeprint configurations.	69
5.4	(a) SFQ coded “wet paint” image, $\text{PSNR} = 29.77\text{dB}$ at 0.0103 bits per pixel. (b) WSFQ coded, $\text{PSNR} = 30.19\text{dB}$ at 0.0102 bits per pixel. Note that the WSFQ coder retains the sharpness of the edges much more effectively than the SFQ coder.	70

- 5.5 Portion of “peppers” test image coded using (a) SFQ (PSNR = 29.08 at 0.07 bpp) and (b) WSFQ (PSNR = 29.25 at 0.07 bpp). A white box indicates a dyadic block described by a geometry tree in one or more subbands. 70
- 6.1 (a) Magnitude of the 1D step response of Daubechies-8 (D8) wavelet coefficients at four consecutive scales. The horizontal axis represents the location of the step edge and the vertical axis represents the magnitude of the corresponding wavelet coefficient. (b) State sequence of the D8 coefficients for the edge location corresponding to the vertical line. (c) Step response of the complex wavelet coefficients with the same locales. (d) State sequence for complex coefficients. Note that the L-S-L-L state sequence for the D8 coefficients is a “broken” chain of large states, while the L-L-L-L state sequence for the complex coefficients is an unbroken chain. 73
- 6.2 Wavelet filter bank structure in 1D. 77
- 6.3 (a) Wavelet transform of the cameraman test image; wavelet coefficients with large magnitude are black, while those with small magnitude are white. (b) Each wavelet coefficient analyzes a local region in the image. The region is broken into four subregions at the next finest scale. This leads to (c) a natural quad tree organization of the wavelet coefficients in each subband. (d) The HMT model associates a hidden state (white circle) with each wavelet coefficient (black circle). The dependencies between the wavelet coefficients are modeled as dependencies between the hidden states. 79

- 6.4 (a) Real (solid) and imaginary (dashed) parts of a complex wavelet basis function. (b) Real parts of the six 2D wavelet basis functions for a given scale and location. 82
- 6.5 An edge contour (a) in the plane reconstructed using wavelet coefficients at one scale j from (b) the real DWT and (c) the CWT. The CWT reconstruction is consistent at all locations and orientations, while the DWT reconstruction varies from location to location. 83
- 6.6 Frequency plane tilings for the real and complex wavelet transforms. Each shade represents the frequency localization for a subband. The three subbands of the real wavelet transform, shown in part (a), captures features along lines at angles of $\{0^\circ, \pm 45^\circ, 90^\circ\}$. The complex wavelet transform, shown in part (b), can discriminate between features at positive and negative frequencies. Hence there are six subbands capturing features along lines at angles of $\{+15^\circ, +45^\circ, +75^\circ, -15^\circ, -45^\circ, -75^\circ\}$ 84
- 6.7 (a) The 1D real HMT associates a hidden state with each wavelet coefficient. (b) In the CHMT, the real and imaginary parts share a hidden state. In 2D, the wavelet coefficients and their associated hidden states are arranged in a quadtree as in Figure 6.3. 88
- 6.8 Scatter plot of the joint distribution of the real and imaginary parts of the complex wavelet coefficients c_i from one subband of the “boats” test image. The marginals for the real and imaginary parts are shown as histograms along the x (real) and y (imaginary) axes. Since the joint distribution is circularly symmetric, we treat the real and complex parts as independent. 88

- 6.9 A 256×256 segment of the 512×512 Boats image denoised using a variety of approaches, with PSNRs. (a) Original image with 256 gray levels. (b) Noisy image ($\sigma_n = 25$). (c) Redundant wavelet hard thresholding (WHT) using D8 wavelet [1, 2]. (d) HMT denoising using D8 DWT [3]. (e) Redundant RHMT denoising using D8 [4]. (f) CHMT denoising. The CHMT denoising algorithm gives a significant improvement both visually and in PSNR. In this particular example, the areas around the wires suffers from fewer ringing artifacts in the CHMT restored image than in the other restored images. 93
- 6.10 Classification results for an image composed of two textures (“grass” and “herringbone”). (a) Texture montage image. Classification of 4×4 pixel regions using an HMT with (b) real wavelets, and (c) complex wavelets. Classification of 2×2 pixel regions using (d) real wavelets and (e) complex wavelets. 96
- 6.11 (a) shows the graphical model for the standard CHMT. The states (represented by the open circles) controls the magnitude of the wavelet coefficients (represented by the filled circles). In (b), the edge location, which governs the phase, is modeled as a separate hidden state represented by a square and is inherited down through scale. Part (c) shows the model used in Section 6.6; the current state depends on the state of the previous wavelet coefficient and the actual phase of the previous wavelet coefficient. 101

- 6.12 *State classification results using the model presented in Section 6.6. The figure on bottom is a multiscale display of the states of the wavelet coefficients. White areas correspond to coefficients in the S state, gray areas to T, and black areas to E. Note that all of the major edges in the slice are resolved at the finest scale with the exception of the pair on the right that are separated by a distance smaller than the width of the finest scale wavelet basis function.* 102
- 6.13 *Magnitude edge response for the six CWT basis functions at a given scale and location as a function of θ for $r = 0$. The phase response varies linearly with r . The relative magnitudes are tied to the angle θ of local geometrical structure while the phase is tied to the offset r .* 102
- 6.14 *Local edge detection using the CWT. Highlights of different regions in the “cameraman” image are shown on the left, on the right are edge estimates from the corresponding complex wavelet coefficients. The upper row shows the original region of the image, the lower row shows the edge estimates.* . . 103
- 6.15 *Contour sketches at two different scales of the “cameraman” image shown in Figure 6.14. The sketches were created directly from the (r, θ) values of the geometrical states estimated by the Viterbi algorithm.* 104
- 6.16 *The geometric hidden Markov tree model. The tree of hidden states, shown on the left, models the multiscale behavior of the contours. Associated with each hidden state is a set of six complex wavelet coefficients (drawn as black dots on the right).* 104

List of Tables

4.1	<i>Wavelet-wedgeprint approximation summary.</i>	54
4.2	<i>Prototype wavelet-wedgeprint coder summary.</i>	58
6.1	<i>Comparison of PSNRs in dB of 256×256 images denoised using the wavelet algorithms of Figure 6.9 plus MATLAB's spatially adaptive Wiener filter (Wiener2) for noise variances $\sigma_{\text{noise}} = 10$ and 25.</i>	92
6.2	<i>The three fundamental modes of CWT coefficients.</i>	98

Chapter 1

Introduction

At the core of image processing lies the problem of modeling *image structure*. Building an accurate, tractable mathematical characterization that distinguishes a “real-world photograph-like” image from an arbitrary set of data is fundamental to any image processing algorithm.

A vital part of this characterization is the *image representation*. Rather than treat an image X directly as a 2D function (continuous spatial domain) or as sequence of pixels (discrete spatial domain), we will recast X using an *atomic decomposition*; a weighted linear combination of atoms f_i from a dictionary $\mathcal{D}$:

$$X(s) = \sum_i \alpha_i f_i(s) + \epsilon(s) \quad \{f_i\} \subset \mathcal{D}, \quad s := (s_1, s_2) \in [0, 1]^2. \quad (1.1)$$

A good representation essentially lowers the dimensionality of the modeling problem — we want $\mathcal{D}$ such that every image of interest can be well approximated ($\epsilon(s)$ small) using relatively few f_i .

In addition to finding suitable representations, we will also be interested in designing *image models*. An image model gives us a systematic way of favoring certain configurations of atoms (choices of $\{\alpha_i\}$ and $\{f_i\} \subset \mathcal{D}$) over others. The model specifies the *relationships* between the atoms in the representation caused by structure in the image.

We are now faced with the two central questions: “What kinds of structure exist in images?” and “What is a representation+model that captures this structure”? While con-

structuring a complete mathematical taxonomy of “interesting” image structure is intractable, we can identify two types of general structure that are prevalent in natural, photograph-like images: Images contain smooth, homogeneous regions and these regions are separated by smooth contours. That is, images exhibit *grayscale regularity* in smooth regions and *geometric regularity* along edge contours.

Over the last decade, the wavelet transform [5] has emerged as the preeminent tool for image modeling. The success of wavelets is due to the fact that they provide a *sparse* representation for smooth signals interrupted by isolated discontinuities [6] (this is a perfect model for “image slices” — 1D cross sections of a 2D image). Simple, tree-based models for wavelet coefficients [3, 7, 8] have defined the state-of-the-art. These models leverage the principle that wavelet coefficients representing smooth regions in the image tend to have small magnitude, while those representing edges have relatively large magnitudes. Arranging the coefficients in a tree (binary tree in 1D, quadtree in 2D), this property allows us to *prune* the wavelet tree over smooth regions, while growing it over edges. For 1D image slices, tree-pruning model based algorithms enjoy not only practical success in denoising and compression, but are also theoretically optimal [9, 10].

For 2D images with edges, wavelets coupled with quadtree models do not enjoy the same degree of success. Although wavelet-based images processing algorithms define the state-of-the-art, they have significant shortcomings in their treatment of edge structure. No matter how smooth a contour is, large wavelet coefficients cluster around the contours, their number increasing at fine scales. The wavelet transform is not sparse for simple images made up of smooth regions separated by smooth contours [11]; it simply takes too many wavelet basis functions to accurately build up an edge. In short, the wavelet transform

makes it easy to model grayscale regularity, but not geometric regularity.

There are two paths we can take: we can either develop more complicated wavelet models, abandoning the simplicity and computational efficiency of tree based models, or we can develop new representations for which simple models will suffice. The second approach has recently become a major focus for the field of Computational Harmonic Analysis. Several new representations have been proposed (see, for example, “curvelets” in [11–13], “bandelelets” in [14], “wedgelets” in [15], and “platelets” in [16]) with nice theoretical sparsity properties, but have yet to have a widespread impact in practical applications.

In this thesis, we will develop a modeling framework for images that are *piecewise smooth* — images that are smooth away from a smooth contour. By combining wavelets and *wedgelets* (discussed below), we are able to take advantage of *both* grayscale and geometrical regularity using conceptually simple and practical tree models. The combined representation has nearly the same theoretically optimal properties for representing (and compressing) piecewise smooth images that wavelets have for piecewise smooth image slices.

In Chapter 3, we isolate the problem of capturing geometrical regularity by modeling *wedgelet decompositions of simple cartoon images* (images which are constant on either side of a smooth contour — see Figure 3.1). These images are purely geometrical in the sense that the only thing of interest in the image is the contour structure.

The wedgelet representation, introduced in [15], is natural for simple cartoon images. A *wedgelet* is a piecewise constant function on a dyadic square S that is discontinuous along a line through S with *orientation* (r, θ) , see Figure 3.1(a) for an illustration. A *wedgelet representation* of an image X consists of a dyadic partition of the domain of X along with a wedgelet function in each dyadic square, see Figure 3.1(c).

The wedgelet representation gives us a vehicle for modeling the geometrical regularity of contours in images. The smoothness of the contours in the cartoon image will be reflected in relationships between the orientations of the wedgelets being used to build up the image. These relationships are succinctly captured using a quadtree based *multiscale geometry model*.

We can characterize contour smoothness using a wedgelet representation coupled with the multiscale geometry model in much the same way as we use wavelets coupled with quadtree models to characterize grayscale smoothness. Both smooth regions and contours in images become “uninteresting” as we zoom in on them: smooth regions become essentially flat and contours become essentially straight. Just as wavelet models are designed to exploit the fact that there is little innovation in grayscale approximations of smooth regions as we move to finer and finer scale, the geometry model allows us to take advantage of the fact that there is also little innovation in geometrical approximations of the contours (see Figure 3.4).

Using the geometry model, we can select a wedgelet representation for an image X (as in (1.1)) by solving a complexity regularized optimization problem that balances *fidelity* (the error in the approximation) against geometrical faithfulness (the likelihood of the representation under the model). The structure of the geometry model allows us to solve this problem with an efficient dynamic program. In the context of coding, the algorithm finds the best representation in a rate-distortion sense. By taking into account the relationships between wedgelet orientations used along smooth contours in cartoon images, image coders using the multiscale geometry model have optimal rate-distortion asymptotics.

In Chapter 4, we turn our attention to a specific set of piecewise smooth images: images that are twice differentiable away from a twice differentiable contour. Neither wavelets nor

wedgelets are effective by themselves on these images; wavelets fail to represent the contour efficiently, while wedgelets cannot be used to build up the smooth regions. We demonstrate that we can efficiently represent both the smooth regions and the contour by combining wavelets and *wedgeprints* – wedgelets that have been projected into the wavelet domain. We derive a simple prototype coder (based on similar pruning principles as wavelet based coders) that achieves near optimal asymptotic rate-distortion performance.

Chapter 5 describes a complete image compression algorithm based on the multiscale geometry model and the wavelet-wedgeprint representation. At its core lies an optimization algorithm that determines the rate-distortion optimal placement of the wavelets and wedgeprints. The coder outperforms JPEG2000 by up to 1.5 dB in peak-signal-to-noise-ratio at low bitrates, and produces images with cleaner edges.

In Chapter 6, we shift focus to building a statistical image model directly in the wavelet domain. For applications other than compression, the approximate shift-invariance and directional selectivity of the slightly redundant *complex wavelet transform* [17–19] make it particularly well-suited to modeling singularity structure. Around edges in images, complex wavelet coefficients behave very predictably, exhibiting dependencies that we will exploit using a hidden Markov tree model [3]. We demonstrate the effectiveness of the complex wavelet model with several applications: image denoising, multiscale segmentation, and feature extraction.

We start in Chapter 2 with a short review (from both the practical and theoretical perspective) of the strengths and shortcomings of the wavelet representation for images. Chapter 3 presents the multiscale geometry modeling framework for wedgelet representations of cartoon images. In Chapter 4, we extend this framework to piecewise smooth

images, demonstrating that we can take advantage of geometrical regularity as well as grayscale regularity using a combined wavelet-wedgelet representation coupled with tree pruning algorithms. This combined representation is used in a full scale image coder in Chapter 5. Chapter 6 presents the hidden Markov tree complex wavelet model. We end with some discussion and conclusions in Chapter 7.

Related Work

An entire research community has recently emerged whose goal is to develop representations and models that are well-suited for images that contains geometrical features. Here, we briefly review some of the prevailing concepts from this inchoate field of study, and relate them to the work presented in this thesis.

The *curvelet* representation [12,20] is a tight frame expansion well-suited for simple images that are smooth away from a smooth contour. Like the combined wavelet-wedgeprint representation we will present in Chapter 4, curvelets have near-optimal approximation asymptotics for such images [11,12]. Unlike the combined wavelet-wedgeprint representation, the curvelet transform decomposes an image in terms of a *fixed* set of atoms. The atoms in the curvelet representation are anisotropic; conceptually, they can be thought of as wavelets “stretched out” along a line. While well-suited for images containing edges, curvelets fail on point singularities; using curvelets in the place of wavelets essentially trades one kind of suboptimality for another (at least in theory). Recent research on curvelets has centered on fast, digital implementations [21] and applications to image denoising [22] and contrast enhancement [23]. As yet, there are no curvelet-based image compression algorithms, an application that will be a focus point for this thesis.

The *contourlet* transform [13, 24] is an image representation in the spirit of curvelets (using anisotropic basis functions) based on directional filterbanks. Again, the representation uses a fixed set of atoms. Recently, theoretical results for contourlet approximation of simple images with edges that mirror the results for curvelets have been published [25]. The contourlet transform has been used for denoising [13], but has yet to be the basis of a full-scale image coder.

Bandelets [14, 26] decompose an image in terms of a warped orthonormal wavelet basis. The warping adaptively stretches out basis functions along contours in the image, while using isotropic functions in the smooth regions. Given the contour structure of the image, bandelets have optimal asymptotic approximation properties for a broad class of simple piecewise smooth images. The complexity of the adaptation to the contours is not accounted for in these results, however. In applications where the geometrical information is known before hand (in the application of human face compression, for example), bandelet-based compression algorithms have been very successful [26]. For general image compression, preliminary results are promising [14], but do not show an improvement over the wavelet-based JPEG2000 standard.

The *platelet* representation [16] is another approach to adaptively decomposing piecewise smooth images. Platelets are essentially modified wedgelets; they are piecewise *linear* (with discontinuity along a straight line) instead of piecewise constant. In [16], platelets are applied to estimation in photon-limited medical imaging.

In Chapter 4, we will show that there are significant advantages to using replacing a cluster of wavelet coefficients around an edge with one atom. Similar in concept are the *wavelet footprints* of [27]. Wavelet footprint algorithms, used primarily for 1D signals,

dynamically replace many coefficients representing a singularity with a single “footprint” atom. Doing so allows a lower-order representation that is still fairly accurate. Wavelet footprints have been applied successfully to denoising in 1D [27]. Applications to 2D image denoising and compression are still being explored [28].

Chapter 5 presents a quadtree-based compression algorithm that combines wavelets and wedgeprints (wedgelets projected into the wavelet domain). A similar scheme is presented in [29]. The tree pruning algorithm is slightly different, and the approximation results have only been published for parametric classes of images (smooth regions that are polynomial separated by piecewise polynomial contours).

The wedgelet inference algorithms of Chapter 3, which are based on a quadtree structured model, can be interpreted as graphical model geometric feature detectors. The *beamlet* algorithms of [30] present applications of other graph traversal algorithms (not model based) for extracting linear ridges from simple images that have been corrupted by noise.

Chapter 2

Wavelets and Approximation

Wavelet-based algorithms have enjoyed success across a broad range of image processing problems. Their practical success stems from their ability to capture most of the image in using relatively few coefficients (energy compaction), and the ease with which we can model image structure in the wavelet domain. Section 2.1 briefly reviews the wavelet transform, and examines how the *quadtrees* organization of the wavelet coefficients allow to characterize and take advantage of structure in images using relatively simple models.

In Section 2.2, we survey results from *nonlinear approximation theory* to help us understand the effectiveness of wavelets from a theoretical standpoint. For signals and images consisting of smooth regions punctuated by isolated singularities, we can construct very accurate, low-order approximations using wavelets. The wavelet representation is asymptotically optimal for such signals; there is no other orthogonal basis in which the approximation error goes to zero faster as the number of terms increases [31]. However, the same is not true for images that contain smooth regions separated by smooth contours; the presence of an edge singularity significantly reduces the approximation power of the wavelet transform. It is this weakness of the wavelet transform that we will work on correcting in Chapters 3, 4, and 5.

The nonlinear approximations we will discuss in Section 2.2 play to strength of the wavelet transform by *adapting* to the image being analyzed. The approximation results show that there *exist* approximations with a certain accuracy. In Section 2.3, we will review an algorithm to *find* good wavelet approximations. The quadtree structure of the wavelet

representation allows us to use a fast dynamic program to find the best approximation (trading off accuracy and complexity) for a given signal or image.

2.1 Wavelets and quadtree models

The wavelet transform decomposes an image $X(s) \in L^2([0, 1]^2)$ using shifts and dilates of mother wavelet functions $\{\psi^0, \psi^{90}, \psi^{\pm 45}\}$ and a scaling function ϕ [5]:

$$X(s) = \sum_{k \in \mathcal{K}(j_0)} u_{j_0, k} \phi_{j_0, k}(s) + \sum_{b \in \mathcal{B}} \sum_{j \geq j_0} \sum_{k \in \mathcal{K}(j)} d_{j, k}^b \psi_{j, k}^b(s) \quad (2.1)$$

where $\phi_{j_0, k}(s) = 2^{j_0} \phi(2^{j_0} s - k)$ and $\psi_{j, k}^b(s) = 2^j \psi(2^j s - k)$. The wavelet coefficients are indexed by a *scale* $j \geq j_0 > 0$, a *location* $k \in \mathcal{K}(j) = \{2^{-(j+1)}, 3 \cdot 2^{-(j+1)}, \dots, (2^{j+1} - 1) \cdot 2^{-(j+1)}\} \times \{2^{-(j+1)}, 3 \cdot 2^{-(j+1)}, \dots, (2^{j+1} - 1) \cdot 2^{-(j+1)}\}$ and a *subband* $b \in \mathcal{B} = \{0, 90, \pm 45\}$ (labelled as such for the directional information they give). We will take the $\{\phi_{j_0, k}, \psi_{j, k}^b\}$ to be an orthogonal basis with $\psi_{j, k}^b$ compactly supported on a square with sidelength $C_{\text{supp}} \cdot 2^{-j}$.

The subbands of the wavelet transform can be arranged on a *quadtree* $\mathcal{T}$, as shown in Figure 2.1. Associated with each node in the quadtree is a local region of the image and a wavelet coefficient for a basis function lying over that region. As we move to finer scales, each *parent* node spawns four *child* nodes; the child regions are nested inside the parent region. The nodes of the quadtree are naturally indexed with the same scale and location parameters (j, k) used for the wavelet decomposition; associated with each node (j, k) is a wavelet coefficient $d_{j, k}^b$ and a basis function $\psi_{j, k}^b$.

*For the sake of brevity, we are leaving out some of the details of wavelet bases on an interval. While the basis functions that “touch” the boundaries cannot be strict dilates and shifts of a mother wavelets, there are constructions with all of the properties (compact support, orthogonality, vanishing moments) that will be needed in subsequent sections of this thesis. See Chapter 7.5 in [6] for details.

Figure 2.1 (a) Wavelet transform of the “cameraman” test image; wavelet coefficients with large magnitude are black, while those with small magnitude are white. (b) Each wavelet coefficient analyzes a local region in the image. The region is broken into four subregions at the next finest scale. This leads to (c) a natural quadtree organization of the wavelet coefficients in each subband. The nodes of the quadtree are indexed using the same parameters (j, k) as the wavelet coefficients: the scale j denotes the level of the quadtree, while the location k denotes the position inside the level.

