Geometric Tools for Image Compression

Michael Wakin, Justin Romberg, Hyeokho Choi, Richard Baraniuk

Department of Electrical and Computer Engineering, Rice University
6100 Main St., Houston, TX 77005

Abstract

Images typically contain strong geometric features, such as edges, that impose a structure on pixel values and wavelet coefficients. Modeling the joint coherent behavior of wavelet coefficients is difficult, and standard image coders fail to fully exploit this geometric regularity. We introduce wedgelets as a geometric tool for image compression. Wedgelets offer piecewise-linear approximations of edge contours and can be efficiently encoded. We describe the fundamental challenges that arise when applying such a tool to image compression. To meet these challenges, we also propose an efficient rate-distortion framework for natural image compression using wedgelets.

1. Introduction

Much of the information in a typical image is communicated by the *edges* of the pictured objects, and these edges generally contain a great deal of high-frequency energy. For these reasons, an effective algorithm for image compression must (explicitly or implicitly) have an efficient means for compressing edge information. In this paper, we demonstrate that standard techniques for image compression can be improved through a better awareness of the *geometric structure* that edges impose on an image.

Today's standard approach for image compression involves the 2-D wavelet transform. Wavelets are indeed well-suited to compress smooth and textured regions, but due to a lack of shift-invariance, their behavior is rather complicated near edges. Many large wavelet coefficients are typically required to describe an edge contour. These coefficients have a coherency which is imposed by the structure of the edge, but most compression schemes fail to fully exploit this geometric regularity. The Estimation-Quantization (EQ) coder [1], for example, simply models the increased variance of wavelet coefficients along an edge. As a result, there is a potential loss of efficiency in encoding these coefficients, and ringing artifacts are typically introduced when quantization destroys their coherency.

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

Email: {wakin, jrom, choi, richb}@rice.edu. Web: dsp.rice.edu

The potential for improvement upon such techniques is highlighted by the following observation: edges naturally lend themselves to parsimonious description. A straight edge, for example, can be completely described by only a few parameters. Our goal is to reconcile this discrepancy – to develop an image compression framework which uses parsimonious descriptions for geometric information.

While the motivation for the problem may be clear, the optimal solution is somewhat unclear. A variety of recent attempts at image compression have attempted to capitalize on geometric regularity. *Curvelets* and *contourlets* [2] define a basis of elements which provide excellent nonlinear approximations for geometric features. The dictionary, however, is slightly redundant, and has yet to yield a practical compression algorithm. *Bandelets* [3] warp a wavelet basis around an edge contour, but the procedure is quite complicated, and questions remain as to how bits should be allocated among the stages of the procedure.

In this paper, we examine the possibility of image compression using *geometric tools:* dictionaries of elements that provide parsimonious descriptions of geometric features. The *wedgelet* dictionary, developed by Donoho [4], is one such tool. Wedgelets offer piecewise-linear approximations to edge contours, and the wedgelet dictionary can be arranged to allow efficient indexing and compression. Section 2 discusses wedgelets in detail and introduces an efficient technique for compressing a wedgelet approximation.

The use of any geometric tool for image compression faces certain fundamental challenges. The geometric tool, for example, must be integrated into a complete image coder, and this coder must intelligently decide where and how to allocate its bitrate. Section 3 uses the wedgelet dictionary to illustrate such challenges.

Finally, in Section 4, we explain how wedgelets may be integrated into the Space-Frequency Quantization algorithm [5], a powerful wavelet-compression technique. The resulting coder meets many of the challenges identified in Section 3. Our Wedgelet-SFQ coder optimizes its bit allocation between wedgelets (geometry) and wavelets (texture), and uses an optimized framework for wavelet compression of residual geometric information. The compression gains we achieve over SFQ illustrate the true promise of geomet-

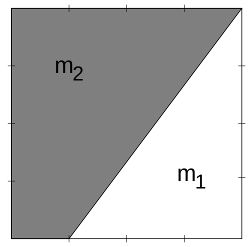

Fig. 1. *Wedgelet on an $N \times N$ dyadic block: a position index k describes the endpoints of the edge, and m_1 and m_2 specify the average grayscale intensities on each side.*

ric compression techniques.