The wavelet coefficients specify the local grayscale innovation between approximations of the image at different resolutions [6]. In smooth regions of the image, where there is very little grayscale innovation from scale to scale, we expect the wavelet coefficients to have small magnitude. Near edges, the innovation is significant, and the wavelet coefficients will tend to have relatively large magnitude. Put simply, wavelets act as *local edge detectors*. This property is illustrated in Figure 2.1(a); the black, large magnitude wavelet coefficients cluster around edges in the image, while the smooth regions contain white, small magnitude coefficients.

The quadtree organization of the wavelet coefficients also allows us to use simple rules to model the *relationships* between wavelet coefficients at different scales. These rules are based on *persistence* through scale of wavelet magnitudes. Parent coefficients analyzing smooth regions will have children that analyze smooth subregions; we expect small magnitudes to persist through scale. Similarly, large magnitude parents tend to have large magnitude children.

Image coders based on *zerotrees* [7, 8, 32] take explicit advantage of the persistence of small magnitudes. Over smooth regions in the image, we expect entire subtrees of wavelet coefficients to have small magnitude. These subtrees of “insignificant” wavelet coefficients can be safely *pruned* (all of their descendants discarded or quantized to zero), greatly reducing the number of bits needed to code the image while suffering little increase in error. The remaining “significant” coefficients lie on a connected tree, and form an accurate, low-order approximation to the image which is coded. The Space-Frequency Quantizer [8] is an example of a relatively simple but competitive zerotree-based wavelet coder that uses an efficient dynamic program to find the connected tree configuration that gives the optimal rate-distortion balance.

The success of the SFQ and other tree-based wavelet coders speaks to the practical strength of the wavelet+zerotree modeling framework. In the next section, we will discuss some of the theoretical results that explain why coders of this type are so effective, and suggest how they might be improved.

2.2 Wavelets, nonlinear approximation, and asymptotic rate-distortion

To help us understand the strengths and weaknesses of the wavelet+zerotree framework from a theoretical standpoint, we will draw on results from the field of *nonlinear approximation theory*. We gauge the suitability of wavelets for an image class $\mathcal{X}$ by how quickly they can “build up” accurate approximations. Nonlinear approximation is a simplified version of image compression: given an image X from a class $\mathcal{X}$ we form a series of N -term approximates, images that have no more than N non-zero wavelet coefficients, and measure how quickly the approximation error goes to zero as $N \rightarrow \infty$. By restricting the coefficients used

in the approximation to lie on a quadtree, we can translate the nonlinear approximation results into asymptotic rate-distortion results for a simple prototype image coder.

Although we discuss nonlinear approximation only in the context of image compression here, approximating a signal or image using just a few terms is fundamental to many processing applications [31, 33, 34], including denoising [35], restoration [36], and feature detection [30].

2.2.1 Quadtree-based wavelet approximation

Given the wavelet coefficients $\{d_{j,k}^b\}$ of an image X , each connected quadtree $\mathcal{T}$ corresponds to a subset of $\{d_{j,k}^b\}$ and hence to an approximation of X ^{*}:

$$\hat{X}_{\mathcal{T}} = u_{0,0}\phi_{0,0} + \sum_b \sum_{(j,k) \in \mathcal{T}} d_{j,k}^b \psi_{j,k}^b. \quad (2.2)$$

We define the best N -term quadtree estimate $\hat{X}_{\mathcal{T}_N}$ using

$$\mathcal{T}_N := \operatorname{argmin}_{\mathcal{T}: |\mathcal{T}| \leq N} \|X - \hat{X}_{\mathcal{T}}\|_2^2. \quad (2.3)$$

The effectiveness of quadtree-based wavelet approximations of the type (2.2) for a class of images $\mathcal{X}$ is quantified with the *approximation exponent*: the largest p such that for $X \in \mathcal{X}$ [†]

$$\|\epsilon(s)\|_2^2 := \|X - \hat{X}_{\mathcal{T}_N}\|_2^2 \sim N^{-p}. \quad (2.4)$$

^{*}Here we use the same tree $\mathcal{T}$ for each of the subbands of the wavelet transform — since this section deals only with asymptotics, this will not affect the results presented here. However, practical algorithms will in general treat each subband separately.

[†]This section concentrates on asymptotic approximation and rate-distortion bounds. To make the discussion cleaner, we will use the notation $f(k) \sim g(k)$ when there exists a constant C independent of k such that $f(k) \leq Cg(k)$.

The larger the exponent p , the more quickly we are able to “build up” accurate approximations to images in $\mathcal{X}$ using wavelets arranged on a quadtree.

We start by considering image slices — 1D cross sections of a 2D image $X_{\text{slice}}(t) = X(t, s_2)$ for a fixed s_2 . If X_{slice} is uniformly smooth (X_{slice} is bounded and has r bounded derivatives), we can find ^{*} a binary tree based wavelet approximation (the 1D analog to (2.2)) with $\|\epsilon(t)\|_2^2 := \|X_{\text{slice}} - \hat{X}_{\text{slice}}\|_2^2 \sim N^{-2r}$ [9]. Since X_{slice} is uniformly smooth, the tree will be pruned to about the same depth everywhere. Note that the approximation exponent reflects the smoothness of X_{slice} ; the more derivatives it has, the faster the approximation error decays to zero. [†] The exponent $2r$ is optimal for this class; no N -term approximation from a fixed basis will have faster asymptotic error decay.

Amazingly, tree based wavelet approximation still achieves error decay $\|\epsilon(t)\|_2^2 \sim N^{-2r}$ on image slices that have r bounded derivatives except at a finite number of discontinuities (edges) [9]. The approximation naturally adapts to the underlying tree to the image slice; the tree is shallow (pruned at a relatively coarse scale) over the smooth regions and deeper (extended to finer scales) around the edges. Since the edges are perfectly local, the tree does not need to be extended in too many places, and the number of terms being used to represent the edge does not dominate the the number used to represent the smooth regions (see [9] for details).

For images $X(s)$ that are uniformly smooth, we have the analogous 2D result $\|\epsilon(s)\|_2^2 := \|X(s) - \hat{X}_{\mathcal{T}}\|_2^2 \sim N^{-r}$, again the optimal approximation exponent for this class. Similar to 1D, when we add an *isolated* singularity, the exponent does not change.

^{*}In practice, optimal approximations can be found using efficient tree pruning algorithms.

[†]As a technical detail, we also need the wavelets to have more than r vanishing moments — see [6].

Now, we introduce an *edge contour* into an otherwise smooth image and consider the special case $X \in C^2/C^2$:

$$X(s) = X_1(s) \cdot H_\beta(s) + X_2(s) \cdot (1 - H_\beta(s)) \quad (2.5)$$

where $X_1(s), X_2(s) \in C^2([0, 1]^2)$ and $H_\beta(s) \in \text{Horiz}^2$ is a *Horizon class* image *

$$H_\beta(s) := H_\beta(s_1, s_2) = \begin{cases} 1, & s_2 > \beta(s_1) \\ 0, & s_2 \leq \beta(s_1) \end{cases} \quad \beta(s_1) \in C^2([0, 1]). \quad (2.6)$$

For a quadtree wavelet approximation for an image in the class (2.5), we have $\|\epsilon(s)\|_2^2 \sim N^{-1}$ [11] (this result is reviewed in detail in the first part of Chapter 4). The number of wavelets it takes to represent the edge dominates the number of wavelets used in the smooth regions. Whereas introducing isolated singularities did not affect the approximation ability of wavelets, adding an edge contours retards the asymptotic behavior from N^{-2} to N^{-1} .

In Chapter 4, we will show that by augmenting the wavelet representation with *wedgeprints*, we can achieve $\|\epsilon(s)\|_2^2 \sim N^{-2}$ using a similar quadtree-based approximation.

2.2.2 A Prototype Wavelet Coder

The quadtree structure of the approximations in Section 2.2.1 allows us to design a simple coder with *asymptotic rate-distortion* exponent equal to the approximation exponent.

The coder has two stages [†]:

*We can also use the more general *Star* class of [15].

[†]For a complete analysis of image coders of this type, see [9]

Quadtree Approximation : The coder finds a connected quadtree $\mathcal{T}$, as in (2.2), used to construct an approximation to X .

Quantization : The coder quantizes the wavelet coefficients $d_{j,k}^b$ in approximation (2.2) using a known uniform quantizer, creating the quantized approximation $\hat{X}_{\mathcal{T}}^q$. The coder creates a bitstream from the quantized coefficients and a code for the quadtree configuration $\mathcal{T}$.

We quantify the performance of the coder on an image class $\mathcal{X}$ by its asymptotic rate-distortion exponent: the largest p such that given $X \in \mathcal{X}$ and a bit budget R , we can find an approximation $\hat{X}_{\mathcal{T}}^q$ using no more than R bits total with

$$D(R; X) := \|X - \hat{X}_{\mathcal{T}}^q\|_2^2 \sim R^{-p}. \quad (2.7)$$

The link between the asymptotic and rate-distortion exponents is made by realizing that an N -term quadtree approximation can be coded using $\sim N$ bits without affecting the error asymptotics. Given a quadtree approximation $\hat{X}_{\mathcal{T}_N}$, we can quantize $\hat{X}_{\mathcal{T}_N}$ using $\sim N$ bits while suffering quantization error $\|\hat{X}_{\mathcal{T}_N}^q - \hat{X}_{\mathcal{T}_N}\|_2^2 \sim \|X - \hat{X}_{\mathcal{T}_N}\|_2^2$ [9]. Coding the quadtree $\mathcal{T}_N$ using a straightforward, top-down approach requires exactly N bits. As a result,

$$\|X - \hat{X}_{\mathcal{T}}\|_2^2 \sim N^{-p} \Leftrightarrow D(R; X) \sim R^{-p}. \quad (2.8)$$

2.3 Quadtree Selection

We are left with the practical question of how to choose the quadtree approximations underlying the approximations in Section 2.2. For images containing singularities, the

quadtrees used in the approximation (2.2) can be quite uneven, pruned shallow in smooth regions and grown deeper over edges. Of course, a coding (or other processing) algorithm would not know the locations of the contours *a priori*. The encoder needs to make decisions about how to prune the wavelet tree dynamically.

Fortunately, the quadtree structure of the approximation also allows us to formulate and solve an optimization problem to find the best configuration for a given image. Given an image and an N , we wish to find the quadtree $\hat{\mathcal{T}}$ with $|\hat{\mathcal{T}}| \leq N$ that has smallest approximation error $\|X - \hat{X}_{\hat{\mathcal{T}}}\|_2^2$. Putting the problem in Lagrange form, we solve

$$\hat{\mathcal{T}} = \underset{\mathcal{T}}{\operatorname{argmin}} \left(\|X - \hat{X}_{\mathcal{T}}\|_2^2 + \lambda \cdot |\mathcal{T}| \right). \quad (2.9)$$

The parameter λ controls how large a quadtree can be used in the approximation; as $\lambda \rightarrow 0$, we allow $|\mathcal{T}| \rightarrow \infty$.

A solution to (2.9) for any λ can be found efficiently using the classical CART * dynamic programming algorithm [37]. CART makes a single sweep up the quadtree; at each node, the local cost in error of pruning below this node is weighed against the savings in rate. By passing these decisions upwards through ancestor nodes, we can find the wavelet quadtree that globally maximizes (2.9).

The solution $\hat{\mathcal{T}}$ will be optimal in the sense that no other quadtree of size $|\hat{\mathcal{T}}|$ has smaller error. However, CART does not find a solution for a specified N ; as we make λ larger in (2.9), the size of $\hat{\mathcal{T}}$ becomes smaller but “skips over” certain values of N . In practice, though, this is not a major concern.

*“CART” stands for “Classification and Regression Trees” the context in which problems of type (2.9) are common.

Chapter 3

Wedgelet Models and Algorithms

3.1 Introduction

The success that the wavelet representation enjoys is due in large part to the fact that simple tree models can capture the grayscale regularity in smooth images and piecewise smooth 1D image slices. The manner in which this regularity presents itself in the wavelet domain (through entire subtrees of small coefficients) allows us a practical mechanism for exploiting it (fast tree pruning algorithms). The fact that we can use wavelets and quadtree models to build good, low complexity approximations to these image types is a testament to their effectiveness.

In this chapter, we will focus on the class of purely geometrical *simple cartoon images* — images that are constant on either side of a smooth contour. Using *wedgelets* [15] and a *multiscale geometry model*, we will develop a success story for this class of images that parallels wavelets' for uniformly smooth images. In much the same way that we use simple dependencies between wavelet coefficients to capture the grayscale regularity in the image, we will use dependencies between wedgelets to capture geometrical regularity.

The wedgelet representation, introduced in [15], is eminently suited for simple cartoon images. A *wedgelet* is a piecewise constant function on a dyadic square S that is discontinuous along a line through S with *orientation* (r, θ) . see Figure 3.1(a) for an illustration. A *wedgelet representation* of an image X consists of a dyadic partition of the domain of X along with a wedgelet function in each dyadic square, see Figure 3.1(c). The wedgelet representation can be thought of as a *decorated quadtree*; a quadtree dictates how to divide

Figure 3.1 (a) A wedgelet in a dyadic block is parameterized by an angle θ , an offset r , and heights h_1, h_2 . (b) Example of a “simple cartoon image”, (c) The wedgelet representation divides the domain of the image into dyadic squares, using a piecewise constant function in each square to approximate the image.

the domain into dyadic squares, and attached to each leaf of the quadtree is a wedgelet telling us how to approximate the image in that local region.

Of central importance is the *wedgelet inference* problem: Given an image X , we need a way of constructing a low complexity wedgelet representation that closely approximates X . In [15], this issue is addressed by finding a representation $\widehat{\mathcal{W}}$ that optimally balances the error of the approximation $\|X - \widehat{\mathcal{W}}\|_2^2$ against the number of wedgelets used $|\widehat{\mathcal{W}}|$. Such representations are solutions to the optimization problem

$$\widehat{\mathcal{W}} = \underset{\mathcal{W}}{\operatorname{argmin}} \|X - \mathcal{W}\|_2^2 + \lambda \cdot |\mathcal{W}| \quad (3.1)$$

where λ controls the relative importance of the *fidelity* (error) of the approximation versus its *complexity* (number of terms). As $\lambda \rightarrow 0$ in (3.1), we allow more terms in the representation, lowering the approximation error. The solution to (3.1) for any given λ can be found efficiently using a simple tree pruning dynamic program similar to those used in wavelet approximation algorithms (as in Section 2.2).

The wedgelet inference problem can be thought of in more general terms as a *fidelity vs. complexity* optimization problem:

$$\min_{\mathcal{W}} \|X - \mathcal{W}\|_2^2 + \lambda \cdot \text{Complexity}(\mathcal{W}). \quad (3.2)$$

Implicit in the choice of complexity term is a *model*; certain wedgelet configurations are being favored (assigned lower complexity) over others. Balancing the error against the number of terms $|\mathcal{W}|$ as in (3.1) implicitly treats all configurations with the same partition size equally. Interpreted statistically, orientations are *independent* given the partition. As the example in Figure 3.3 shows, treating the orientations independently does not take full advantage of the contour structure in images. Given a partition, there are very few joint sets of wedgelet orientations that are *geometrically coherent*, and these should be favored very highly over geometrically incoherent sets.

In Section 3.3, we develop a more sophisticated modeling framework that characterizes the joint behavior of wedgelet orientations along smooth contours. We introduce a *multiscale geometry model* that assigns a probability $P(\mathcal{W})$ to every possible wedgelet representation. We design $P(\mathcal{W})$ to assign high probabilities (equivalent to low complexity) to wedgelet configurations that are geometrically coherent, and low probability (high complexity) to those that are not. The model can be used as a *prior distribution* on the wedgelet orientations, and can be used as such in statistical image processing applications such as coding, denoising, and detection. Here we will be particularly concerned with coding wedgelet representations; the algorithms developed here are applied as part of a novel image coder in Chapter 5.

Incorporating the multiscale geometry model into the wedgelet inference problem, (3.2) becomes

$$\min_{\mathcal{W}} \|X - \mathcal{W}\|_2^2 + \lambda \cdot [-\log_2 P(\mathcal{W})]. \quad (3.3)$$

The complexity term $-\log_2 P(\mathcal{W})$ is the Shannon codelength [38], and corresponds to the number of bits an ideal coder would take to specify $\mathcal{W}$. In Section 3.3.3, we will show that the structure of the multiscale geometry model allows us to solve (3.3) efficiently using a dynamic program similar to the *Viterbi* algorithm [39].

In Section 3.4, we analyze the asymptotic rate-distortion performance of a wedgelet coder based on a prototype geometry model. By taking into account the relationships between wedgelet orientations used to approximate for $X \in \text{Horiz}^2$, the coder achieves $D(R; X) \sim 1/R^2$, the best possible rate for this class of images.

3.2 Wedgelet Representations

3.2.1 Wedgelet Basics

A *wedgelet* is a piecewise constant function on a square domain S that is discontinuous along a line $\ell_{r,\theta}$ passing through S . Four parameters specify a wedgelet on S : an angle θ and a normalized offset r for the *orientation* of $\ell_{r,\theta}$, and heights h_1, h_2 for the two values the wedgelet takes on either side of $\ell_{r,\theta}$ (see Figure 3.1(a)). The wedgelet function on S with parameters (r, θ, h_1, h_2) is denoted by $w(S; r, \theta, h_1, h_2)$.

A *wedgelet representation* of an image X with domain $[0, 1]^2$ is an approximation of X build out of wedgelets on dyadic squares; squares S with $\text{Side Length}(S) = 2^{-j}$ for a *scale* j . Choosing a wedgelet representation $\mathcal{W}$ requires (1) a choice of dyadic partition $\mathcal{P}$ of the

domain of X and (2) for each dyadic square $S \in \mathcal{P}$, a choice of wedgelet (we will include the “null” wedgelet, $w(S; \cdot) = \text{constant}$ among our choices); $\mathcal{W} = \{\mathcal{P}, w(S; r, \theta, h_1, h_2) \mid \forall S \in \mathcal{P}\}$. As an example, a wedgelet representation of a simple image is shown in Figure 3.1(c).

Since we are concentrating on purely geometrical images, we will only consider wedgelet representations with fixed heights. Every wedgelet in $\mathcal{W}$ is of the form $w(S; r, \theta, A, B)$ or $w(S; r, \theta, B, A)$ where A and B are constants. This reduces the height information to a “coloring” bit specifying which side of $\ell_{r, \theta}$ has height A and which has height B . The wedgelet representation is thus binary valued, allowing us to focus on the geometrical aspect of the approximation.

A wedgelet representation can be organized on a *decorated quadtree*. Each node in the quadtree corresponds to a dyadic square; the squares at the leaves form the partition $\mathcal{P}$. The leaves are also decorated with an orientation and a coloring. We construct the wedgelet approximation by visiting each node on the quadtree, starting at the root and working downwards. At each interior node, we subdivide the corresponding dyadic square into four subsquares (a *parent* node spawns four children). At each leaf, we draw the wedgelet with orientation and coloring specified by the decoration.

In general, the wedgelet orientations will be chosen from a finite *dictionary* $\mathcal{D}_j$, possibly different at each scale j . In Sections 3.3.1 and 3.3.2, we will make use of a *tickmark* dictionary (see also [15, 30, 40]). The tickmark dictionary is constructed by placing M equally spaced ticks along each side of S . Each pair of ticks not on the same side of S are the endpoints for a line segment corresponding to an orientation in the dictionary (see Figure 3.2); there are $6M^2$ orientations total. The tickmark dictionary is natural for enforcing connectivity constraints, as we will see in Section 3.3.2, since the orientations are naturally

Figure 3.2 A tickmark dictionary contains wedgelets with line segments that intersect the dyadic square at one of the M equally spaced ticks on each side. Above, we have $M = 4$ for a total of 96 unique line segments (the size of the wedgelet dictionary is 97 with the null wedgelet).

parameterized by where they intersect the edges of S .

There are two steps in finding a wedgelet representation for an image. The *analysis* step tests each orientation $(r_i, \theta_i) \in \mathcal{D}_j$ in every dyadic square S , measuring the fit of the corresponding wedgelet with the L^2 error $\|X(S) - w(S; r_i, \theta_i, A, B)\|_2$. The *inference* step then uses this information to choose a dyadic partition and a set of wedgelet orientations.

3.2.2 Wedgelets and Approximation

Like wavelets for smooth images (and image slices), we can build accurate, low order approximations to cartoon images using wedgelets. As shown in [15], for an $X \in \text{Horiz}^2$ and a given N , there is a wedgelet representation $\mathcal{W}$ (using continuous valued orientations) composed of no more than N wedgelets ($|\mathcal{W}| \leq N$) such that $\|X - \mathcal{W}\|_2^2 \sim N^{-2}$.

By quantizing the wedgelet orientations (choosing them from a discrete dictionary), we can refine the nonlinear approximation result into an asymptotic rate-distortion result. In [40], the asymptotic rate-distortion of an oracle wedgelet coder (where the encoder has knowledge of the underlying contour in X) was derived as $D(R; X) \sim 1/R^2$; such a coder would operate simply by compressing the 1D contour (using 1D wavelets). To retain an

Figure 3.3 Two sets of wedgelet orientations for the same partition. We want a geometrical model that favors “geometrically coherent” joint sets of orientations like those in (a) over incoherent sets like those in (b).

approximation error of $\sim N^{-2}$, each of the orientations in the nonlinear approximation must be quantized using $\sim \log_2 N$ bits (coding the partition used in the approximation take $\sim N$ bits). Thus for a simple prototype coder that codes each of the orientations independently given the partition (also presented in [40]), we will have $D(R, X) \sim (\log_2 R)^2 / R^2$.

In Section 3.4, we will show that by taking dependencies between the wedgelet orientations into account, a prototype coder achieves the oracle performance $D(R, X) \sim 1/R^2$ for $X \in \text{Horiz}^2$.

3.3 Wedgelet Modeling and Inference

Since edge contours in images are smooth (geometrically regular), wedgelet representations approximating these contours will be structured. In this section, we show how to capture this structure with a multiscale geometry model (MGM).

The MGM is based on a multiresolution approximation framework analogous to the 2D wavelet transform. When we use wavelets to represent an image, the image is projected onto a series of approximation spaces at different scales [6] (see Figure 3.4(a)). As the

Figure 3.4 Just like (a) grayscale values are refined as we project onto finer scaling spaces, (b) geometry can be refined by using a denser dyadic partition.

scales get finer, the projection onto the scaling space moves closer to the actual image. The wavelet coefficients specify the *local differences* between the images projected onto successive scaling spaces; a model on the wavelet coefficients is a model on how the grayscale values are changing as the approximation becomes finer. At fine scales, we expect small wavelet coefficients in regions where the image is smooth.

The contour structure in an image can also be approximated at a series of scales. At each scale j , we partition the domain of the image into dyadic squares with sidelength 2^{-j} . The “scaling space projection” of the contour is a polygonal approximation constructed from line segments inside these dyadic blocks (see Figure 3.4). The MGM models the *difference* between successive polygonal approximations. Due to the geometrical regularity of the contours, we expect these differences to be small.

There are two parts to the MGM. The first is a mapping that takes a polygonal approx-

imation at scale $j + 1$ to a coarser approximation at scale j . This mapping is analogous to the wavelet transform’s scaling filter [6], which specifies how to take the grayscale approximation of the image to next coarsest scale. Given a fine scale polygonal approximation, we can build a *geometry pyramid* $\mathcal{G}$ through a repetitive application of this mapping. Like a wedgelet representation, a geometry pyramid is naturally arranged on a quadtree; the key difference is that *every* node of the geometry pyramid quadtree is decorated with orientation information.

The second part of the MGM is a set of conditional probabilities specifying how we expect the polygonal approximations in $\mathcal{G}$ to refine locally. For an orientation at a parent node, we assign a probability to each possible configuration of child orientations. We will assign high probabilities to child configurations that are “close” to the parent orientation, and low probabilities to those that are very different, thus favoring small geometric innovations over radical changes.

More formally, if $\mathcal{D}_j$ is the wedgelet dictionary at scale j , the two parts of the multiscale geometry model are:

- A mapping $*\mathcal{C}_j : (\mathcal{D}_{j+1})^4 \rightarrow \mathcal{D}_j$ for each scale j that specifies the parent orientation for a certain configuration of child orientations. Given the leaf orientations used in a wedgelet representation $\mathcal{W}$, we iteratively apply $\mathcal{C}$ to obtain the geometry pyramid $\mathcal{G}$. $\mathcal{G}$ is quadtree structured (indexed by corresponding dyadic square S_n) with an orientation at each node. For nodes n at the finest scale J , we set

$$\mathcal{G}(S_n) = (r_n, \theta_n) \tag{3.4}$$

*Here $(\mathcal{D}_{j+1})^4 = \mathcal{D}_{j+1} \times \mathcal{D}_{j+1} \times \mathcal{D}_{j+1} \times \mathcal{D}_{j+1}$.

where (r_n, θ_n) is the orientation of the wedgelet used at node n in $\mathcal{W}$. At coarser scales $j < J$

$$\mathcal{G}(S_n) = \mathcal{C}_j(\mathcal{G}(S_{c_1(n)}), \mathcal{G}(S_{c_2(n)}), \mathcal{G}(S_{c_3(n)}), \mathcal{G}(S_{c_4(n)})) \quad (3.5)$$

where $c_1(n), \dots, c_4(n)$ are the children of node n .

- Conditional probabilities

$$A_{\kappa, \rho; j} = P(\text{Child orientations} = \kappa := \{\kappa_1, \kappa_2, \kappa_3, \kappa_4\} | \text{Parent orientation} = \rho) \quad (3.6)$$

with

$$\sum_{\kappa \in \mathcal{C}_j^{-1}(\rho)} A_{\kappa, \rho; j} = 1 \quad (3.7)$$

where

$$\mathcal{C}_j^{-1}(\rho) = \{\kappa \in (\mathcal{D}_{j+1})^4 \text{ s.t. } \mathcal{C}_j(\kappa) = \rho \in \mathcal{D}_j\}. \quad (3.8)$$

Given $\mathcal{W}$, $\mathcal{G}$ is uniquely defined. Using a multiscale geometry model, we can factor the joint probability of a wedgelet representation as

$$P(\mathcal{W}) := P(\mathcal{G}) = \frac{1}{|\mathcal{D}_0|} \prod_{\substack{j=0, \dots, J-1 \\ n \in \text{Nodes}(j)}} A_{\kappa(n), \mathcal{G}(S_n)} \quad (3.9)$$

where $\kappa(n) = \{\mathcal{G}(S_{c_1(n)}), \dots, \mathcal{G}(S_{c_4(n)})\}$ and $\text{Nodes}(j)$ is the set of nodes in the quadtree at scale j .

A progressive encoding algorithm for $\mathcal{W}$ follows easily from the factorization (3.9). The coding starts at the root, and continues down the tree. The conditional probability at

each node can be calculated from the nodes already encoded, making for a straightforward application of an arithmetic coder [41].

3.3.1 End point geometry model

As an example, we will construct a geometry model that ensures that the endpoints of child orientations 1) roughly coincide with those of the parent, and 2) roughly coincide with each other. The *end point* geometry model uses a tickmark wedgelet dictionary with M ticks per side (see Section 3.2 to choose an orientation for each node on the quadtree). Considering four child blocks together with their parent, as in Figure 3.5(a), we can collect pairs of child tickmarks into *superticks*. The superticks along the perimeter of the 4 children correspond to the tickmark locations for the parent. Given a parent orientation, we will use these superticks to define sets of possible child orientations that “line up” with each other.

For each pair of parent orientation supertick endpoints v_1, v_2 (Figure 3.5(b)), there are a series of paths from v_1 to v_2 through different combinations of interior superticks (Figure 3.5(c)). The possible child orientation configurations for a given parent have their endpoints in the superticks along one of these paths (Figure 3.5(d)).

There are three main cases in determining the supertick paths used to define the possible child orientations. The first, and simplest, is when v_1 and v_2 are located on the perimeter of the same child block (see Figure 3.6(a)). There is only one supertick path is a straight line from v_1 to v_2 , corresponding to four possible child combinations *. For v_1 and v_2 on adjacent quadrants (see Figure 3.6(b)), there are $M/2$ paths going through one intermediate supertick, each corresponding to 16 possible child configurations, and $(M/2)^3$ paths

*For a path passing through k superticks, there are 4^{k-1} possible child orientation configurations

Figure 3.5 (a) Child tickmarks (denoted with lines, $M = 4$ here) are collected in pairs to form the parent tickmarks (denoted by ovals). (b) For a parent orientation with end points v_1, v_2 , there are a number of paths which we can use to refine the geometry, one of which is shown in (c). (d) The path corresponds to 64 possible sets of child orientations; we can choose one of 4 orientations in each of the three child quadrants that the supertick path passes through.

going through three intermediate superticks, each corresponding to 256 child configurations.

For endpoints on child quadrants diagonal from one another (see Figure 3.6(c)), there are $2(M/2)^2$ paths going through two intermediate superticks, with each path corresponding to 64 possible child configurations.

The end point model assigns probabilities to child configurations based on their maximum deviation from the line segment defined by the parent orientation. Letting ℓ_ρ denote the line segment associated with the parent orientation ρ and ℓ_κ denote the union of the four line segments associated with the child orientations $\kappa_1, \dots, \kappa_4$, we set

$$A_{\kappa, \rho; j} \propto e^{-\alpha H(\ell_\rho, \ell_\kappa)} \quad (3.10)$$

where $H(\cdot, \cdot)$ is the Hausdorff distance, and α is a parameter set by the user.

Figure 3.6 (a) For superticks bordering the same quadrant, there is only one path. (b) For superticks bordering on adjacent quadrants, there are paths passing through two and four quadrants. (c) For superticks bordering on diagonal quadrants, there are paths passing through three quadrants in both the clockwise and counterclockwise directions.

3.3.2 Connected Wedgelets

The MGM framework can also be used to find *connected chains* of wedgelets to approximate the image. In this context, the wedgelet inference problem is equivalent to boundary estimation; the chain of wedgelet separates the image into two distinct, well defined regions.

The construction of the connected wedgelet MGM follows closely that of the end point model of Section 3.3.1. The key difference is that instead of collecting pairs of tickmarks into superticks as we move up through scale, we use the finest scale tickmarks to define the orientation dictionary at all scales. We then only consider truly connected paths between end points on the perimeter. For example, in Figure 3.5(d), we see a tickmark dictionary with $M = 4$ become a dictionary with $M = 8$ at the next finest scale. Instead of 64 “almost” connected paths, there are 16 truly connected paths illustrated (4 for each of the 4 pairs of tickmarks on the perimeter of the parent square).

The drawback to enforcing strict connectivity is that the size of the orientation dictionary

quickly becomes unmanageable as we move up through scale. If we use a dictionary with M ticks at the finest scale J , the dictionary will use $2^{(J-j)}M$ ticks at coarser scale j . For practical use, the scale of the problem must be reduced by either not considering all possible connected paths between two endpoints, or by using suboptimal inference algorithms (see [30] for an example of a suboptimal graph traversal algorithm that ensures connectivity).

3.3.3 Wedgelet Inference using the Geometry Model

Given an image X and a value for λ , we want to find the wedgelet representation $\widehat{\mathcal{W}}$ (with fixed heights and associated multiscale geometry representation $\widehat{\mathcal{G}}$) that solves the fidelity v. complexity optimization problem (3.3). Taking advantage of the factorization (3.9), we can find $\widehat{\mathcal{W}}$ using the *Viterbi* (or “max-product”) algorithm [39, 41].

The Viterbi algorithm works upwards through the quadtree, scoring the consequence (δ below) of choosing each of the possible orientations in every dyadic square. Since the child block orientations refine independently once they have been spawned from the parent, we only need to keep track of the *best* set of child orientations for each parent (γ below). At the top the tree, we simply choose the root orientation that has the best overall score, and then backtrack down the quadtree to find the best set descendants to make up $\widehat{\mathcal{G}}$.

We will concentrate on finding the best single-scale wedgelet representation (a “full” quadtree). Unlike the independent problem (3.1), the vast disparity in the complexities of different wedgelet representations (with very few having high probability) makes the single-scale case interesting. It is straightforward to modify the algorithm to find the best pruned representation.

The algorithm can be broken into four steps:

Initialization: For each node n (corresponding to dyadic square S_n) at level J , and for each possible wedgelet orientation in $\mathcal{D}_J$ (indexed by m), the score at each leaf node is just the local error of the wedgelet

$$\delta_n(m) = \|X(S_n) - w(S_n; r_m, \theta_m, \cdot, \cdot)\|_2^2. \quad (3.11)$$

Recursion: For levels $j = J - 1, \dots, 0$, the score for node n at orientation $(r_m, \theta_m) \in \mathcal{D}_j$ is calculated by combining the scores of the child nodes $c_1(n), \dots, c_4(n)$ at scale $j + 1$ and the log-conditional probability. Letting

$$\xi_{m \rightarrow \kappa} = \delta_{c_1(n)}(\kappa_1) + \delta_{c_2(n)}(\kappa_2) + \delta_{c_3(n)}(\kappa_3) + \delta_{c_4(n)}(\kappa_4) + \lambda^2 [-\log_2 A_{\kappa, m}] \quad (3.12)$$

where $\kappa = \{\kappa_1, \kappa_2, \kappa_3, \kappa_4\}$ is a candidate child orientation configuration, the score and best child configuration are calculated with

$$\delta_n(m) = \min_{\kappa \in \mathcal{C}_j^{-1}(m)} \xi_{m \rightarrow \kappa} \quad (3.13)$$

$$\gamma_n(m) = \operatorname{argmin}_{\kappa \in \mathcal{C}_j^{-1}(m)} \xi_{m \rightarrow \kappa}. \quad (3.14)$$

Termination: For the root node at level 0, choose

$$\widehat{\mathcal{G}}(S_0) = \operatorname{argmin}_{(r_m, \theta_m) \in \mathcal{D}_0} \delta_n(m). \quad (3.15)$$

Backtracking: For each node n at levels $j = 0, \dots, J - 1$, let $\{\hat{\kappa}_1, \dots, \hat{\kappa}_4\} = \gamma_n(\widehat{\mathcal{G}}(S_n))$ be the best child orientation configuration for the orientation chosen in S_n . Then, for

Figure 3.7 Wedgelet inference on a segment of the “cameraman” image. Using the multi-scale geometry model, the orientations on the right take 162 bits to code. Coding them using an independent tree model requires 307 bits.

$i = 1, \dots, 4$ set

$$\widehat{\mathcal{G}}(S_{c_i(n)}) = \hat{\kappa}_i. \quad (3.16)$$

Finally, we use the orientations at scale J of $\widehat{\mathcal{G}}$ to construct the wedgelet approximation $\widehat{\mathcal{W}}$.

The Viterbi algorithm is efficient. The computational complexity is linear in number of quad tree nodes, and the number of child configurations in $\mathcal{C}_j^{-1}$ with non-zero probability (as we will see in Section 3.4, this can be bounded by a constant in most relevant cases). For applications where speed is absolutely essential, fast, suboptimal algorithms (see the “list” and “stack” algorithms in [42, 43], for example) can be used with essentially the same effectiveness as Viterbi.

An example of rate-distortion optimal wedgelet inference using the Viterbi algorithm with the endpoint model of Section 3.3.1 is shown in Figure 3.7. Using the MGM, the wedgelet representation takes 162 bits to code, whereas the independent model uses 307 bits.

3.4 Wedgelet Coding Asymptotics

In this section, we will analyze the performance of MGM based wedgelet coders on images $* X \in \text{Horiz}^2$. We will see that under mild conditions on the orientation dictionaries $\mathcal{D}_j$ used at each, a certain class of geometry model leads to asymptotically optimal coding performance.

Our development is broken into four short pieces. Section 3.4.1 introduces the key concept of *trapping* a contour inside of a strip centered on an extended line segment (see Figure 3.8) through a dyadic square. As a consequence of the contour smoothness the dimensions of this strip are $\sim 2^{-j} \times \sim 2^{-2j}$. Thus, we can use orientations at a node in $\mathcal{G}$ to *localize* the contour in the image. Section 3.4.2 shows that by using an orientation dictionary $\mathcal{D}_j$ with proper spacing in the (r, θ) plane, we can still trap the contour inside a strip with similar dimensions centered on a line segment from $\mathcal{D}_j$; we can quantize the orientations and still achieve the same localization properties. In Section 3.4.3, we show that if a parent orientation (r_ρ, θ_ρ) localizes a contour, then children that also localize the contour have orientations close to (r_ρ, θ_ρ) . Section 3.4.4 shows how to exploit these three observations in a simple MGM-based coder that has the optimal asymptotic rate-distortion performance.

3.4.1 Contour Localization

The key feature of images $X \in \text{Horiz}^2$ is that there exists a dyadic scale J_0 such that for all scales $j \geq J_0$ there is a polygonal approximation β_j of β with the following properties

*All of the results in this section apply to images in the slightly more general Star class Star^2 used in [15].

(see [15]):

1. β_j is constructed from line segments with their endpoints on dyadic boxes with side-length 2^{-j} .
2. The end points of the line segments in β_j are “samples” of β ; the line segments and β intersect on the boundaries of the dyadic squares.
3. There is at most one line segment in each dyadic square.
4. The number of line segments N_j used to construct β_j can be bounded as

$$N_j \leq C_n 2^j. \quad (3.17)$$

Since β is smooth, we know that it will never stray very far from β_j . In fact, we can use the line segments in β_j to locally bound β in the plane. Let ℓ be one of the line segments (with orientation (r_ℓ, θ_ℓ)) though a dyadic square S at scale j used to construct β_j . Because $\beta \in C^2$, and we know that ℓ intersects β at two points, we can draw a rectangle (or *strip*) of length L and width $\omega = C_\omega L^2$ centered on point (r_ℓ, θ_ℓ) such that β enters and leaves the rectangle on its ends (the sides perpendicular to ℓ , see Figure 3.8). The line segment ℓ not only locally approximates β in S , but also tells us something about the behavior of β around S ; we can extend ℓ to have length $L \sim 2^{-j}$ and then *trap* β inside a strip of width $\sim L^2 \sim 2^{-2j}$ around the extended line segment. In doing so, we know that every point on the extended line segment is within the strip width of the contour β .

Figure 3.8 The smoothness of the contour β allows it to be locally approximated by a straight line around every dyadic square through which it passes. Specifically, for a dyadic square at scale j , there is a line segment ℓ (with orientation (r, θ)) of length $\sqrt{2} \cdot 2^{-j}$ such that every point on ℓ is within $\omega_j \sim 2^{-2j}$ of β . We can draw a strip of width ω_j and length $\sqrt{2} \cdot 2^{-j}$ around ℓ that locally contains β (β enters and leaves through the ends of the strip).

3.4.2 Quantizing Orientations

The polygonal approximation β_j is constructed out of line segments in dyadic squares at arbitrary, continuous-valued orientations. We will now show that we can get just as good an approximation (to within a constant factor) by using line segments from a finite wedgelet dictionary with size $|\mathcal{D}_j| \sim 2^{-2j}$.

We will use a wedgelet dictionary with orientations uniformly spaced in the r and θ directions

$$\mathcal{D}_j = \left\{ (r_m, \theta_n) = \left(-\frac{\sqrt{2}}{2} + m\Delta_j^r, n\Delta_j^\theta \right) \mid m = 0, 1, \dots, C_r 2^j; n = 0, 1, \dots, C_\theta 2^j \right\} \quad (3.18)$$

where *

$$\Delta_j^r = \frac{\sqrt{2}}{C_r} 2^{-2j} \quad (3.19)$$

$$\Delta_j^\theta = \frac{\pi}{C_\theta} 2^{-j}. \quad (3.20)$$

Consider all of the orientations in a region of the (r, θ) plane centered on $(r_m, \theta_n) \in \mathcal{D}_j$

$$\mathcal{C}_{m,n} = \left\{ (r, \theta); r \in \left[r_m - \frac{\Delta_j^r}{2}, r_m + \frac{\Delta_j^r}{2} \right], \theta \in \left[\theta_n - \frac{\Delta_j^\theta}{2}, \theta_n + \frac{\Delta_j^\theta}{2} \right] \right\}. \quad (3.21)$$

Around each line segment with an orientation in $\mathcal{C}_{m,n}$ we can draw a strip of length * $L_j = 3/2 \cdot 2^{-j}$ and width $\omega_j = 9/4 \cdot C_\omega 2^{-2j}$ that traps β . We want to find a strip of length $\hat{L}_j = \sqrt{2} \cdot 2^{-j}$ and width $\hat{\omega}_j \sim 2^{-2j}$ around a line segment with orientation (r_m, θ_n) that traps β . A sufficient condition on such a strip is that it traps all line segments of length L_j with orientations

$$\hat{\mathcal{C}}_{m,n} = \left\{ (r, \theta); r \in \left[r_m - \frac{(\Delta_j^r + \omega_j)}{2}, r_m + \frac{(\Delta_j^r + \omega_j)}{2} \right], \theta \in \left[\theta_n - \frac{\Delta_j^\theta}{2}, \theta_n + \frac{\Delta_j^\theta}{2} \right] \right\}. \quad (3.22)$$

Using some basic geometry, we see that all of the line segments in $\hat{\mathcal{C}}_{m,n}$ can be trapped

*Line segments in a dyadic square with sidelength 2^{-j} have $\theta \in [0, \pi]$ and $r \in [-\sqrt{2}/2 \cdot 2^{-j}, \sqrt{2}/2 \cdot 2^{-j}]$.

Therefore, the θ are spaced $\sim 2^{-j}$ between 0 and π , while the r are spaced $\sim 2^{-j}$ between $-\frac{\sqrt{2}}{2} 2^{-j}$ and $\frac{\sqrt{2}}{2} 2^{-j}$

*We need the length of the strip to be a little larger than the diagonal length $\sqrt{2} \cdot 2^{-j}$.

inside a strip of width

$$\widehat{\omega}_j = \sqrt{2} \cdot 2^{-j} \tan(\Delta_j^\theta/2) + \frac{\Delta_j^r + \omega_j}{\cos(\Delta_j^\theta/2)} + 2r_c \left(\frac{1}{\cos(\Delta_j^\theta/2)} - 1 \right) \quad (3.23)$$

$$\leq \frac{\sqrt{2}}{90} \cdot 2^{-j} \Delta_j^\theta + \frac{\sqrt{2}-1}{90} \Delta_j^\theta (\Delta_j^r + \omega_j) + (\Delta_j^r + \omega_j) + \frac{2-\sqrt{2}}{90} 2^{-j} \Delta_j^\theta \quad (3.24)$$

where we have safely assumed that $\Delta_j^\theta \leq 35^\circ$. Since $\Delta_j^\theta \sim 2^{-j}$, $\Delta_j^r \sim 2^{-2j}$, and $\omega_j \sim 2^{-2j}$, there is a constant $C_{\widehat{\omega}}$ such that

$$\widehat{\omega}_j \leq C_{\widehat{\omega}} 2^{-2j}. \quad (3.25)$$

As a result of (3.25), we can “quantize” each representative line segment β_j to produce a polygonal approximation $\widehat{\beta}_j$ constructed from line segments with orientations in $\mathcal{D}_j$. Each $\widehat{\ell}$ used in the construction of $\widehat{\beta}$ defines a strip of length $\sqrt{2} \cdot 2^{-j}$ and width $C_{\widehat{\omega}} 2^{-2j}$ which traps β . Thus $\widehat{\beta}_j$ approximates β essentially as well as β_j .