2. Wedgelets for geometric representation

A *wedgelet* is a dyadic $N \times N$ block of pixels containing a picture of a single straight edge. As shown in Fig. 1, the edge separates two constant regions of grayscale intensity m_1 and m_2; pixel values along the edge are computed by an appropriate weighted averaging. We restrict the possible endpoints of the edge so that the position may be indexed by a single discrete parameter $k \in \{1, 2, \ldots, M\}$. A properly selected discrete set leads to fast algorithms for computing the multiscale wedgelet fits for a given set of data [6]. We denote by Θ the collection of parameters $\{k, m_1, m_2\}$ required to describe a wedgelet.

The *wedgelet dictionary*, introduced by Donoho [4], is the dyadically organized collection of all possible wedgelets. As illustrated in Fig. 2, contours in an image may be approximated by a *wedgelet decomposition*, a tiling of wedgelets chosen from this dictionary. A dyadic wedgelet decomposition can be interpreted as a quadtree, where each node d_i includes a set of wedgelet parameters Θ_i describing the corresponding dyadic block. Leaf nodes are used to assemble the picture of the wedgelet decomposition, while interior nodes are useful for predicting and encoding parameters at the leaf nodes. Finer approximations to a contour can be obtained by dividing a leaf node into four children.

Wedgelets offer a *parsimonious* description of geometry; we naturally look to use wedgelet decompositions for geometric image compression. We propose a top-down, predictive scheme for encoding a wedgelet decomposition. Our algorithm exploits the redundancy among wedgelet parameters in the quadtree. In particular, we note that a node d_i and its children describe the same spatial location. Thus, once wedgelet parameters are encoded for a node, these parameters offer predictions for its four children. Letting d_j be a child of d_i, we can predict $\widehat{\Theta_j}$ by drawing a picture of the wedgelet described by Θ_i, dividing the picture into four quadrants, and extracting the wedgelet parameters from the appropriate quadrant. With knowledge of Θ_i, we need only encode the prediction errors to know Θ_j.

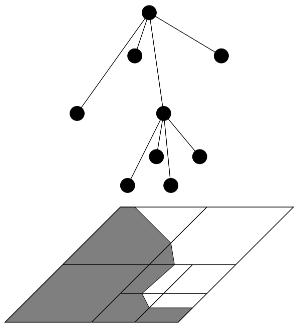

Fig. 2. *A dyadic wedgelet decomposition can be interpreted as a quadtree, where each node includes a set of wedgelet parameters, and leaf nodes specify the pictured wedgelets.*

For the purposes of this paper, we explain how to encode the prediction error only for the wedgelet index k. To encode the difference between the wedgelet index k_j and its prediction $\widehat{k_j}$, we define a probability distribution on the Hausdorff distance $\delta(k_j, \widehat{k_j})$ according to $p(\delta) \sim e^{-\gamma\delta}$, where γ is a constant that controls the preference given to accurate predictions. Using arithmetic coding according to this distribution, accurate predictions require few bits to encode, while large prediction errors are more costly.

A Viterbi-like algorithm determines the optimal wedgelet decomposition, using a Lagrangian parameter λ to balance the rate required to subdivide each parent wedgelet against the decrease in distortion. The Viterbi algorithm also reveals the optimal wedgelet parameters to encode at each node.

3. Challenges for natural image compression

The previous section suggests a compression method for Horizon class images (those consisting entirely of sharp edges along smooth contours). Our goal, however, is to code natural images, which contain smooth regions, textures, and gradients, in addition to the geometric features. Unfortunately, due to errors introduced when approximating real edges with wedgelet step edges, the application of wedgelets to natural image compression is not straightforward.

In [7], we describe a scheme where we encode a wedgelet decomposition of an image f, creating a cartoon-like primitive sketch c. We then subtract this sketch from the original image, and compress the residual $t = f - c$ using a standard wavelet-based coder. This leads to an interesting problem, however: residual artifacts created by wedgelet approximations pose many of the same difficulties for wavelets that edges present. These artifacts typically resemble tall, thin ridges which have a geometric structure similar to the edges themselves. Coherency must be preserved among their quantized wavelet coefficients in order to prevent ringing. It is possible to make a refinement to the residual image [8], masking out any possible artifacts located near edges in the cartoon sketch. This makes the resulting residual image much easier to compress using wavelets, and we see a noticeable improvement in visual quality over standard coding techniques. The destruction of information (during masking), however, prevents the full coder from being competitive in terms of PSNR.