3.4.3 Parent-child localization

From the results in the two previous sections, we know that given a $X \in \text{Horiz}^2$, there is a geometry pyramid $\mathcal{G}$ of depth J with orientations decorating every node that specify line segments used to locally trap β . In this section, we will show that the difference between the orientation used to localize β in a parent block in $\mathcal{G}$ and the orientation used in a child block must be small. Thus, given a parent orientation at scale j , the possible orientations for the children lie in a small subset of $\mathcal{D}_{j+1}$.

Let ℓ_p be the extended line segment with associated strip that contains β around S , and let ℓ_{c_i} be a line segment with associated strip through a child block c_i of S . We know that the strips around ℓ_p and ℓ_{c_i} both have width $\sim 2^{-2j}$, and both contain β . Therefore, there

exists a C_{pc} (not depending on j or ℓ_p) such that all of the possible child line segments (extended to length $\sqrt{2} \cdot 2^{-(j+1)}$ as in Figure 3.8) in c_i are contained in a strip of width

$$\omega_{pc} = C_{pc} 2^{-2j} \quad (3.26)$$

and length $\sqrt{2} \cdot 2^{-(j+1)}$ around ℓ_p . As a result, there are constants C_{pcr} and $C_{pc\theta}$ such that

$$|r_p - r_{c_i}| \leq C_{pcr} 2^{-2j} \quad (3.27)$$

$$|\theta_p - \theta_{c_i}| \leq C_{pc\theta} 2^{-j} \quad (3.28)$$

where (r_p, θ_p) and (r_{c_i}, θ_{c_i}) are the orientations of the parent and child line segments, respectively. As above, the parent and child orientations can be quantized to an element in $\mathcal{D}_j$, and the same result holds (with slightly larger constants).

Given the parent orientation (r_p, θ_p) , equations (3.27) and (3.28) specify a region in the (r, θ) plane in which possible child orientations are located. As we move to finer scales (make j large), the range of possible r values shrinks like $\sim 2^{-2j}$ and the range of possible θ values shrinks like $\sim 2^{-j}$. Recall from (3.18) that the spacing between the r and θ values in the orientation dictionary $\mathcal{D}_j$ decrease at this same rate. Even though the size of the orientation dictionary is growing by a factor of four at every scale, the number of possible children given a parent can be bounded by a constant C_{ch} .

3.4.4 Geometry Modeling and Rate-Distortion Performance

To take advantage of the dependencies between the parent and child orientations in $\mathcal{G}$, we use a MGM with mapping $\mathcal{C}_j$ which ensures that if the child orientations localize the

contour, the parent orientations will as well. Let $\ell_{\kappa_1}, \dots, \ell_{\kappa_4}$ be extended line segments with orientations $\kappa_1, \dots, \kappa_4 \in \mathcal{D}_{j+1}$ for a set of four children at scale $j + 1$ (use $\ell_{\kappa_i} = \emptyset$ for the null orientation). The parent for this child set configuration will be the extended line segment at scale j that is “closest” to the children. Using ℓ_κ to denote the union of the four child line segments, we set

$$C_j(\kappa) = \operatorname{argmin}_{\rho \in \mathcal{D}_j} H(\ell_\rho, \ell_\kappa). \quad (3.29)$$

If all four child orientations are null, the parent orientation will also be null. Given a parent orientation, we will assign equal probability to each set of “geometrically faithful” child orientations; that is, for ℓ_κ falling in the strip defined by (3.26):

$$A_{\kappa, \rho; j} = \frac{1}{T} \quad (3.30)$$

where T is the total number of possible child configurations for the parent orientation; $A_{\kappa, \rho; j} \geq 1/C_{\text{ch}}$. If the parent is the null orientation, all four children orientations will also be null with probability 1.

Given an $X \in \text{Horiz}^2$, and a scale $J > J_0$ with corresponding $\widehat{\beta}_J$, we can use the above geometry model to code $\widehat{\beta}_J$ using $\sim 2^J$ bits. $\widehat{\beta}_J$ uses $N_J \sim 2^J$ non-null orientations at the finest scale (the leaves of the quadtree) to approximate the contour β . We will have to code an orientation for each node on the quadtree between scales J and J_0 that is an ancestor of a leaf being used in $\widehat{\beta}_J$. There are fewer than $4/3N_J$ to code in all. Each orientation requires no more than a constant $\lceil \log_2 C_{\text{ch}} \rceil$ number of bits. For each leaf, we will also need one “coloring” bit telling us on which sides of the line segment the “1” and “0” regions of X is are. Finally, the number of bits to code the “seed” orientations at scale J_0 can be bounded

by a constant (negligible for $J \gg J_0$). Thus, there exists a constant C_{rate} (depending on the curvature bound K) such that we can bound the total number of bits required to code the $\widehat{\beta}_J$ for a given image X as

$$R_J \leq C_{\text{rate}} 2^J. \quad (3.31)$$

We can also use $\widehat{\beta}_J$ to construct a wedgelet approximation of X . In each dyadic square S at scale J containing a line segment ℓ with orientation (r, θ) used in $\widehat{\beta}_J$ we use a wedgelet $w(S; r, \theta, 1, 0)$ or $w(S; r, \theta, 0, 1)$ depending on the coloring. The L^2 error between $X(S)$ and $w(S; r, \theta, \cdot, \cdot)$ is bounded by the area of the strip associated with (r, θ) in S

$$\|X(S) - w(S; r, \theta, \cdot, \cdot)\|_2^2 \leq C_{es} 2^{-3J}. \quad (3.32)$$

In the null regions, we can match the image exactly using a constant (with value 0 or 1 depending on the coloring). There are N_J total wedgelets in the representation $\mathcal{W}_J$, for total error (or distortion)

$$D_J := \|X - \mathcal{W}_J\|_2^2 \leq N_J \cdot C_{es} 2^{-3J} \leq C_{\text{dist}} 2^{-2J}. \quad (3.33)$$

Combining (3.31) and (3.33) we can bound the distortion as a function of rate for any $X \in \text{Horiz}^2$:

$$D(R; X) \leq C_{\text{rd}} \frac{1}{R^2}. \quad (3.34)$$

We have shown that for every $X \in \text{Horiz}^2$ there *exists* a multiscale geometry representation (and associated wedgelet representation) with rate-distortion trade-off (3.34). Using the Viterbi algorithm presented in Section 3.3.3, we can find the *best* wedgelet representa-

tion, which will work at least as well as the one used in the construction above.

3.5 Summary

We now have a modeling framework for wedgelet representations of cartoon images that parallels the wavelet framework for uniformly smooth images. Quadtree based models characterize dependencies between wedgelet orientations along a contours, similar to the way zerotree models characterize the dependencies between wavelet coefficients in smooth regions. Fast selection algorithms find the best wedgelet representation for a given image, similar to the pruning algorithms used in wavelet coders. Finally, we have a prototype coder for cartoon images using the wedgelet representation and a multiscale geometry model that has optimal asymptotic rate-distortion performance.

This parallel structure will allow us to combine the wavelet and wedgelet frameworks, taking the best properties of each, to model *piecewise* smooth images, a problem which we will consider in the next two chapters.

Chapter 4

Approximation and Compression of Piecewise Smooth Images

As discussed in Chapter 2, uniformly smooth image segments can be closely approximated using a small number of wavelet basis functions. The approximation power of wavelets on an image $X \in C^2([0, 1]^2)$ is optimal; as the number of wavelets N being used to form the approximation (1.1) gets large, the error decreases as N^{-2} . Moreover, we can restrict the wavelet basis functions used in the approximation to correspond to nodes on a quadtree [9], allowing using to translate the nonlinear approximation result $\|\epsilon(s)\|_2^2 \sim N^{-2}$ into a rate-distortion result $D(R; X) \sim R^{-2}$.

Chapter 3 establishes a parallel story for wedgelets on cartoon (more formally, Horizon and Star class) images. A simple coder based on the quadtree structured wedgelet representation and multiscale geometry model yields the same (optimal) rate-distortion asymptotics.

In this chapter, we will show that we can achieve the same N^{-2} nonlinear approximation power and near-optimal asymptotic rate-distortion performance $D(R; X) \sim (\log R)^2/R^2$ for image segments $X \in C^2/C^2$ that are smooth away from a smooth contour β (see

Figure 4.1) *:

$$\begin{aligned}
 X(s) &= X_1(s) \cdot H_\beta + X_2(s) \cdot (1 - H_\beta) \\
 X_1(s), X_2(s) &\in C^2([0, 1]^2) \quad H_\beta \in \text{Horiz}^2(K)
 \end{aligned} \tag{4.1}$$

using a dictionary composed of wavelets and *wedgeprints* — wedgelets projected onto fine wavelet scales (a concept similar to the wavelet footprints of [27]). Given an image segment, we can find a suitable representation using a fast dynamic program. These results serve as a theoretical justification the combined wavelet-wedgelet image coder presented in Chapter 5.

In addition to improving the approximation and coding rate, the choice of representation carries geometrical side information. The coder will choose to use wedgeprint along contours in the image, making semantic information about the locations edges available in the compressed domain. These “edge maps” could prove useful for classification or performing rapid database searches in the compressed domain.

In Section 4.1, we present asymptotic approximation and rate-distortion bounds for the wavelet-wedgeprint dictionary. Section 4.2 briefly discusses how to choose a representation given an image.

4.1 Representing piecewise smooth images with wavelets and wedgeprints

In this section, we will show that by using a dictionary composed of orthonormal wavelets and *wedgeprints* (defined below) we can achieve $\|\epsilon(s)\|_2^2 \sim N^{-2}$ for C^2/C^2 image segments.

*The analysis of Section 4.1 holds just as well if we take $H_\beta \in \text{Star}^2$, as in Section 3.4. The key is being able to use a line to localize the contour, as in Figure 3.8.

Figure 4.1 (a) “Cameraman” image, (b) image segment, (c) horizon class image playing the role of H_β in (4.1).

* The atoms used in the approximation are organized on a quadtree. This structure of the wavelet-wedgeprint dictionary allows us to efficiently encode the approximation, making it straightforward to translate the approximation result into a rate-distortion bound.

We will start by reviewing quadtree structured wavelet approximation (see [9, 10] for a thorough treatment of the subject) of piecewise smooth images. Over smooth regions in the image, the wavelet quadtree can be pruned back to shallow depths. Over the contour, the quadtree must be deeper, causing a bottleneck in approximation performance. However, since the contour is smooth, local regions of the wavelet approximation are highly structured. As a result, entire subtrees of wavelets used in the approximation can be replaced with a single wedgeprint without a significant increase in error.

The approximation begins with a full wavelet quadtree with maximum scale J . As shown below, this gives us $\|\epsilon(s)\|_2^2 \sim 2^{-J}$ for $X \in C^2/C^2$ using $\sim 2^{2J}$ terms. Pruning the quadtree back to scale $J/4$ in the smooth regions gives us the same asymptotic error $\|\epsilon(s)\|_2^2 \sim 2^{-J}$ using only $\sim 2^J$ terms. Using wedgeprints, we will be able to prune the contour regions back

*As in Chapter 2, we will concentrate on asymptotic approximation and rate-distortion bounds. To make the exposition cleaner, we will forgo the explicit calculation of many of the constants involved.

to scale $J/2$ while keeping $\|\epsilon(s)\|_2^2 \sim 2^{-J}$. The result is an approximation with accuracy $\sim 2^{-J}$ using $\sim 2^{-J/2}$ terms, and we have $\|\epsilon(s)\|_2^2 \sim N^{-2}$.

4.1.1 Quadtree based wavelet approximation

Let $\{\phi_{0,0}, \psi_{j,k}^b\}_{B,j,k}$ be an orthogonal basis of compactly supported wavelets ($\phi_{0,0}$ is the scaling function). Assume that $\psi_{j,k}^b$ has at least two vanishing moments. Let $d_{j,k}^b$ be the wavelet coefficients for an image $X \in C^2/C^2$

$$X(s) = u_{0,0}\phi_{0,0} + \sum_b \sum_j \sum_k d_{j,k}^b \psi_{j,k}^b(s). \quad (4.2)$$

As discussed in Chapter 2, the $d_{j,k}^b$ and corresponding $\psi_{j,k}^b$ can be arranged on a *quadtree* $\mathcal{T}$. The nodes of $\mathcal{T}$ are naturally indexed with the same scale and location parameters (j, k) used for the wavelet decomposition; associated with each node (j, k) in $\mathcal{T}$ are three * wavelet coefficients $\{d_{j,k}^b\}_B$ and basis functions $\{\psi_{j,k}^b\}_B$.

Below we will construct approximations of X by *pruning* $\mathcal{T}$. Pruning $\mathcal{T}$ at node (j', k') creates a new quadtree $\hat{\mathcal{T}} \subset \mathcal{T}$ with the descendants of node (j', k') (but not the node itself) removed. The approximation $\hat{X}$ is formed from $\hat{\mathcal{T}}$ by using only the $(j, k) \in \hat{\mathcal{T}}$ in the reconstruction formula (4.2).

The $\psi_{j,k}^b$ are compactly supported inside a square with sidelength $C_{\text{supp}}2^{-j}$. We will assign a label of “type S” or “type E” to each node on the quadtree depending on whether or not the contour β passes through the support of $\psi_{j,k}^b$; type S nodes correspond to smooth regions while type E nodes correspond to contour regions. Due to the multiresolution

*For this exposition, we will combine all three wavelet subbands into one quadtree. In practice, the procedure described could be performed on each subband quadtree independently.

properties of wavelets, if (j', k') is type S, all of its descendants are also type S (the support of a parent basis function includes the support of its children). If (j', k') is type E, all of its ancestors are also type E.

We will denote the number of type S coefficients at scale j by $N_S(j)$ and the number of type E coefficients $N_E(j)$. Because β is smooth, we have

$$N_S(j) \sim 2^{2j} \quad (4.3)$$

$$N_E(j) \sim 2^j \quad (4.4)$$

The image is uniformly C^2 on the support of a type S $\psi_{j,k}^b$; as a result, the coefficient magnitudes decay quickly across scale [6]. For type E coefficients, the decay is much slower:

$$|d_{j,k}^b|^2 \sim 2^{-6j} \quad \forall d_{j,k}^b \text{ type S} \quad (4.5)$$

$$|d_{j,k}^b|^2 \sim 2^{-2j} \quad \forall d_{j,k}^b \text{ type E.} \quad (4.6)$$

Let $\mathcal{T}_{\downarrow(j',k')}$ denote the descendants of node (j', k') in $\mathcal{T}$. The cost of pruning the wavelet quadtree at (j', k') is

$$\text{Prune Error} = \sum_b \sum_{\substack{(j,k) \in \\ \mathcal{T} \setminus \mathcal{T}_{\downarrow(j',k')}}} |d_{j,k}^b|^2 \quad (4.7)$$

where $\mathcal{T} \setminus \mathcal{T}_{\downarrow(j',k')}$ is the pruned quadtree. The pruning error is different for (j', k') of different

type:

$$\begin{aligned} \text{Prune Errors}_S &\sim \sum_{j=j'}^{\infty} 2^{2(j-j')} 2^{-6j} \\ &\sim 2^{-6j'} \quad (j', k') \text{ type S} \end{aligned} \quad (4.8)$$

$$\begin{aligned} \text{Prune Error}_E &\sim \sum_{j=j'}^{\infty} N_E(j-j') 2^{-2j} + \sum_{j=j'}^{\infty} N_S(j-j') 2^{-6j} \\ &\sim 2^{-2j'} \quad (j', k') \text{ type E.} \end{aligned} \quad (4.9)$$

Starting with a full quadtree (all scales and all locations — the approximation error is zero), we first prune it back to scale J . From the (4.5)–(4.6) and (4.8)–(4.9) above, this truncation incurs an error

$$\text{Error}_{\text{trunc}} \sim 2^{-J} \quad (4.10)$$

and leaves us with

$$N_{\text{trunc}} \sim 2^{2J} \quad (4.11)$$

terms in the approximation.

Next, we prune the quadtree below every node of type S between scales $J/4, \dots, J/2$. The total number of prunings is $\sim 2^{J/2}$, with each one incurring an error $\sim 2^{-3J/2}$ for a total error of

$$\text{Error}_{\text{pruneS}} \sim 2^{-J}. \quad (4.12)$$

using the reduced number of terms

$$N_{\text{pruneS}} \sim 2^J. \quad (4.13)$$

We are left with an approximation of accuracy $\sim 2^{-J}$ using $\sim 2^J$ terms, meaning $\|\epsilon(s)\|_2^2 \sim N^{-1}$. Reducing the number of terms would require pruning coefficients of type E, which would in turn increase the error significantly. For example, pruning the type E nodes back to scale $J/2$ would reduce the number of terms to $\sim 2^{J/2}$, but increase the error to $\sim 2^{-J/2}$. There is no way remaining to manipulate the wavelet quadtree to get the optimal approximation rate $\|\epsilon(s)\|_2^2 \sim N^{-2}$.

4.1.2 Wedgeprint approximation

To resolve the wavelet approximation bottleneck, we will use $\sim 2^{J/2}$ *wedgeprint* functions to construct the approximation of X from scales $J/2, \dots, J$ instead of $\sim 2^J$ wavelets. To develop the idea, we return briefly to the spatial domain to re-examine how wedgelets can be used to locally approximate X along the contour.

Let $S_{j,k}$ be the square support region for a type E $\psi_{j,k}^b$; $S_{j,k}$ has sidelength $C_{\text{supp}} 2^{-j}$. We will show that there exists a wedgelet $* w(S_{j,k}; r, \theta, h_1, h_2)$ on $S_{j,k}$ such that

$$\|X(S_{j,k}) - w(S_{j,k}; r, \theta, h_1, h_2)\|_2^2 \sim 2^{-3j}. \quad (4.14)$$

Since β is twice continuously differentiable, there exists a line segment ℓ with orientation (r, θ) through $S_{j,k}$ such that β is contained in a strip of width $\sim 2^{-2j}$ around ℓ (recall Section 3.4 and Figure 3.8). On either side of this strip (we will call the two sides “region 1” and “region 2, see Figure 4.2), $X(S_{j,k})$ can be closely approximated by a constant. Since X has bounded first derivative, there are constants C_1 and C_2 such that $X(S_{j,k})$ varies by

*In Chapter 3 we defined wedgelets as functions on the dyadic squares. The definition easily extends to general square domains.

Figure 4.2 Since the contour β is smooth inside of dyadic square $S_{j,k}$ with sidelength $C_{\text{supp}} 2^{-j}$ it can be bounded by a strip of width $\sim 2^{-2j}$ around a line ℓ .

no more than $C_1 2^{-j}$ and $C_2 2^{-j}$ in regions 1 and 2 respectively. That is, we can find h_1, h_2 such that

$$|X(\text{region 1}) - h_1| \leq C_1 2^{-j} \quad (4.15)$$

$$|X(\text{region 2}) - h_2| \leq C_2 2^{-j}. \quad (4.16)$$

Using a wedgelet $w(S_{j,k}; r, \theta, h_1, h_2)$ to approximate $X(S_{j,k})$ gives us an error of $\sim 2^{-4j}$ in regions 1 and 2, and an error of $\sim 2^{-3j}$ (the area of the strip) inside the strip. Thus (4.14) holds.

We can translate (4.14) into the wavelet domain; instead of approximating $X(S_{j,k})$ using a wedgelet, we will approximate $X(S_{j,k})$ projected onto a wavelet subspace using a wedgeprint.

Let $W_{J/2,k}^J$ be the subspace of $L^2([0, 1]^2)$ spanned by the wavelet basis functions at scales $J/2, \dots, J$ in the subtree rooted at node $(J/2, k)$

$$W_{J/2,k}^J = \text{span} \left\{ \psi_{j,k}^b, (j, k) \in \mathcal{T}_{J/2,k} \right\}. \quad (4.17)$$

Figure 4.3 Illustration of a wedgeprint — a wedgelet projected into the wavelet domain

The projection of X onto $W_{J/2,k}^J$ is given by

$$\begin{aligned} \text{Proj} \left(X \rightarrow W_{J/2,k}^J \right) &= \text{Proj} \left(X(S_{\psi_{J/2,k}^b}) \rightarrow W_{J/2,k}^J \right) \\ &= \sum_b \sum_{(j,k) \in \mathcal{T}_{J/2,k}} d_{j,k}^b \psi_{j,k}^b. \end{aligned} \quad (4.18)$$

We define the wedgeprint $\varphi_{J/2,k}(s; r, \theta)$ as the normalized projection of a $(0, 1)$ wedgelet onto $W_{J/2,k}^J$ (an illustration of a wedgeprint is shown in Figure 4.3)

$$\varphi_{J/2,k}(s; r, \theta)' = \text{Proj} \left(w(S_{\psi_{J/2,k}^b}; r, \theta, 0, 1) \rightarrow W_{J/2,k}^J \right) \quad (4.19)$$

$$\varphi_{J/2,k}(s; r, \theta) = \frac{\varphi_{J/2,k}(s; r, \theta)'}{\|\varphi_{J/2,k}(s; r, \theta)'\|_2}. \quad (4.20)$$

By the Bessel inequality, we have

$$\begin{aligned} &\left\| \langle X, \varphi_{J/2,k} \rangle \varphi_{J/2,k} - \text{Proj} \left(X \rightarrow W_{J/2,k}^J \right) \right\|_2^2 \\ &\leq \left\| X(S_{\psi_{J/2,k}^b}) - w(S_{\psi_{J/2,k}^b}; r, \theta, h_1, h_2) \right\|_2^2. \end{aligned} \quad (4.21)$$

As a result, we can replace the $\sim J/2$ wavelet basis functions in subtree $\mathcal{T}_{J/2,k}$ with one

wedgeprint $\varphi_{J/2,k}$ and incur an error of only $\sim 2^{-3J/2}$.

Returning to our pruned wavelet quadtree from equations (4.12)–(4.13), we will replace the subtree of wavelets beneath each type E node at scale $J/2$ with a wedgeprint. There are

$$N_\varphi = N_E(J/2) \sim 2^{J/2} \quad (4.22)$$

such subtrees, making the combined wavelet-wedgeprint approximation within

$$\text{Error}_\varphi \sim 2^{-J} \quad (4.23)$$

of the pruned wavelet approximation.

To summarize (see Table 4.1), the combined wavelet-wedgeprint approximation has three different stages. From scales $0, \dots, J/4$, we use all of the wavelet coefficients, a total of $\sim 2^{J/2}$. From scales $J/4, \dots, J/2$, we use all of the type E wavelet coefficients and their children, a total of $\sim 2^{J/2}$. At scale $J/2$, we use $\sim 2^{J/2}$ wedgeprints to approximate the type E wavelets between scales $J/2$ and J . Collecting the results, the wavelet-wedgeprint representation uses

$$N_{\text{total}} \sim 2^{J/2} \quad (4.24)$$

terms to construct an approximation with error

$$\text{Error}_{\text{total}} \sim 2^{-J}. \quad (4.25)$$

Table 4.1 Wavelet-wedgeprint approximation summary.

Scales	Action	# coeffs	error
$0, \dots, J/4$	keep all	$\sim 2^{J/2}$	0
$J/4 + 1, \dots, J/2$	prune type S	$\sim 2^{J/2}$	$\sim 2^{-J}$
$J/2, \dots, J$	use wedgeprint	$\sim 2^{J/2}$	$\sim 2^{-J}$
$J + 1, \dots$	truncate	0	$\sim 2^{-J}$

Combining (4.24) and (4.25), we have the asymptotic approximation result

$$\|\epsilon(s)\|_2^2 \sim N^{-2} \quad (4.26)$$

We have established the following result:

Theorem 1 *Let $X \in C^2/C^2$. Given a maximum scale J , there exists a quadtree based wavelet-wedgelet approximation $X_{\hat{T}(J)}$ with error less than $C_E \cdot 2^{-J}$ using fewer than $C_N \cdot 2^{J/2}$ terms, where C_N and C_E are constants independent of J . As a result, we can bound the approximation error as a function of the number of terms N in the representation $\|\epsilon(s)\|_2^2 \leq \text{Const} \cdot N^{-2}$.*

4.1.3 From approximation to rate-distortion

In the context of image compression, we are interested more in asymptotic *rate-distortion* performance than approximation decay. An image coder will need to spend bits not only on the expansion coefficients, but also must encode *which* atoms were chosen in the approximation (the addressing information).

Like wavelet based coding, we will have to encode the quadtree underlying the approximation, which can be done simply and efficiently, in addition to quantizing the expansion coefficients. However, for the combined wavelet-wedgeprint representation, we must also

quantize the line orientations ℓ of the wedgeprints used at the leaves of the quadtree, while making sure that this quantization does not introduce significant additional error.

In this section, we will show that the structure of the wavelet-wedgeprint approximation allows us to translate (4.26) into a rate-distortion bound without too much difficulty. Given the quadtree based approximation $X_{\hat{T}(J)}$ constructed in the previous section, we will describe a simple prototype coder that encodes $X_{\hat{T}(J)}$ using fewer than $R \sim J2^{J/2}$ bits with quantization distortion (playing the role of “error” in the approximation) $D \sim 2^{-J}$.

We will quantize each of the expansion coefficients in the approximation $X_{\hat{T}(J)}$ using a uniform scalar quantizer with fixed stepsize Q_J , and denote the result $X_{\hat{T}(J)}^q$. We have $N_{\text{total}} \leq C_N \cdot 2^{J/2}$ coefficients, and suffer total approximation error $\text{Error}_{\text{total}} \leq C_E \cdot 2^{-J}$.^{*} To ensure that the quantization distortion $D_q(J) = \|X_{\hat{T}(J)} - X_{\hat{T}(J)}^q\|_2^2$ does not dominate the approximation error, we need

$$N_{\text{total}} \cdot Q_J^2 \leq \text{Error}_{\text{total}} \quad (4.27)$$

and so we choose

$$Q_J = C_Q \cdot 2^{-3J/4} \quad (4.28)$$

where $C_Q = \sqrt{C_E/C_N}$.

With the quantizer stepsize remaining the same, the number of bits we will use for a coefficient depends on its possible range of values. From (4.5), we have that type S coefficients obey $|d_{j,k}^b| \leq C_5 \cdot 2^{-3j}$. The number of bins we will require for a coefficient at

^{*}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.

scale j is therefore

$$\text{bins}_S(j) = \frac{C_S \cdot 2^{-3j}}{Q_J} = \text{Const} \cdot 2^{3J/4-3j} \quad (4.29)$$

making the number of bits the coder spends on each coefficient

$$\text{bits}_S(j) = \log_2(\text{bins}_S(j)) = \text{Const} + \frac{3J}{4} - 3j. \quad (4.30)$$

The total number of bits used for type S wavelet coefficients for scales $j = 0, \dots, J/4$ is then

$$R_q^S(J) \leq \sum_{j=0}^{J/4} N_S(j) \cdot \text{bits}_S(j) \sim 2^{J/2}. \quad (4.31)$$

(The type S coefficients at scales $j = J/4 + 1, \dots, J/2$ can either be quantized to zero or quantized using the “type E” quantizer described below.)

The type E wavelet coefficients decay more slowly, and require more bits per coefficient (there are fewer of them, though). Using (4.6), we have

$$\text{bits}_E(j) = \text{Const} + \frac{3J}{4} - j. \quad (4.32)$$

For the type E coefficients at scales $j = 0, \dots, J/2$, the coder uses

$$R_q^E(J) \sim J \cdot 2^{J/2} \quad (4.33)$$

bits.

It is straightforward to show that the magnitude of the N_φ wedgeprint coefficients at

scale $J/2$ can be bounded by

$$|\langle X, \varphi_{J/2,k} \rangle| \leq \text{Const} \cdot 2^{-J/2}. \quad (4.34)$$

As a result, the total number of bits spent coding the wedgeprint coefficients is

$$R_q^\varphi(J) \sim J \cdot 2^{J/2}. \quad (4.35)$$

For each wedgeprint, we also need to specify an orientation. The line segment ℓ chosen in each square in (4.14) has a continuous valued orientation (r, θ) . Using the results of Section 3.4.2, we can quantize the orientation (r, θ) using $\sim J$ bits and still have (4.14) hold (with slightly larger constant), making the total orientation quantization error behave like

$$D^\ell(J) \sim 2^{-J}. \quad (4.36)$$

Since there are $\sim 2^{J/2}$ wedgeprints, we will spend a total of

$$R_q^\ell(J) \sim J \cdot 2^{J/2} \quad (4.37)$$

bits coding the orientations. If the coder uses a multiscale geometry model (as in Chapter 3), (4.37) can be sharpened to $\sim 2^{J/2}$.

In addition to quantizing the expansion coefficients, the encoder must also code the quadtree configuration of the approximation. Associated with each node in $\widehat{T}(J)$ is a label from $\{\text{Type-S, Type-E, Prune-Zerotree, Prune-Wedgeprint}\}$ which dictates how to

Table 4.2 *Prototype wavelet-wedgeprint coder summary.*

Type	Action	Total Bits	Error
S	quantize	$\sim 2^{J/2}$	$\sim 2^{-J}$
E	quantize	$\sim J \cdot 2^{J/2}$	$\sim 2^{-J}$
wedgeprint	quantize and prune	$\sim J \cdot 2^{J/2}$	$\sim 2^{-J}$
orientation ℓ	quantize (jointly)	$\sim 2^{J/2}$	$\sim 2^{-J}$
$\widehat{T}(J)$ labels	encode top-down	$\sim 2^{J/2}$	n/a

quantize the coefficient, and whether or not to consider its descendants. By coding the levels of the tree successively, we can represent the labeled quadtree using a number of bits proportional (by a factor of two) to the number of nodes in $\widehat{T}(J)$:

$$R_{\widehat{T}(J)} \sim 2^{J/2}. \quad (4.38)$$

Collecting the results above, the total number of bits that the encoder uses is

$$R(J) = R_q^S(J) + R_q^E(J) + R_q^\varphi(J) + R_q^\ell(J) + R_{\widehat{T}(J)} \sim J \cdot 2^{J/2} \quad (4.39)$$

while suffering total distortion

$$D(J) = \text{Error}_{\text{total}} + D_q(J) + D^\ell(J) \sim 2^{-J}. \quad (4.40)$$

Combining (4.39) and (4.40),

$$D(R; X) \sim \frac{(\log_2 R)^2}{R^2} \quad (4.41)$$

we see that our simple wavelet-wedgeprint coder achieves near optimal asymptotic rate-distortion performance.

We can summarize our results with the following theorem (see also Table 4.2):

Theorem 2 *The wavelet-wedgelet approximation $X_{\hat{T}(J)}$ of Theorem 1 can be encoded using fewer than $C_R \cdot J \cdot 2^{J/2}$ bits to error within $C_D \cdot 2^{-J}$, where C_R and C_D are constants independent of J . As a result, we can bound the distortion as a function of the number of bits used by the prototype coder described above $D(R; X) \leq \text{Const} \cdot (\log_2 R)^2 \cdot R^{-2}$.*

4.2 Selecting the wavelet-wedgeprint representation

We have shown that there exists a configuration of wavelets and wedgeprints that closely approximates a C^2/C^2 image. We are able to capture the contours in the image at fine scales using very few wedgeprint functions. Since we do not know the locations of the contours *a priori*, the encoder needs to make dynamic decisions about where to use wavelets, where to use wedgeprints, and which quantization scheme to use at a certain node.

The atoms in our combined wavelet-wedgeprint representation lie on a quadtree, with wedgeprint constrained to decorate the leaves. Therefore, we can select a representation for our simple prototype coder (a full scale image coder using wavelets and wedgeprints is presented in Chapter 5) by formulating a rate-distortion (fidelity vs. complexity) optimization problem

$$\min_T \quad \text{Distortion} + \lambda \cdot \text{Rate} \quad (4.42)$$

and solving it using the same quadtree optimization algorithms that we have become familiar with from previous chapters. If we treat the wedgeprint orientations independently, (4.42) can be solved with the CART algorithm [37] (see also Section 2.3). If we use a multiscale geometry model to code the orientations jointly, we solve (4.42) using the Viterbi algorithm (as in Section 3.3.3). A complete image coder (along with a full description of how to use the CART algorithm) will be presented in Chapter 5.

Figure 4.4 (a) Wavelet pruning of “Cameraman” ($N=7359$, $PSNR=28.77dB$), (b) Wavelet-wedgeprint pruning ($N=5158$, $PSNR=30.27dB$).

It is worth noting that in the end, the encoder cares little about the actual contour locations in the image. It simply tries to minimize (4.42) for a given λ . However, the encoder will tend to choose wavelets over smooth regions and wedgeprints over linear contour regions, because the local rate-distortion trade-off is favorable. The encoder thus naturally adapts to the geometrical structure in the image and does not rely on any pre-processed edge detection.

Figure 4.4 shows approximations of the “Cameraman” test image using standard wavelet tree pruning and wavelet-wedgeprint tree pruning. Using the wavelet-wedgeprint dictionary, we achieve significant gains both visually (sharper edges and reduced ringing) and in peak signal-to-noise ratio while using 30% fewer terms in the approximation.

4.3 Summary

In this chapter, we have added a new piece to our framework for “multiscale geometric image processing” that leverages the best of wavelets for representing smooth image regions and wedgelets for representing smooth edge contours. For C^2/C^2 images, the combined wavelet/wedgeprint dictionary provides optimal asymptotic approximation performance,

and a simple prototype image coder provides near-optimal rate-distortion performance. These results serve as justification for the practical image coder described in the next chapter.

While we have used compression to illustrate the effectiveness of our approach, a statistical wavelet/wedgeprint model will enable new algorithms for estimation, detection, classification, segmentation and other statistical image processing tasks.

Chapter 5

A Geometric Wavelet-Based Image Coder

In the previous chapter, we developed a simple prototype coder using a combined wavelet-wedgeprint representation on a quadtree that achieved near-optimal asymptotic rate-distortion performance $D(R; X) \sim (\log R)^2/R^2$. In this chapter, we extend this simple algorithm and propose a complete compression framework for natural images.

As discussed in Chapter 2, smooth regions in the spatial domain correspond to entire subtrees of small coefficients in the wavelet domain. Wavelet image coders exploit this structure by using a *zerotree* in place of the actual subtree of coefficients (equivalent to pruning the subtree). One symbol encodes many coefficients simultaneously, saving a significant number of bits.

The Space-Frequency Quantizer (SFQ) [8] is a state-of-the-art image coder based on the wavelet zerotree paradigm. The SFQ coding algorithm is conceptually very simple; it prunes quadtrees of “insignificant” coefficients, and quantizes the remaining “significant” coefficients independently using a set uniform quantizer. For a given uniform quantizer, it optimizes (in a rate-distortion sense) the placement of the zerotrees using a fast dynamic program.

Although the SFQ coder efficiently encodes the smooth regions of the image, it does not fully exploit the geometric regularity of the contours. At fine scales, many significant wavelet coefficients are needed to represent contour regions of the image. But since the contour itself is very simple, the wavelet coefficients are “geometrically coherent”, exhibiting strong dependencies that the SFQ coder ignores.

In this chapter, we extend the SFQ framework in an attempt to take advantage of contour regularity. In addition to quantizing significant coefficients and pruning zerotrees, we will replace subtrees of geometrically coherent wavelet coefficients with *geometry trees* — local wedgelet representations projected into the wavelet domain. The wedgelet modeling and inference algorithms developed in Chapter 3 are quadtree structured, allow us to fold them into the optimization program at the heart of SFQ. The resulting coder, called Wedgelet-SFQ (WSFQ), outperforms SFQ at a wide range of bitrates, and produces images with crisp, visually pleasing edges.

5.1 Rate-distortion optimal tree pruning

The WSFQ coder assigns a *map symbol* $\mu_{j,k}^b \in \{I, Z, G\}$ to each node (j, k) in the wavelet quadtree $\mathcal{T}^b$. Each $\mu_{j,k}^b$ indicates the quantization strategy for the descendants of (j, k) on $\mathcal{T}^b$.

Symbol $\mu_{j,k}^b = I$ indicates scalar quantization. The four children of node (j, k) in quadtree $\mathcal{T}^b$ are deemed significant, and are quantized uniformly. All significant coefficients use the same quantization step size, which is optimized for the target rate. The number of bits needed to code each child is (to a very good approximation) $-\log_2 p(\hat{d}_{j',k'}^b)$, where $p(\cdot)$ is the probability distribution over the bins and $\hat{d}_{j',k'}^b$ is the quantized child coefficient. In contrast to the other symbol choices, I specifies the strategy only for the children (immediate descendants) $\text{Children}(j, k)$. Additional map symbols will be coded for the children (hence the symbol I , for “interior”).

Symbol $\mu_{j,k}^b = Z$ indicates zerotree quantization. All descendants of node (j, k) in

*The WSFQ coder uses a different quadtree for each of the three wavelet subbands, in contrast to the prototype coder presented in Chapter 4.

quadtree $\mathcal{T}^b$ are quantized to zero. No further information (including map symbols) is needed for these nodes.

Symbol $\mu_{j,k}^b = G$ indicates geometry tree quantization. In this situation, the encoder constructs and encodes (using the wedgelet algorithms of Chapter 3) the best wedgelet approximation $\mathcal{W}_{(j,k)}$ on the support of $\psi_{j,k}^b$ (see [44] for details on how this is done), projects it into the wavelet domain, and uses the corresponding coefficients as the quantized values for the descendants of node (j, k) .

The map symbols are chosen carefully to optimally balance the total number of bits used to coder the image against the error in the quantization. Starting at the bottom of each quadtree and working upwards, we can calculate the consequence in *rate* (number of bits) and *distortion* (error) of assigning each of the three symbols at a given node. For a given λ , we want to find the configuration of map symbols that minimizes the functional

$$\min D + \lambda \cdot R \tag{5.1}$$

where D is the total distortion, and R is the total rate (the same value of λ is used for finding the wedgelet representation).

Since we are quantizing each subtree independently given the map symbols, the global optimization problem (5.1) for the entire quadtree can be broken down into smaller optimization problems over subtrees. We have the following *principle of optimality*: we can make an optimal decision for the map symbol at node (j, k) using rate and distortion information for the optimal decisions made for the four child subtrees.

The map symbols have the following rate-distortion trade-offs:

Significant: The cost in bits of choosing $\mu_{j,k}^b = 1$ is the number of bits used to quantize the child coefficients plus the number of bits needed to code each of their descendants

$$R_{j,k}^{(1)} = \sum_{(j',k') \in \text{Children}(j,k)} \left(-\log_2 p(\hat{d}_{j',k'}^b) + R_{j',k'} \right) \quad (5.2)$$

The distortion for this option is

$$D_{j,k}^{(1)} = \sum_{(j',k') \in \text{Children}(j,k)} \left((d_{j',k'}^b - \hat{d}_{j',k'}^b)^2 + D_{j',k'} \right) \quad (5.3)$$

Zerotree: The cost in bits of choosing $\mu_{j,k}^b = Z$ is zero; $R_{j,k}^{(Z)} = 0$. The distortion is the cost of pruning the subtree

$$D_{j,k}^{(Z)} = \sum_{(j',k') \in \mathcal{T}_{1(j,k)}^b} \left(d_{j',k'}^b \right)^2 \quad (5.4)$$

where $\mathcal{T}_{1(j,k)}^b$ are the descendants of node (j, k) .

Geometry tree: The cost in bits of choosing $\mu_{j,k}^b = G$ is $R_{j,k}^{(G)} = -\log_2 P(\mathcal{W}_{(j,k)})$, where $P(\mathcal{W}_{(j,k)})$ is the likelihood of the best wedgelet decomposition rooted at node (j, k) under the multiscale geometry model (see Section 3.3). The distortion is

$$D_{j,k}^{(G)} = \sum_{(j',k') \in \mathcal{T}_{1(j,k)}^b} \left(d_{j',k'}^b - \tilde{d}_{j',k'}^b \right)^2 \quad (5.5)$$

where the $\tilde{d}_{j',k'}^b$ are the wavelet coefficients corresponding to $\mathcal{W}_{(j,k)}$.

Starting at the bottom of the quadtree and working upwards, at each node we choose

$$\mu_{j,k}^b = \operatorname{argmin}_{\bar{\mu} \in \{I, Z, G\}} D_{j,k}^{(\bar{\mu})} + \lambda R_{j,k}^{(\bar{\mu})} \quad (5.6)$$

and set

$$R_{j,k} = R_{j,k}^{(\mu_{j,k}^b)} \quad (5.7)$$

$$D_{j,k} = D_{j,k}^{(\mu_{j,k}^b)}. \quad (5.8)$$

After it reaches the top of the quadtree, the encoder moves downwards, quantizing the coefficients according to the map symbols chosen on the upsweep.

5.2 Practical considerations

The procedure outlined in the previous section finds the best quantized representation for a given quantization step size and bin probability distribution $p(\cdot)$. In practice, we want to adapt this distribution to the actual significant coefficients we end up coding. Of course, this requires knowledge of the map symbols (which in turn require knowledge of p). To resolve this, we can apply the tree pruning procedure in the last section iteratively (see [8] for details), using an updated estimate of p at each iteration.

Another practical issue is the coding of the map symbols $\mu_{j,k}^b$. Instead of coding the $\mu_{j,k}^b$ independently using a constant number of bits, they can be effectively predicted from previously encoded coefficients (again, see [8] for details). Gains can be realized by passing through the symbol tree, changing the map symbols where the gain in prediction efficiency outweighs the loss in quantization efficiency.

Finally, it is possible to correct some of the error incurred when we quantize a wavelet subtree using a geometry tree. To correct geometric errors, we can implement standard SFQ on the *residual* wavelets coefficients left in the subtree after the wedgeprint has been removed (see [44] for details). The rate-distortion benefits for this correction can be computed before choosing the map symbol, allowing us to incorporate these corrections into the optimal tree pruning algorithm.

5.3 WSFQ compression results

As a first test of the WSFQ image coder, we use it to compress the synthetic image shown in Figure 5.1(a). The synthetic image has exactly the type of C^2/C^2 structure the WSFQ is designed to capture. Figure 5.3 shows the compressions results (at fairly low bitrates) using SFQ and WSFQ. The SFQ coder yielded a PSNR * of 32.84dB at 0.098 bits per pixel (bpp), while WSFQ yielded a PSNR of 34.19dB at the same bitrate (JPEG-2000 yields a PSNR of 30.89dB at this bitrate). Map symbol segmentations for SFQ and WSFQ are shown in Figures 5.3(b) and 5.3(d) respectively; by using geometry trees, WSFQ was able to capture fine scale contour structure that SFQ was forced to ignore.

On the natural images in Figure 5.1(b) and (c), WSFQ also outperforms SFQ. The “wet paint” image (results are shown in Figure 5.4) illustrate the strengths of WSFQ. Not only does WSFQ outperform SFQ in terms of PSNR, but the edges in the WSFQ compressed images are much cleaner. Another example, with the “peppers” image, is shown in Figure 5.5. Places where there are visual improvements are marked with white

*Peak-Signal-to-Noise ratio (PSNR) between an image X and an estimate $\hat{X}$ is a commonly used measure of distortion; $\text{PSNR}(X, \hat{X}) = 10 \log_{10} \frac{n \cdot \text{MaxInt}}{\|X - \hat{X}\|^2}$ where n is the number of pixels in the image and MaxInt is the maximum possible intensity value.

Figure 5.1 (a) Synthetic test image, (b) “Wet paint” test image, and (c) portion of “peppers” test image used in compression experiments.

Figure 5.2 Performance improvement of WSFQ (solid) and SFQ (dashed) relative to JPEG-2000 compression for (a) “Peppers” image and (b) “Cameraman” image.

boxes.

Figure 5.2 shows the SFQ and WSFQ performance on the “cameraman” and “peppers” image at a range of bitrates. Both algorithms outperform JPEG-2000 at most rates. At low bitrates, the WSFQ coder offers significant improvement over SFQ for images with isolated, sharp edges. At high rates, we see that WSFQ has diminishing performance gains relative to SFQ. At high bitrates, we can afford to code subtle textures near edges, so not as much is gained by geometrical approximation.

Figure 5.3 (a) Synthetic image coded using SFQ. (b) Multiscale wavelet-domain segmentation from SFQ tree-pruning. Zerotrees are represented in black; significant coefficients are gray. (c) Image coded using WSFQ. (d) WSFQ segmentation. Wedgeprints are represented in white; note that the three subbands used slightly different wedgeprint configurations.

Figure 5.4 (a) SFQ coded “wet paint” image, PSNR = 29.77dB at 0.0103 bits per pixel. (b) WSFQ coded, PSNR = 30.19dB at 0.0102 bits per pixel. Note that the WSFQ coder retains the sharpness of the edges much more effectively than the SFQ coder.

Figure 5.5 Portion of “peppers” test image coded using (a) SFQ (PSNR = 29.08 at 0.07 bpp) and (b) WSFQ (PSNR = 29.25 at 0.07 bpp). A white box indicates a dyadic block described by a geometry tree in one or more subbands.

Chapter 6

Statistical Models for Complex Wavelets

6.1 Introduction

The compact representation of the discrete wavelet transform (DWT) has led to many successful signal and image processing algorithms, including wavelet-based estimation (denoising), detection and classification, segmentation, fusion, and synthesis. Wavelets can be interpreted as multiscale singularity detectors, allowing us to describe the wavelet coefficients for a large class of signals and images in terms of two key properties [3, 4]:

2Populations: Smooth signal/image regions are represented by small wavelet coefficients, while edges, ridges, and other singularities are represented by large coefficients.

Persistence: Large and small wavelet coefficient values cascade along the branches of the wavelet tree. *

Interpreting these properties statistically, 2Populations implies that wavelet coefficients have non-Gaussian marginal statistics; that is, their marginal probability density functions have a large peak at zero (many small coefficients) and heavy tails (a few large coefficients). Persistence leads to wavelet coefficient values that are statistically dependent along the branches of the wavelet tree. In this chapter, we will use the 2Populations and Persistence properties to design a statistical model for signals and images. This model defines a *prior* for the joint probability distribution of the wavelet coefficients, with wavelet coefficients that

*Recall from Chapter 2 that the wavelet coefficients for a signal or image are naturally arranged on a tree.

exhibit the 2Populations and Persistence probabilities being more likely than those that do not. The model is based on two recent developments in multiscale signal processing: the wavelet-domain *hidden Markov tree* (HMT) model [3] and the *dual-tree complex wavelet transform* [45].

Image processing algorithms inspired by the 2Populations and Persistence properties have proven simple and effective. For example, denoising by thresholding [46], which is in many ways the canonical wavelet-domain processing algorithm, exploits the sparseness of the wavelet representation: since a close approximation of the image can be built out of a few large coefficients, the remaining small magnitude coefficients contain mostly noise and can safely be discarded. The revolutionary zero-tree wavelet compression algorithm [7] leverages the Persistence property by coding entire subtrees of small wavelet coefficients with just a few bits. The wavelet-domain HMT model [3] explicitly incorporates both of these properties and has proved very effective in denoising [4] and segmentation [47] applications.

However, current wavelet-domain image processing algorithms have a tendency to produce images with mild ringing artifacts around the edges. At the heart of the problem is the fact that the orthogonal wavelet transform is not *shift-invariant* — singularities at different locations in the image are represented differently with wavelets.

The negative impact of shift-variance is best illustrated by investigating the step response of a particular wavelet basis function. Figure 6.1(a) plots the different values the magnitude of a real orthogonal DWT coefficient takes for different shifts of a one-dimensional (1D) step edge. As the location of the step changes, the magnitude of the wavelet coefficient shown representing this edge oscillates — the wavelet basis function oscillates, so its step response also oscillates. As a result, the wavelet coefficients representing the edge are not always

Figure 6.1 (a) Magnitude of the 1D step response of Daubechies-8 (D8) wavelet coefficients at four consecutive scales. The horizontal axis represents the location of the step edge and the vertical axis represents the magnitude of the corresponding wavelet coefficient. (b) State sequence of the D8 coefficients for the edge location corresponding to the vertical line. (c) Step response of the complex wavelet coefficients with the same locales. (d) State sequence for complex coefficients. Note that the L-S-L-L state sequence for the D8 coefficients is a “broken” chain of large states, while the L-L-L-L state sequence for the complex coefficients is an unbroken chain.

”large”, and the branches down the wavelet tree that describe discontinuities are often not the unbroken progressions of large coefficients that algorithms based on the Persistence property expect.

One way to counteract the effects of shift-variance is to use the undecimated wavelet transform [2,4]. Denoising results using this method are typically better than the standard, orthogonal DWT, but the performance gain comes at a price: the $O(N \log N)$ undecimated algorithms are more expensive computationally than the $O(N)$ algorithms based on the

orthogonal DWT. * Also, the redundancy of the undecimated wavelet transform ($N \log N$ coefficients compared to N for the orthogonal DWT) makes it inappropriate for certain applications like image compression.

In this chapter, we address the problem of shift-variance by using the dual tree complex wavelet transform (CWT) [45, 48]. The CWT expands a signal or image in terms of a complex wavelet with complementary real and imaginary parts. In Section 6.3, we will show that when the real and imaginary wavelet basis functions are chosen appropriately, the complex wavelet coefficients will be approximately shift-invariant. *

The magnitude step response of a complex wavelet basis function, shown at four consecutive scales in Figure 6.1(c), is a smooth envelope. In contrast to the real DWT, the CWT step response magnitude varies monotonically with the distance of the edge from the location of the wavelet basis function. This leads to strong persistence of large magnitudes around edges independent of their alignment, as shown in Figure 6.1(c). As a result, edge structure is more regular in the CWT coefficients than in the DWT, making the “local edge detection” heuristic (and hence the 2Populations and Persistence properties) more accurate.

The CWT is slightly redundant (by a factor of two in 1D and four in 2D), and the CWT basis functions form a tight frame. The redundancy, however, is far less than the undecimated DWT, which is $\log N$ times redundant in 1D and $3 \log N$ in 2D. The computational cost of the CWT remains $O(N)$, the same as the orthogonal DWT.

In Section 6.4, we will extend the wavelet-domain HMT [3] to model complex wavelet coefficients. To capture the 2Populations property, we model each wavelet coefficient as a

*Throughout this chapter, we will use N to denote the number of data points in the signal in question; for an image, N is the number of pixels.

*By “shift-invariant” we mean that wavelet coefficients at a given scale carry the same information for all shifts of the signal being analyzed.

two-state Gaussian mixture; a controlling hidden state records the magnitude of the coefficient as either S (small) or L (large). To capture the Persistence property, we introduce dependencies across scale (between each parent and its children) in the form of transition probabilities between these hidden states. We ensure Persistence by assigning high probabilities to L-L and S-S transitions. Each path down the wavelet tree forms a Markov-1 chain in such a way that the unbroken progressions L-L-L-L and S-S-S-S are more likely than broken ones such as L-S-L-L and S-S-L-S.

As a prior density on the wavelet coefficients of the noise-free data, the HMT model plays the rôle of a statistical *grammar*. Under this grammar, unbroken strings of L and S states are preferable (more “grammatically correct”) than broken strings. Numerous experiments with real data have verified that the HMT is an effective model for a large class of real-world signals and images [3, 4, 47].

The complex wavelet HMT yields an even more powerful model. To demonstrate the effectiveness of the complex wavelet hidden Markov tree (CHMT) in modeling images, we present in Section 6.5 results for image denoising and texture segmentation. The results show a clear improvement over standard HMT methods (~ 1 dB gain in denoising performance over the real HMT, as shown in Figure 6.9 and Table 6.1).

We start in Section 6.2 by examining the lack of shift-invariance in the real orthogonal wavelet transform. Section 6.3 shows how to address the shift-invariance problem by using complex wavelets. In Section 6.4, we construct the CHMT model. Section 6.5 presents the improved denoising and classification results, and Section 6.8 contains discussion and conclusions.

6.2 The wavelet transform and shift-invariance

Similar to its 2D counterpart (see Chapter 2), the 1D discrete wavelet transform (DWT) decomposes a signal $f(t) \in L^2(\mathbb{R})$ in terms of a shifted and dilated “mother” wavelet $\psi(t)$ and scaling function $\phi(t)$:

$$f(t) = \sum_{\ell \in \mathbb{Z}} u_{j_0, \ell} \phi_{j_0, \ell}(t) + \sum_{j \geq j_0} \sum_{\ell \in \mathbb{Z}} d_{j, \ell} \psi_{j, \ell}(t) \quad (6.1)$$

where $\phi_{j_0, \ell}(t) := 2^{j_0/2} \phi(2^{j_0} t - \ell)$ and $\psi_{j, \ell}(t) := 2^{j/2} \psi(2^j t - \ell)$ [5]. If the $\{\phi_{j_0, \ell}, \psi_{j, \ell}, j \geq j_0, \ell \in \mathbb{Z}\}$ form an orthonormal basis for $L^2(\mathbb{R})$, then the *scaling coefficients* $u_{j_0, \ell}$ and *wavelet coefficients* $d_{j, \ell}$ can be calculated using the standard L^2 inner product: $u_{j_0, \ell} = \langle f, \phi_{j_0, \ell} \rangle$ and $d_{j, \ell} = \langle f, \psi_{j, \ell} \rangle$.

The wavelet transform is computed recursively using the filter bank structure shown in Figure 6.2(a). Given the scaling coefficients $\{u_{j+1, \ell}, \ell \in \mathbb{Z}\}$ at scale $j+1$, we compute the scaling coefficients $\{u_{j, \ell}, \ell \in \mathbb{Z}\}$ and wavelet coefficients $\{d_{j, \ell}, \ell \in \mathbb{Z}\}$ at scale j by passing the $\{u_{j+1, \ell}\}$ through digital filters H_0 and H_1 and downsampling by a factor of two. The impulse responses $h_0[n], h_1[n]$ of H_0 and H_1 are related to the scaling and wavelet basis functions in (6.1) by $\phi(t) = \sqrt{2} \sum_n h_0[n] \phi(2t - n)$ and $\psi(t) = \sqrt{2} \sum_n h_1[n] \phi(2t - n)$ [6]. To reconstruct the signal from its wavelet (and scaling) coefficients, we apply the filter bank structure shown in Figure 6.2(b), which computes the scaling coefficients $u_{j+1, \ell}$ at scale $j+1$ by upsampling $u_{j, \ell}$ and $d_{j, \ell}$, filtering with G_0 and G_1 (whose impulse responses are just time reversed versions of h_0 and h_1 in the orthogonal case), and adding the results.

In discrete time, we can interpret each sample of an N -point signal as a scaling coefficient at the finest scale $j = \log_2 N$, and apply the filter bank structure in Figure 6.2 recursively

Figure 6.2 Wavelet filter bank structure in 1D.

to efficiently compute the wavelet transform with $O(N)$ computational complexity [6].

For convenience, we will often enumerate the wavelet coefficients with one index, writing (for example) d_i in place of $d_{j,\ell}$. We will also use bold $(\mathbf{d})$ to denote the vector of all wavelet coefficients.

A major drawback of the wavelet representation is that it is not shift-invariant. Ideally, we would like the representation of a 1D input signal $* f(t)$ in terms of the set $\{d_{j,\ell}, \ell \in \mathbb{Z}\}$ of wavelet coefficients at a certain scale j to be the same for all shifts of $f(t)$. Restating, given the wavelet coefficients at scale j for one shift of the input signal, we would like to be able to predict (interpolate) the wavelet coefficients at scale j for *any* shift of the input.

To see why this is not true in general for the orthogonal wavelet transform, we examine (as in [49]) the computation of the wavelet coefficients of $f(t)$:

$$d_{j,\ell} = \langle f, \psi_{j,\ell} \rangle \sim \int_{-\infty}^{\infty} f(t) \psi(2^j t - \ell) dt = (f * \psi(2^j \cdot))(t) \Big|_{t=\ell}. \quad (6.2)$$

The $\{d_{j,\ell}, \ell \in \mathbb{Z}\}$ are *samples*, spaced every 2^{-j} in time, of the convolution of $f(t)$ with $\psi(2^j t)$. To be able to interpolate the wavelet coefficients at different shifts, we need the Fourier product $F(\omega)\Psi(2^{-j}\omega)$ to be strictly bandlimited. Since wavelets used in practical

*For clarity, we present the discussion on shift-invariance in 1D. The extension to 2D is straightforward.

applications have finite support in time, $F(\omega)\Psi(2^{-j}\omega)$ will in general have infinite support and be only approximately bandlimited. As a result, the $\{d_{j,\ell}\}$ will alias, and although we can design H_0, H_1 so that this aliasing cancels when we reconstruct, we cannot use $\{d_{j,\ell}\}$ to interpolate the wavelet coefficients for arbitrary shifts of the input.

Figure 6.1(a) illustrates the consequences of the DWT's lack of shift-invariance in the context of the 2Populations and Persistence properties. As small changes are made in the position of the 1D step edge, the energy in the edge "leaks" between scales. The profile of the wavelet coefficient magnitudes across scale is not a string of large coefficients as predicted by Persistence; rather it follows a pattern of large and small magnitudes that is different at every location.

In Section 6.3, we will show that we can combat this lack of shift-invariance by using complex valued wavelet filters (leading to complex valued wavelet basis functions). By analyzing only the positive frequencies in the input signal, we essentially cut its bandwidth in half and greatly reduce the aliasing.

6.3 Complex Wavelets

Although wavelet based algorithms have enjoyed a certain amount of success, they often produce images with ringing artifacts around the edges. The lack of shift-invariance in the orthogonal wavelet transform is the culprit: since an edge is essentially a singularity moving around in space, the real DWT represents it differently at every location. In this section, we will show that complex wavelets [17,19,45,48], by providing an approximately shift-invariant multiscale representation, strengthen the 2Populations and Persistence assumptions and hence provide a more accurate characterization of edge structure.

Figure 6.3 (a) Wavelet transform of the cameraman test image; wavelet coefficients with large magnitude are black, while those with small magnitude are white. (b) Each wavelet coefficient analyzes a local region in the image. The region is broken into four subregions at the next finest scale. This leads to (c) a natural quad tree organization of the wavelet coefficients in each subband. (d) The HMT model associates a hidden state (white circle) with each wavelet coefficient (black circle). The dependencies between the wavelet coefficients are modeled as dependencies between the hidden states.

6.3.1 The dual-tree complex wavelet transform

The 1D dual-tree complex wavelet transform (CWT) [45] decomposes a signal $f(t)$ in terms of a *complex* shifted and dilated mother wavelet:

$$f(t) = \sum_{\ell \in \mathbb{Z}} u_{j_0, \ell} \phi_{j_0, \ell}(t) + \sum_{j \geq j_0} \sum_{\ell \in \mathbb{Z}} c_{j, \ell} \psi_{j, \ell}(t) \quad (6.3)$$

with $\phi_{j_0, \ell}$ and $\psi_{j, \ell}$ complex: $\phi_{j_0, \ell} = \phi_{j_0, \ell}^r + \sqrt{-1} \phi_{j_0, \ell}^i$, $\psi_{j, \ell} = \psi_{j, \ell}^r + \sqrt{-1} \psi_{j, \ell}^i$. The $\psi_{j, \ell}^r$ and $\psi_{j, \ell}^i$ are themselves real wavelets (see Figure 6.4(a)): the complex wavelet transform is a combination of two real wavelet transforms, in 1D the $\{\phi_{j_0, \ell}^r, \phi_{j_0, \ell}^i, \psi_{j, \ell}^r, \psi_{j, \ell}^i\}$ form a tight frame with $2 \times$ redundancy. The real and imaginary parts of the dual-tree CWT are computed using separate filter bank structures (hence the name “dual-tree”) with wavelet filters H_0^r and H_1^r for the real part and H_0^i, H_1^i for the imaginary part.

In 2D, the CWT decomposes an image $X(s), s = (s_1, s_2) \in \mathbb{R}^2$ using dilations and translations of a complex scaling function and six complex wavelet functions

$$X(s) = \sum_{k \in \mathbb{Z}^2} u_{j_0, k} \phi_{j_0, k}(s) + \sum_{b \in \mathcal{B}} \sum_{j \geq j_0} \sum_{k \in \mathbb{Z}^2} c_{j, k}^b \psi_{j, k}^b(s). \quad (6.4)$$

The six subbands of the 2D CWT are labeled $\mathcal{B} = \{+15^\circ, +45^\circ, +75^\circ, -15^\circ, -45^\circ, -75^\circ\}$ for the oriented direction of the wavelet function as shown in Figure 6.4(b). For a separable 2D CWT based on 1D complex ϕ, ψ , we have $\psi^{+15^\circ}(s) = \phi(s_1)\psi(s_2)$, $\psi^{+45^\circ}(s) = \psi(s_1)\psi(s_2)$, $\psi^{+75^\circ}(s) = \psi(s_1)\phi(s_2)$, $\psi^{-15^\circ}(s) = \phi(s_1)\overline{\psi(s_2)}$, $\psi^{-45^\circ}(s) = \psi(s_1)\overline{\psi(s_2)}$, and $\psi^{-75^\circ}(s) = \psi(s_1)\overline{\phi(s_2)}$ where $\overline{\psi}$ is the complex conjugate of ψ . The 2D CWT expansion represents $X(s)$ using a tight frame with $4 \times$ redundancy.

As in the real case, we will index the complex wavelet coefficients using one number, writing c_i in place of $c_{j,k}^b$.

6.3.2 The CWT and shift-invariance

In Section 6.2, we saw that the wavelet coefficients of a signal $f(t)$ enjoy shift-invariance if either the scaling coefficients $\{u_{j,\ell}\}$ at every j or the wavelet filters H_0, H_1 are appropriately bandlimited. In this section, we will see that by using complex filters that suppress negative frequencies, we can achieve (approximate) shift-invariance.

One approach is to analyze only the positive frequencies in the input signal $f(t)$ using a DWT with real mother wavelet ψ^r [19, 50, 51]. Let $f^+(t)$ be the projection of f onto the positive frequencies; if $F(\omega)$ and $F^+(\omega)$ are the Fourier transforms of f and f^+ ; then

$$F^+(\omega) = \begin{cases} F(\omega) & \omega \geq 0 \\ 0 & \omega < 0. \end{cases} \quad (6.5)$$

For real $f(t)$, $F(\omega)$ is symmetric, and we can recover $f(t)$ from $f^+(t)$. Since $F^+(\omega)\Psi^r(2^{-j}\omega)$ has half the bandwidth of $F(\omega)\Psi^r(2^{-j}\omega)$, the aliasing in each subband is greatly reduced, and the wavelet coefficients are approximately shift-invariant. This approach analyzes a complex signal $f^+(t)$ with real wavelets $\psi_{j,k}^r(t)$.

Equivalently, we can analyze $f(t)$ with two wavelets ψ^r, ψ^i that are a Hilbert transform pair ([18] gives conditions on the filters H_0^r, H_0^i for this, and [17, 51, 52] provide some simple design techniques). Then the $\psi_{j,k}^+(t) = \psi_{j,k}^r(t) + \sqrt{-1}\psi_{j,k}^i(t)$ only contain positive frequencies, so again $F(\omega)\Psi^+(2^{-j}\omega)$ has half the bandwidth of $F(\omega)\Psi^r(2^{-j}\omega)$, leading to approximate shift-invariance. This approach analyzes a real signal $f(t)$ with complex wavelets

Figure 6.4 (a) Real (solid) and imaginary (dashed) parts of a complex wavelet basis function. (b) Real parts of the six 2D wavelet basis functions for a given scale and location.

$$\psi_{j,k}^+(t).$$

Figure 6.4(a) shows an example of a 1D dual-tree complex wavelet basis function. The real and imaginary basis functions share the same locale, but are “90° out of phase”; ψ^r is large when ψ^i is small and vice versa. Also of note are the symmetry properties: if we design ψ^r (as in Figure 6.4) to be even symmetric, then ψ^i will be odd symmetric.

The improved shift-invariance of the CWT renders the 2Populations and Persistence heuristics much more accurate. Returning to Figure 6.1, recall that the magnitude of a real wavelet coefficient varies widely with the location of the singularity inside the basis function’s locale — the basis functions $\psi_{j,\ell}$ are oscillatory, so their step responses are also oscillatory. As the 1D step edge in Figure 6.1(a) is shifted within the locale of the Daubechies-8 (D8) wavelet, the resulting wavelet coefficients swing wildly and come very close to zero for some shifts. We can also see in the figure that we do not have true persistence of magnitudes; some coefficients with small magnitudes have children with large magnitudes.

In contrast, as we can see from Figure 6.1(c), the magnitudes of the CWT coefficients vary smoothly with the location of the 1D step edge inside the basis function’s locale. The

Figure 6.5 An edge contour (a) in the plane reconstructed using wavelet coefficients at one scale j from (b) the real DWT and (c) the CWT. The CWT reconstruction is consistent at all locations and orientations, while the DWT reconstruction varies from location to location.

closer the edge is to the center of the locale, the larger the corresponding magnitude. As the edge moves from one locale to the next, the energy transfers smoothly between the corresponding CWT coefficients. Thus models based on the 2Populations and Persistence properties, such as the HMT discussed in Section 6.4, are more accurate for images when placed on the CWT coefficient magnitudes rather than the real wavelet coefficient magnitudes.

Figure 6.5 shows a simple 2D contour reconstructed from one scale of its real and complex wavelet transforms. As the contour moves through space, the reconstruction using the real DWT coefficients changes erratically, while the CWT reconstructs all local shifts and orientations in the same manner.

6.3.3 The CWT and directional selectivity

In 1D we can interpret the complex wavelet transform as performing a subband analysis on only the positive frequencies of the signal. In 2D, this allows us to distinguish between a feature oriented at an angle α and one oriented at angle $-\alpha$. For example, the CWT

Figure 6.6 Frequency plane tilings for the real and complex wavelet transforms. Each shade represents the frequency localization for a subband. The three subbands of the real wavelet transform, shown in part (a), captures features along lines at angles of $\{0^\circ, \pm 45^\circ, 90^\circ\}$. The complex wavelet transform, shown in part (b), can discriminate between features at positive and negative frequencies. Hence there are six subbands capturing features along lines at angles of $\{+15^\circ, +45^\circ, +75^\circ, -15^\circ, -45^\circ, -75^\circ\}$.

has separate subbands for features at $\pm 15^\circ$ whereas the real wavelet transform lumps these into one “horizontal” subband (see Figure 6.6). An edge in an image at a 15° angle causes the CWT coefficients in the “ $+15^\circ$ ” subband to be large, while an edge at -15° causes the CWT coefficients in the “ -15° ” subband to be large. In contrast, both of these edges would cause the real coefficients in the “ 0° ” (horizontal) subband of the DWT to be large. Thus, the 2D complex wavelet transform provides us with greater *directional selectivity* than the DWT.

By providing explicit information about singularities at a broader range of orientations, the CWT allows us to distinguish between and characterize images that are different in more subtle ways. From a model selection standpoint, the CWT allows us to choose from a richer set of models. For example, an image with features oriented at $+\alpha^\circ$ and another image with the same features oriented at $-\alpha^\circ$ will always have the same (or at least very similar) DWT HMT models. But using the complex HMT, they will in general have different models, allowing us to statistically discriminate between the two images.

6.4 The Complex Wavelet HMT Model

The wavelet-domain hidden Markov tree (HMT) [3] models the joint probability density function of the wavelet coefficients of an image * by explicitly incorporating the 2Populations and Persistence properties. The modeling proceeds in two stages: modeling the marginal density of each wavelet coefficient and modeling the dependencies between the wavelet coefficients. The HMT model for the real DWT has been successfully employed for image denoising [4] and segmentation [47]. In this section we will extend the HMT to model dual-tree complex wavelet coefficients. Because of the CWT's improved Persistence and better directional selectivity, the complex HMT (CHMT) is a more accurate image model, evidence of which is shown in the applications of Section 6.5.

6.4.1 The Wavelet-domain hidden Markov tree

The HMT models the marginal distribution of each real DWT coefficient as a *Gaussian mixture*. To each d_i , we associate a discrete hidden state q_i that takes on values $m = S, L$ with probability mass function (pmf) $p(q_i)$. Conditioned on $q_i = m$, d_i is Gaussian with mean $\mu_{i,m}$ and variance $\sigma_{i,m}^2$. Thus, its overall marginal pdf is given by

$$g(d_i) = \sum_{m=S,L} p(q_i = m) g(d_i|q_i = m), \quad (6.6)$$

with $g(d_i|q_i = m) \sim \mathcal{N}(\mu_{i,m}, \sigma_{i,m})$. While each wavelet coefficient d_i is conditionally Gaussian given its state variable q_i , overall it has a non-Gaussian density. Since the wavelet coefficients of images (and other piecewise smooth signals) are typically zero mean, we

*We will discuss the HMT in 2D for concreteness, although the ideas apply to data of any dimension.

constrain $\mu_{i,m} = 0$.

To match the Persistence property of large and small coefficients, the HMT applies a Markov-1 dependency structure to the hidden states across scale (a hidden Markov model). Using links to denote dependencies, the HMT model takes the form of the tree in Fig. 6.3(d). The state transition probabilities between the connected states model the persistence of large/small coefficient magnitudes across scale. Using $\rho(i)$ to denote the index of the parent of node i , the parameter $p(q_i = m | q_{\rho(i)} = n)$ gives the probability that a child coefficient d_i has hidden state m when its parent $d_{\rho(i)}$ has state n .

The HMT model parameters consist of the Gaussian mixture means and variances $\mu_{i,m}, \sigma_{i,m}^2$, the transition probabilities $p(q_i | q_{\rho(i)})$, and the pmf $p(q_0)$ for the root state q_0 . Each of the three DWT subbands is modeled with a separate (independent) model. Group these into the parameter vector Θ . In practice, we usually *tie*, sharing the same HMT parameters within each scale band [3]. A reduced-parameter HMT that exploits the self-similarity of image data is derived in [4].

The HMT can be trained to capture the wavelet-domain features of the image(s) of interest using the iterative expectation-maximization (EM) algorithm [3]. For a given set of training data, the trained model $g(\mathbf{d} | \Theta)$ approximates the joint pdf of the wavelet coefficients. See [3, 4, 47] for more details.

The 2Populations and Persistence properties induce a certain structure on the hidden state quadtree of an image. By 2Populations, we expect most of the nodes to be in the S state. From Persistence, smooth regions of the image should be modeled by entire subtrees in the S state, while in singular regions there should be chains of L states branching down through scale. However, referring back to the 1D example in Figure 6.1(b), we see that a

simple step edge causes the "broken" state sequence L-S-L-L. Since the HMT is designed to favor unbroken sequences over broken ones, the relative likelihood of the step edge will be low, and the edge will not be properly treated by HMT based algorithms.

In the next section, we will show that using the HMT model with the complex wavelet transform remedies the problem of broken state sequences around singularities.

6.4.2 Complex wavelet HMT

For the complex HMT, we associate with each CWT coefficient c_i a hidden state q_i taking value S,L depending on whether $|c_i|$ is small or large. Consider each complex coefficient c_i as a random vector (u_i, v_i) . We approximate the marginal density $g(c_i)$ as a two-state, 2D Gaussian mixture

$$g(c_i|q_i = m) = \frac{1}{\sqrt{2\pi}\sigma_{i,m}} \exp\left(-\frac{u_i^2 + v_i^2}{2\sigma_{i,m}^2}\right). \quad (6.7)$$

Factoring (6.7) into

$$g(c_i|q_i = m) = \frac{1}{\sqrt{2\pi}\sigma_{i,m}} \exp\left(\frac{-u_i^2}{2\sigma_{i,m}^2}\right) \exp\left(\frac{-v_i^2}{2\sigma_{i,m}^2}\right) \quad (6.8)$$

reveals that we model the real and imaginary parts as two-state, 1D Gaussian mixtures with *shared state variable* q_i ; see Figure 6.7. This is quite reasonable. Small $|c_i|$ result only when both $|u_i|$ and $|v_i|$ are small simultaneously (a smooth region of the image). This is captured by $q_i = S$. On the contrary, large $|c_i|$ result when either $|u_i|$ is large, $|v_i|$ is large, or both are large (edge region). This is captured by $q_i = L$. (Note that, as in the DWT HMT case, small outcomes from the L-state Gaussian mixture density are entirely possible — and in fact likely — since this component is zero mean.)

Figure 6.7 (a) The 1D real HMT associates a hidden state with each wavelet coefficient. (b) In the CHMT, the real and imaginary parts share a hidden state. In 2D, the wavelet coefficients and their associated hidden states are arranged in a quadtree as in Figure 6.3.

Figure 6.8 Scatter plot of the joint distribution of the real and imaginary parts of the complex wavelet coefficients c_i from one subband of the “boats” test image. The marginals for the real and imaginary parts are shown as histograms along the x (real) and y (imaginary) axes. Since the joint distribution is circularly symmetric, we treat the real and complex parts as independent.

The scatter plot in Figure 6.8 demonstrates that a mixture of 2D Gaussians is a good approximation to the empirical marginal distribution of the CWT coefficients in one scale. This approach is also equivalent to modeling the magnitude $|c_i|$ as a two-state Rayleigh mixture [53] and the phase $\angle c_i$ as a uniform random variable. This means that the model ignores phase information in the CWT.

The CWT HMT (CHMT) corresponding to (6.7) with scale-to-scale Markov transitions has an almost identical structure to the real DWT HMT. The only differences will be the substitution of (6.7) for (6.6) and the use of six subband trees instead of three. The result is an HMT that supports efficient $O(N)$ training and inference algorithms while featuring better shift invariance and angle selectivity. Hence the training, inference, and processing algorithms developed in [3, 4, 47] are easily extended to the CHMT.

Referring again to 1D step edge example in Figure 6.1, we see that the step edge is captured by the unbroken state sequence L-L-L-L. As a result, the relative likelihood of an edge, a common structure in images, is greater in the complex case than in the real case. The progressions of large and small coefficient magnitudes in the CWT coefficients $c_i = u_i + \sqrt{-1}v_i$ are thus more reliable than in either the real part u_i or imaginary part v_i alone.

As presented above, we model each of the subbands independently; there are 6 separate quadtrees for a CHMT image model. It is also possible to capture dependencies between subbands by using one quadtree with a vector state at each node consisting of the $\{S, L\}$ states for each subband [54].

6.5 Applications

CHMTs are applicable to the gamut of statistical signal and image processing tasks, from estimation and detection/classification to segmentation, fusion, and synthesis. To illustrate the accuracy of the CHMT, we will extend the HMT image denoising algorithms of [3, 4] and multiscale texture classification algorithm of [47]. Results using algorithms based on the CHMT show a marked improvement over those using the standard HMT.

6.5.1 Image denoising

Consider the problem of estimating an image corrupted by additive white Gaussian noise of variance σ_{noise}^2 . Translating the problem into the wavelet domain *, we wish to estimate the original signal's wavelet coefficients $\mathbf{y}$ from the noisy coefficients $\mathbf{c}$:

$$\text{Estimate } y_i \text{ given } c_i = y_i + \eta_i, \quad \eta_i \sim \mathcal{N}(0, \sigma_{\text{noise}}^2).$$

Because the structure of the CHMT is identical to the real HMT, we can estimate the image using the following empirical Bayesian programme [3]:

1. Using the EM algorithm, train a set of CHMT parameters Θ for the complex wavelet coefficients $\mathbf{y}$ based on the observed coefficients $\mathbf{c}$.
2. Compute the posterior state probabilities $p(q_i | \mathbf{c}, \Theta)$ using the fast $O(N)$ up-down algorithm.
3. Calculate the conditional mean estimate for the noise-free complex wavelet coefficient

*Although the complex wavelet basis functions are not orthogonal, they are close enough so that the noise is approximately white in the wavelet domain.

y_i

$$\hat{y}_i = E[y_i | \mathbf{c}, \Theta] = \sum_{m=S,L} p(q_i = m | \mathbf{c}, \Theta) \frac{\sigma_{i,m}^2}{\sigma_{\text{noise}}^2 + \sigma_{i,m}^2} c_i, \quad (6.9)$$

which is a linear combination of two Wiener shrinkage filters. Note that (6.9) shrinks the magnitude $|c_i|$ of the complex wavelet coefficient while leaving the phase $\angle c_i$ unchanged.

It is also possible to forgo the training step by using a set of pre-specified *universal* parameters based on a Besov space analysis of real-world images [4].

HMT-based denoising typically outperforms standard thresholding techniques, because the degree of coefficient shrinkage is determined based not only on the size of the coefficient but also on its relationship with its neighbors across scale [3, 4].

Proper determination of the posterior state probabilities $p(q_i | \mathbf{c}, \Theta)$ is the key to the good performance. To counter variations due to shift variance when using real wavelets, these quantities should ideally be estimated based on a number of different shifts of the data. By considering all possible shifts, we obtain the shift-invariant, undecimated HMT (RHMT) model of [4]. However, the complexity rises with the number of incorporated shifts. The dual-tree CWT provides a computationally attractive alternative.

Table 6.1 and Figure 6.9 compare the performance of a number of different denoising algorithms. While the computational complexity of the CHMT is virtually the same as the real DWT HMT, we see that complex modeling buys us a significant performance improvement both in PSNR * and visually (sharper edges and smoother backgrounds). We

*Peak signal-to-noise ratio for an estimate $\tilde{\mathbf{x}}$ of an N -pixel image $\mathbf{x}$ with values between 0 and 255 is defined as

$$\text{PSNR} = -20 \log_{10} \left(\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|_2}{255 N} \right).$$

Table 6.1 Comparison of PSNRs in dB of 256×256 images denoised using the wavelet algorithms of Figure 6.9 plus MATLAB’s spatially adaptive Wiener filter (Wiener2) for noise variances $\sigma_{\text{noise}} = 10$ and 25.

Image	Boats		Lena		Cameraman	
σ_n	10	25	10	25	10	25
Noisy	28.1	20.1	28.1	20.2	28.1	20.2
Wiener2	31.8	26.0	31.8	26.0	32.0	25.7
WHT	32.0	27.0	31.0	26.6	31.0	26.5
HMT	32.5	27.6	32.1	27.2	32.0	26.6
RHMT	33.2	28.0	32.7	27.7	32.6	27.1
CHMT	33.7	28.5	33.1	28.4	32.9	28.0

attribute this improvement to the near shift invariance and superior directional resolution of the CWT. Indeed, the CHMT outperforms the state-of-the-art (and computationally expensive) redundant HMT denoiser of [4].

6.5.2 Multiscale Texture Classification

In [47], we developed a multiscale texture segmentation algorithm based on the HMT. The algorithm has two key components: a multiscale classification step (termed “raw segmentation”) that uses the HMT to assign class labels at a range of scales, and a fusion step that combines these classifications into a final segmentation. In this section, we will show that the CHMT results in a better multiscale classification, helping us to arrive at a more accurate final segmentation.

Consider the standard *texture classification problem*. Given probability models $\{\Theta_l\}$ for a set of textures and an $K \times K$ data set $\mathbf{x}$, we decide which of the texture models best describes $\mathbf{x}$. This decision is based on the texture model that yields the highest likelihood; that is, we choose the index l that maximizes $g(\mathbf{x}|\Theta_l)$. As K becomes larger, the classification becomes more reliable.

A harder problem is *supervised texture segmentation*. Given the texture models $\{\Theta_l\}$

Figure 6.9 A 256×256 segment of the 512×512 Boats image denoised using a variety of approaches, with PSNRs. (a) Original image with 256 gray levels. (b) Noisy image ($\sigma_n = 25$). (c) Redundant wavelet hard thresholding (WHT) using D8 wavelet [1, 2]. (d) HMT denoising using D8 DWT [3]. (e) Redundant RHMT denoising using D8 [4]. (f) CHMT denoising. The CHMT denoising algorithm gives a significant improvement both visually and in PSNR. In this particular example, the areas around the wires suffers from fewer ringing artifacts in the CHMT restored image than in the other restored images.

and an image comprised of a montage of homogeneous texture regions of unknown shapes, we affix a label to each pixel in the image corresponding to the texture that best describes the surrounding area. The solution to this problem is less straightforward, since we do not have any information about the *shapes* of the regions. Given the shape information (and a model for the texture in each shape), we can use the maximum likelihood technique described above to classify each region independently.

Multiscale classification acts as a bridge between these two problems. At scale $j = 0, 1, \dots$, we divide the $K \times K$ image into dyadic squares of side length $2^{-j}K \times 2^{-j}K$ and then classify each dyadic square using the maximum likelihood technique. The result is a set of “raw segmentations” of the image, where we have assumed that the shapes of the texture regions are dyadic squares. The raw segmentations at coarse scales (small j) are based on many data points, and hence the decisions are more reliable than those at fine scales (large j). However, the raw segmentations at fine scales have better resolution; collections of the smaller dyadic squares can more accurately approximate the true shapes of the texture regions. Given this set of raw segmentations, a *fusion* algorithm (see [47, 55], for example) combines them into a final segmentation.

Wavelet-based texture models, such as the HMT and CHMT, are well suited for the multiscale classification problem. A subtree of wavelet coefficients with root at scale j contains information about the image inside a dyadic square of side length $\sim K2^{-j}$. By examining all subtrees of the wavelet coefficients of an image, we can assign a label to each dyadic square depending on which texture the subtree of wavelet coefficients best describes.

Wavelet HMT based methods approach the problem in two steps [47]:

1. Train HMT models for the wavelet coefficients of each texture. This requires samples

(and hence a priori knowledge) of the textures found in the image.

2. Compute the likelihood of the image wavelet coefficients under each model at multiple scales, and assign the label corresponding to the model that yields the highest likelihood.

The third step is the fusion step, for which we refer the reader to [47, 55].

HMT-based multiscale classification results using real and complex wavelets are shown in Figure 6.10. The CHMT algorithm outperforms (gives fewer errors than) the HMT algorithm, which leads us to conclude that complex wavelets capture more of the texture's key structure. In addition, the CWT's greater directional selectivity automatically makes the classifier more robust to rotations of the texture than the HMT. In related work, [54] outlines a rotational invariant texture retrieval algorithm using a steerable HMT.

We can use this multiscale CHMT classification as a front-end to a supervised segmentation algorithm [47, 55]. These algorithms use a priori information about the shape of the texture regions to combine the raw segmentations into a final segmentation map. For example, [47] presents an algorithm based on a context model for shape, while [55] uses one based on description length. The multiscale classifier based on the complex wavelet tree can be inserted into either of these algorithms without modification. Since the multiscale classification is more accurate using the CHMT, we can expect substantial improvement in the end segmentation results. In addition, [53] uses the CHMT for unsupervised segmentation.

6.6 A 1D coherent CHMT model

The CHMT introduced above models the structure between only the *magnitudes* of the CWT coefficients. However, edges in signals and images cause strong dependencies between

Figure 6.10 Classification results for an image composed of two textures (“grass” and “herringbone”). (a) Texture montage image. Classification of 4×4 pixel regions using an HMT with (b) real wavelets, and (c) complex wavelets. Classification of 2×2 pixel regions using (d) real wavelets and (e) complex wavelets.

the *phases* of the CWT coefficients. The phase of a CWT coefficient representing an edge is directly related to the *location* of the singularity relative to the locale of the basis function. A singularity is characterized not only by a chains of large magnitude coefficients across scale, but also by *coherence* of phase.

The magnitudes and phases of a CWT coefficient and its parent follow one of three distinctive patterns depending on the signal region being analyzed (see Table 6.2):

Smooth region: Both the coefficient and its parent have small magnitude (state S); the phase is undefined.

Isolated edge: When a single edge or other singularity resides within the locale of a CWT basis function and its parent, both coefficients have large magnitude (state L). Moreover, like the phase of a Fourier basis sinusoid, the CWT phase indicates position — in this case, the offset of the edge from the center of the locale. Since the edge positions at the two adjacent scales are simply related, so are the CWT phases.

The phase of the edge response $\angle E_{j,k}(l)$ of a dual-tree complex wavelet basis function $\psi_{j,k}$ is approximately linear with the location of the edge l [56, 57]:

$$\angle E_{j,k}(l) \approx 2^j \vartheta(l - k) - \frac{\pi}{2}. \quad (6.10)$$

We can use (6.10) to predict the phase of a child CWT coefficient c_{j+1,k_c} from the phase of its parent c_{j,k_ρ} on an edge. Combining

$$\arg c_{j,k_\rho} \approx 2^j \vartheta(l - k_\rho) - \frac{\pi}{2} \quad (6.11)$$

$$\arg c_{j+1,k_c} \approx 2^{j+1} \vartheta(l - k_c) - \frac{\pi}{2} \quad (6.12)$$

Table 6.2 *The three fundamental modes of CWT coefficients.*

feature	$ c_{J,k} $	$\angle c_{J,k}$
smooth	S	undefined
1 edge	L	coherent
texture	L	<i>incoherent</i>

we use a mapping $P(\cdot)$ to predict $\arg c_{j+1,k_c}$ from $\arg c_{j,k_\rho}$:

$$\arg c_{j+1,k_c} \approx P(c_{j,k_\rho}) = 2 \arg c_{j,k_\rho} + \frac{\pi}{2} \pm C \quad (6.13)$$

where the offset $C = 2^{j+1}\vartheta(k_\rho - k_c)$ is a constant, since $(k_\rho - k_c) \sim 2^{-j}$ if c_{j+1,k_c} is the left child of $c_{j,\rho}$ and $(k_\rho - k_c) \sim -2^{-j}$ if it is the right child. Due to the isolated edge, the parent and child CWT coefficients have *coherent phase*.

Texture region: When a rough textured region consisting of multiple edges or singularities resides within the locale of a CWT basis function and its parent, both coefficients have large magnitude (state L). However, due to the interaction between the edges, the CWT phases at the two adjacent scales will not be simply related. Indeed, they will appear almost statistically independent. We say that due to the texture these CWT coefficients have *incoherent phase*.

We can draw an interesting analogy between the texture phenomenon and the *speckle effect* that arises in coherent imaging systems, such as synthetic aperture radar (SAR). Speckle occurs whenever the target has features whose complexity lies on the same or smaller scale than the illumination wavelength; a complicated pattern of constructive and destructive interference occurs at the receiver due to the large phase shifts incurred on

reflection. In the CWT we also have a speckle effect when we “probe” a texture region with a complex wavelet.

To modify the HMT model to account for the phase coherence of edges across scale, we will separate the “large” L state into a state for large magnitude coefficients with incoherent phase, we will label the state T for “texture”, and a state for large magnitude wavelet coefficients with coherent phase across scale, labelled E for “edge”. We now have a model with three states, {S, T, E}. The conditional distributions for the S and T states remain the same as the usual HMT:

$$g(c_i|q_i = S) \sim \mathcal{N}(0, \sigma_{i,S}^2) \quad (6.14)$$

$$g(c_i|q_i = T) \sim \mathcal{N}(0, \sigma_{i,T}^2). \quad (6.15)$$

The pdfs (6.14)–(6.15) can also be thought of as Rayleigh distributions (with parameter $\sigma_{i,q}$) on the magnitude $|c_i|$ and an independent, uniform (between 0 and 2π) distribution on the phase $\arg c_i$.

For the E state, the magnitude of the wavelet coefficients is distributed the same as in the T state,

$$g(|c_i| | q_i = E) \sim \text{Rayleigh}(\sigma_{i,T}). \quad (6.16)$$

Given the edge location, we model the phase as being tightly distributed around $\angle E_i(l)$:

$$g(\arg c_i | s_i = E) \sim \beta_{2\pi}(\angle E_i(l), p, p), \quad (6.17)$$

a symmetric Beta density on the circle centered at $\angle E_i(l)$.

Obviously the associated edge location l for a wavelet coefficient in state E is unknown a priori. One approach would be to adopt l as another, continuous valued hidden state variable. The value is inherited from scale to scale, and is used to generate the phase if the wavelet coefficient is in state E (see Figure 6.11(b) for the graphical description). Developing algorithms for such a model is difficult, however, since they require finding the best state sequence in a continuous space.

Instead, we will model the phase as being distributed around a value $P(\arg c_{\rho(i)})$ predicted, using (6.13), from the phase of the parent coefficient $c_{\rho(i)}$:

$$g(\arg c_i | q_i = E) \sim \beta_{2\pi}(P(\arg c_{\rho(i)}), p, p). \quad (6.18)$$

If we consider the predicted phase value as part of the state, we have the graphical model shown in Figure 6.11(c). We no longer have a true hidden Markov structure, since the state of a wavelet coefficient depend on both the state of the parent and the actual value of the parent, but we can still derive efficient fast algorithms for state inference.

To show the effectiveness of this model in discriminating between large magnitude wavelet coefficients cause by texture and ones cause by edges, we can use the Viterbi algorithm to segment the wavelet coefficients into the most likely set of states. Figure 6.12 shows the results on a 1D slice of the popular “cameraman” test image. At course scales, almost all of the wavelet coefficient are classified as texture, since many edges reside inside the support of each of the basis functions. As we move to finer and finer scales, we see edges slowly being resolved; at the finest scale, all edges except the two on the right located extremely close together (meaning, perhaps, that we could not reasonably expect to separate

Figure 6.11 (a) shows the graphical model for the standard CHMT. The states (represented by the open circles) controls the magnitude of the wavelet coefficients (represented by the filled circles). In (b), the edge location, which governs the phase, is modeled as a separate hidden state represented by a square and is inherited down through scale. Part (c) shows the model used in Section 6.6; the current state depends on the state of the previous wavelet coefficient and the actual phase of the previous wavelet coefficient.

them into different entities) are represented by unique wavelet coefficients.

6.7 A Geometrical CHMT

The shift-invariance, rotation-invariance, and directional selectivity properties of the CWT make it well-suited to modeling more precise geometrical structure. Complex wavelet basis functions are more than local edge detectors, they are *local edge estimators*. The edge response of the six CWT coefficients associated with one location k is naturally parameterized by the edge's orientation (r, θ) relative to k . The angle θ controls the magnitudes, while the offset r dictates the phases (see Figure 6.13). Moreover, the mapping from (r, θ) to the set of six edge responses is (almost) invertible. The local geometry is manifest in the CWT coefficients; simply by looking at their values, we can not only decide whether or not an edge is in the vicinity, but we can also estimate the distance r and angle θ . An example of local edge structure being “read off” from CWT coefficients is shown in Figure 6.14.

Rather than using a single “Gaussian with large variance” state to model contours in the image, the geometric HMT (GHMT) breaks the L state down into many *geometry states*, each corresponding to a different type of local linear contour behavior. The states are drawn

Figure 6.12 State classification results using the model presented in Section 6.6. The figure on bottom is a multiscale display of the states of the wavelet coefficients. White areas correspond to coefficients in the S state, gray areas to T, and black areas to E. Note that all of the major edges in the slice are resolved at the finest scale with the exception of the pair on the right that are separated by a distance smaller than the width of the finest scale wavelet basis function.

Figure 6.13 Magnitude edge response for the six CWT basis functions at a given scale and location as a function of θ for $r = 0$. The phase response varies linearly with r . The relative magnitudes are tied to the angle θ of local geometrical structure while the phase is tied to the offset r .

Figure 6.14 Local edge detection using the CWT. Highlights of different regions in the “cameraman” image are shown on the left, on the right are edge estimates from the corresponding complex wavelet coefficients. The upper row shows the original region of the image, the lower row shows the edge estimates.

from a dictionary of possible (r, θ) values, $q_{j,k}^b \in S \cup \{(r_m, \theta_m)\}$, and we use the same state for all six CWT coefficients $\{\psi_{j,k}^b\}_b$ for a given scale and location (j, k) .

Each type of local linear contour structure produces distinct behavior in the CWT coefficients. Given a geometrical state $q_{j,k}^b = (r_m, \theta_m)$, we model the six CWT coefficients as coming from an improper Gaussian whose “mean” is the one dimensional subspace $E_{j,k}^{(r_m, \theta_m)}$ of response values of $\{\psi_{j,k}^b\}_b$ for edges of all different heights at (r_m, θ_m) . We have

$$g\left(\{d_{j,k}^b\}_b | q_{j,k}^b = (r_m, \theta_m)\right) \propto \exp\left(-\text{dist}(\{d_{j,k}^b\}_b, E_{j,k}^{(r_m, \theta_m)})^2 / (2\sigma_g^2)\right) \quad (6.19)$$

where $\text{dist}(\{d_{j,k}^b\}_b, E_{j,k}^{(r_m, \theta_m)})$ is the distance of the CWT coefficients $\{d_{j,k}^b\}_b$ to the subspace $E_{j,k}^{(r, \theta)}$ * and σ_g^2 controls the variance around this subspace.

For the transition probabilities between the geometrical states, we simply use a multi-scale geometry model (see Chapter 3).

The hidden states in the geometrical CHMT are more than just a modeling device; they

*We could make the distribution in (6.19) proper by putting a prior on the *heights* of the edges in the image.

Figure 6.15 Contour sketches at two different scales of the “cameraman” image shown in Figure 6.14. The sketches were created directly from the (r, θ) values of the geometrical states estimated by the Viterbi algorithm.

Figure 6.16 The geometric hidden Markov tree model. The tree of hidden states, shown on the left, models the multiscale behavior of the contours. Associated with each hidden state is a set of six complex wavelet coefficients (drawn as black dots on the right).

correspond to *semantic* contour information. Since edge contours are perceptually the most important features in an image, knowledge of the underlying states is interesting in its own right. Using the Viterbi algorithm to estimate the state sequence underlying an image is equivalent to estimating the presence of geometrical features in the image.

An example of geometrical feature extraction using the geometrical CHMT state estimation is shown in Figure 6.15. The contour sketches are re-created from the (r, θ) values inferred by the Viterbi algorithm. The estimated hidden state values are equivalent to a *vector graphics* representation of the contour sketch, and can be used to form an extremely low bitrate semantic approximation to the image.

6.8 Summary

Wavelet-domain image structure can be roughly described by two general properties. The 2Populations property says that we expect small magnitude wavelet coefficients to represent smooth regions in an image, while large magnitude coefficients represent singularities (2Populations). The Persistence property describes the inter-scale relationship of the wavelet coefficients: we expect large/small magnitudes to persist across multiple scales. Although image models based on these simple rules, such as the HMT, have yielded successful processing algorithms, 2Populations and Persistence are not quite accurate for real orthogonal wavelet transforms. At the heart of the problem is the DWT's lack of shift-invariance; singularities are treated differently at different locations.

We have demonstrated that a solution to the problem of shift-variance is to use models based on the CWT. Each (complex) CWT basis function is comprised of a pair of wavelets in quadrature; roughly speaking the real and imaginary parts share the same envelope, but are 90° out of phase. The effect is a drastic reduction in the sensitivity of wavelet coefficient magnitudes to singularity locations in an image. As a result, the 2Populations and Persistence properties constitute a much more accurate description of complex wavelet domain image structure. In addition, the CWT makes phase information available, enabling models that separate the multiscale behavior of coefficients representing isolated edges from that of coefficients for rough textures.

Extending the HMT — the canonical 2Populations/Persistence modeling framework — to model complex wavelet transforms is straightforward. The new model, the CHMT, inherits the HMT's fast training and inference algorithms. We have demonstrated the effectiveness of the CHMT in two applications. The denoising results show significant gains

over current leading methods in both PSNR (~ 1 dB gain) and visually — the CHMT estimator produces images with crisp edges virtually free of ringing artifacts. Multiscale classification (a front-end to supervised segmentation) using the CHMT also produces raw segmentations with fewer errors than with the usual HMT.

Not only does the CWT offer a more robust representation in which to model magnitude persistence, the CWT coefficients also carry geometrical “side information” which is easy to extract. In 1D, the phase of a CWT coefficient representing an edge is closely tied to the location of the edge relative to locale of the basis function. Using this phase coherence, we are able to separate out large magnitude coefficients caused by textures from those caused by isolated edges. In 2D, we can go one step further and separate out large magnitude coefficients caused by local contour structure at different orientations. This allowed us to develop a CHMT whose hidden states correspond to different types of local linear geometrical behavior. Tying these states together with a multiscale geometry model, we are able to incorporate a characterization of geometrical regularity into the model.

Chapter 7

Discussion and Conclusions

Over the six years I have spent in graduate school, I have found myself explaining the field of image processing to non-experts on more than a few occasions. The conversation usually goes something like this: “What kind of research are you doing for your PhD ?”

“I work in a field called ‘Digital Image Processing’.”

The people who are just making polite conversation leave it at that. The more curious people continue:

“What’s that?”

“Have you ever sent or downloaded pictures from the Internet, say, images on a web page or personal photographs?”

“Sure.”

“Well, one of the thing I work on is developing ways to do that more quickly.”

“Like ‘cable modems’?”

“That’s one way to speed it up, but I actually work on ways to reduce the amount of data you have to send for an image.”

Here, the comedians usually make some crass comment about facilitating the distribution of adult material. A few people, however, have been genuinely interested: “How do you do that?”

“When you take a picture, say with a digital camera, basically what comes out is a list of a few million colors, each one telling you what one tiny piece of the image looks like. But there are a lot of redundancies in this list. For example, if I take a picture of the wall over

there, I'll get a few hundred 'whites' in a row. Instead of saying 'white' hundreds of times, I just say 'there are 328 whites in a row here'."

"Ahh, I see. Neat. So that's what you do?"

"Yes, I do that except with math and computers."

The point of this anecdote is that it is easy to appreciate the fact that digital images contain a lot of structure, even for people who have no experience in signal processing. We know, intuitively, that "images" are more than just a list of numbers. To take advantage of this inherent structure, we develop a mathematical characterization, and use this characterization to separate images from random data sets.

This thesis has concentrated on one specific type of structure: edge contours. We have developed models and algorithms that are well-suited (asymptotically near-optimal) to images containing smooth regions separated by smooth contours. Of course, the geometry in real images is not as clean; the edges are not perfectly sharp, and they contain geometrical features other than edges (thin lines and ridge cusps, for example). As a possible avenue of research, we should consider more sophisticated models for images *given* the geometry (the geometrical HMT of Section 6.7 is a first step in this direction).

The quadtree models of Chapters 4 and 5 can be generalized to accommodate different types of structure. Instead of considering only edges (wedgeprints) or smooth regions (zerotrees), we could consider any number of options, including models for rough texture, quasi-periodic patterns, or other geometrical features. Conceptually, the choice of model at a node specifies a grammar that constrains the expected behavior of the vocabulary (wavelets) at finer scales.

Although the geometry of edges plays a fundamental role in our perception of an im-

age [58], it has for the most part been ignored by the wavelet community until very recently. Developing representations and models that explicitly express the geometry in an image will not only result in improved compression, denoising, and detection (for example) algorithms, but also opens the door to an new and interesting set of applications. Since the local adaptations the representation chooses reflect the edge structure in the image, *semantic* information about the shapes of object is readily available. As an immediate consequence, we can attack problems like object recognition and and classification directly in the *compressed domain*. One important application is database retrieval: we can compress a large library of images using an edge-adaptive algorithm, then search through them for specific geometric structure without decompressing.

References

1. M. Lang, H. Guo, J. E. Odegard, C. S. Burrus, and R. O. Wells, "Noise reduction using an undecimated discrete wavelet transform," *IEEE Signal Processing Lett.*, vol. 3, no. 1, pp. 10–12, Jan. 1996.
2. R. Coifman and D. L. Donoho, "Translation-invariant de-noising," in *Wavelets and Statistics*, Antoniadis/Oppenheim, Ed., vol. 103 of *Lecture Notes in Statistics*. Springer-Verlag, 1995.
3. M. S. Crouse, R. D. Nowak, and R. G. Baraniuk, "Wavelet-based statistical signal processing using hidden Markov models," *IEEE Trans. Signal Processing*, vol. 46, no. 4, pp. 886–902, Apr. 1998.
4. J. K. Romberg, H. Choi, and R. G. Baraniuk, "Bayesian tree-structured image modeling using wavelet-domain hidden Markov models," *IEEE Trans. Image Processing*, vol. 10, no. 7, pp. 1056–1068, July 2001.
5. I. Daubechies, *Ten Lectures on Wavelets*, SIAM, New York, 1992.
6. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, 2nd edition, 1998.
7. J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.
8. Z. Xiong, K. Ramchandran, and M. T. Orchard, "Space-frequency quantization for wavelet image coding," *IEEE Trans. Image Processing*, vol. 6, no. 5, pp. 677–693, 1997.
9. A. Cohen, W. Dahmen, I. Daubechies, and R. DeVore, "Tree approximation and optimal encoding," *J. Appl. Comp. Harm. Anal.*, vol. 11, pp. 192–226, 2001.
10. 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. Inform. Theory*, vol. 48, no. 7, pp. 1895–1921, July 2002.
11. E. J. Candés and D. L. Donoho, "Curvelets – a surprisingly effective nonadaptive representation for objects with edges," in *Curve and Surface Fitting*, A. Cohen et. al, Ed. Vanderbilt University Press, 1999.
12. E. J. Candés and D. L. Donoho, "New tight frames of curvelets and optimal representations of objects with C^2 singularities," *Technical Report, Stanford University*, 2002.
13. M. N. Do, *Directional Multiresolution Image Representations*, Ph.D. thesis, École Polytechnique Fédéral de Lausanne, October 2001.

14. E. Le Pennec and S. Mallat, "Image compression with geometrical wavelets," in *IEEE Int. Conf. on Image Proc. – ICIP '01*, Thessaloniki, Greece, Oct. 2001.
15. D. L. Donoho, "Wedgelets: Nearly-minimax estimation of edges," *Annals of Stat.*, vol. 27, pp. 859–897, 1999.
16. R. Willet and R. Nowak, "Platelets: A multiscale approach for recovering edges and surfaces in photon limited medical imaging," Preprint.
17. N. G. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals," *J. Appl. Comp. Harm. Anal.*, vol. 10, no. 3, pp. 234–253, May 2001.
18. I. W. Selesnick, "Hilbert transform pairs of wavelet bases," *IEEE Signal Processing Lett.*, vol. 8, no. 6, pp. 170–173, June 2001.
19. F. C. Fernandes, *Directional, Shift-Insensitive, Complex-Wavelet Transforms with Controllable Redundancy*, Ph.D. thesis, Rice University, 2001.
20. E. J. Candés and D. L. Donoho, "Curvelets and curvilinear integrals," *J. Approx. Theory*, vol. 113, pp. 59–90, 2000.
21. A. G. Flesia, H. Hel-Or, A. Averbuch, E. J. Candés, R. R. Coifman, and D. L. Donoho, "Digital implementation of ridgelet packets," in *Beyond Wavelets*, J. Stoeckler and G. V. Welland, Eds. Academic Press, 2002.
22. J. L. Starck, E. J. Candés, and D. L. Donoho, "The curvelet transform for image denoising," *IEEE Trans. Image Processing*, vol. 11, pp. 670–684, 2000.
23. J. L. Starck, F. Murtagh, E. J. Candés, and D. L. Donoho, "Gray and color image contrast enhancement by the curvelet transform," *IEEE Trans. Image Processing*, vol. 12, no. 6, pp. 706–717, June 2003.
24. M. N. Do and M. Vetterli, "Contourlets," in *Beyond Wavelets*, J. Stoeckler and G. V. Welland, Eds. Academic Press, 2002.
25. M. N. Do, "Contourlets and sparse image representations," in *Proc. of SPIE conference on Wavelet Applications in Signal and Image Processing X*, San Diego, CA, August 2003.
26. E. Le Pennec and S. Mallat, "Geometrical image compression with bandelets," in *SPIE Visual Communications and Image Processing – VCIP '03*, Lugano, Switzerland, July 2003.
27. P. L. Dragotti and M. Vetterli, "Wavelet footprints: Theory, algorithms, and applications," *IEEE Trans. Signal Processing*, vol. 51, no. 5, May 2003.
28. P. L. Dragotti and M. Vetterli, "Footprints and edgeprints for image denoising and compression," in *IEEE Int. Conf. on Image Proc. – ICIP '01*, Thessaloniki, Greece, Oct. 2001.

29. R. Shukla, P. L. Dragotti, M. Do, and M. Vetterli, "Rate distortion optimized tree structured compression algorithms," *IEEE Trans. Image Processing*, submitted for publication.
30. D. L. Donoho and X. Huo, "Beamlets and multiscale image analysis," in *Multiscale and Multiresolution Methods*, Lecture Notes in Computational Science and Engineering. Springer, 2001.
31. D. L. Donoho, "Unconditional bases are optimal bases for data compression and for statistical estimation," *J. Appl. Comp. Harm. Anal.*, vol. 1, no. 1, pp. 100–115, Dec. 1993.
32. 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, pp. 243–250, June 1996.
33. R. A. DeVore, B. Jawerth, and B. J. Lucier, "Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage," *IEEE Trans. Inform. Theory*, vol. 38, pp. 719–746, 1992.
34. D. L. Donoho, M. Vetterli, R. A. DeVore, and I. Daubechies, "Data compression and harmonic analysis," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2435–2476, Oct. 1998.
35. D. L. Donoho, "De-noising by soft-thresholding," *IEEE Trans. Inform. Theory*, vol. 41, no. 3, May 1995.
36. R. Neelamani, H. Choi, and R. G. Baraniuk, "ForWaRD: Fourier-wavelet regularized deconvolution for ill-conditioned systems," *IEEE Trans. Signal Processing*, 2003, To appear.
37. L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1984.
38. T. Cover and J. Thomas, *Elements of Information Theory*, Wiley, New York, 1991.
39. L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–285, Feb. 1989.
40. M. N. Do, P. L. Dragotti, R. Shukla, and M. Vetterli, "On the compression of two-dimensional piecewise smooth functions," in *IEEE Int. Conf. on Image Proc. – ICIP '01*, Thessaloniki, Greece, Oct. 2001.
41. B. J. Frey, *Graphical Models for Machine Learning and Digital Communication*, MIT Press, Cambridge, MA, 1998.
42. F. Jelenik, *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, 1998.
43. R. Johannesson and K. Zigangirov, *Fundamentals of Convolutional Coding*, Wiley-IEEE Press, 1999.

44. M. B. Wakin, J. K. Romberg, H. Choi, and R. G. Baraniuk, "Geometric methods for wavelet-based image compression," *In preparation*, 2003.
45. N. G. Kingsbury, "Image processing with complex wavelets," *Phil. Trans. Royal Society London A*, vol. 357, pp. 2543–2560, Sept. 1999.
46. D. L. Donoho and I. Johnstone, "Ideal spatial adaptation via wavelet shrinkage," *Biometrika*, vol. 81, pp. 425–455, 1994.
47. H. Choi and R. G. Baraniuk, "Multiscale image segmentation using wavelet-domain hidden Markov models," *IEEE Trans. Image Processing*, vol. 10, no. 9, pp. 1309–1321, Sept. 2001.
48. N. G. Kingsbury, "Shift invariant properties of the dual-tree complex wavelet transform," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. — ICASSP '99*, Phoenix, AZ, Mar. 1999.
49. E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable multiscale transforms," *IEEE Trans. Inform. Theory*, vol. 38, pp. 587–607, Mar. 1992.
50. P. Abry and P. Flandrin, "Multiresolution transient detection," in *Proc. of the IEEE-SP Int. Symp. Time-Frequency and Time-Scale Analysis*, Philadelphia, PA, Oct. 1994, pp. 225–228.
51. R. L. C. van Spaendonck, *Seismic Applications of Complex Wavelet Transforms*, Ph.D. thesis, Delft University of Technology, 2002.
52. I. W. Selesnick, "The design of Hilbert transform pairs of wavelet bases via the flat delay filter," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. — ICASSP '01*, Salt Lake City, UT, May 2001.
53. C. W. Shaffey, N. G. Kingsbury, and I. H. Jermyn, "Unsupervised image segmentation via Markov trees and complex wavelets," in *IEEE Int. Conf. on Image Proc. — ICIP '02*, Rochester, NY, Sept. 2002.
54. M. N. Do and M. Vetterli, "Rotation invariant texture classification and retrieval using steerable wavelet-domain hidden Markov models," *To appear in IEEE Trans. on Multimedia*, 2002.
55. R. Neelamani, J. K. Romberg, H. Choi, R. Riedi, and R. G. Baraniuk, "Multiscale image segmentation using joint texture and shape analysis," in *Proc. SPIE Conference on Wavelet Apps. in Signal and Image Proc. VIII*, San Diego, CA, Aug. 2000.
56. 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.
57. T. H. Reeves and N. G. Kingsbury, "Prediction of coefficients from coarse to fine scales in the complex wavelet transform," *preprint*.

58. D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W.H. Freeman, San Francisco, 1982.