As demonstrated by this exercise, any compression technique that uses an explicit geometric tool must deal with certain fundamental challenges. First, the coder must have an effective interface between its geometric and its non-geometric techniques. Second, the coder should optimize its bit allocation between these techniques. The geometric tool should be used only when it actually *improves* the R/D performance of the full coder. Third, the coder must deal carefully and effectively with errors made by geometric approximations. Knowledge of nearby geometry must be considered when compressing residual textures.

4. Rate-distortion compression using wedgelets

In this section, we present a complete image coder that uses wedgelets as a geometric tool. Based on the Space-Frequency Quantization (SFQ) algorithm [5], our Wedgelet-SFQ approach (first presented in [9]) capitalizes on the SFQ optimization structure to address the challenges described above.

4.1. Space-Frequency Quantization

Space-Frequency Quantization (SFQ) is an efficient algorithm for wavelet-domain compression. The SFQ coder uses a zerotree [10] quantization framework, with scalar quantization used to compress the significant (non-zerotree) wavelet coefficients.

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

Fundamental to the SFQ algorithm is its rate-distortion optimization of the zerotree placements; a tree-pruning operation weighs the rate and distortion consequences of each symbol. The pruning starts at the bottom of the tree and proceeds upwards. Initially, it is assumed that all coefficients are significant, and decisions must be made regarding whether to group them into zerotrees. The coder uses several bottom-up iterations until the tree-pruning converges. At the beginning of each iteration, the coder estimates the probability density $p(\widehat{w})$ of the collection of significant coefficients; this yields an estimate of the entropy (and hence coding cost) of each quantized coefficient. Ultimately, adaptive arithmetic coding is used to transmit these quantization bin indices. The SFQ tree-pruning produces a near-optimal configuration of zerotrees without requiring an exhaustive search over all configurations.

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

Optimization in the SFQ framework begins with Phase I, where the tree is iteratively pruned based on the rate and distortion costs of quantization. Phase I ignores the bits required to transmit map symbols, while Phase II adjusts the tree-pruning to account for these costs. In this paper, we focus on Phase I and its adaptation to include wedgelets; the adaptation for Phase II is similar. In each iteration of the Phase I optimization, those nodes currently labeled significant are examined (those already in zerotrees will remain in zerotrees). The coder has two options at each such node: create a zerotree (symbol 0) or maintain the significance (symbol 1). Each option requires a certain number of bits and results in a certain distortion relative to the true wavelet coefficients. The first option, zerotree quantization of the subtree beginning with node n_i, requires $R_i^{(0)} = 0$ bits, because no information is transmitted besides the map symbol. This option results in distortion

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

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

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

This option results in distortion

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

The decision between the two options is made to minimize the Lagrangian cost $J_i = D_i + \lambda R_i$, where λ is the optimization parameter controlling the tradeoff between rate and distortion. Note that, for *each subtree of wavelet coefficients*, SFQ chooses the R/D optimal compression scheme.

Despite its success, the SFQ coder fails to model the joint behavior of wavelet coefficients along an edge. A standard SFQ optimization generally results in the use of zerotree symbols to represent smooth regions of the image, with scalar quantization used to code other features such as edges. We propose adding a third option (symbol 2) to the SFQ tree-pruning, where a wedgelet decomposition is transmitted at node n_i that can be used to infer the descending wavelet coefficients.

4.2. Wedgelet-SFQ algorithm

W-SFQ uses wedgelet decompositions to provide parsimonious descriptions of wavelet coefficient subtrees. Each node n_i in the wavelet quadtree corresponds to a dyadic block of pixels. For each such node, we compute the wedgelet decomposition within the corresponding image block, and we consider the possibility of explicitly encoding that wedgelet decomposition. We use the techniques of Section 2 to optimize and encode this wedgelet decomposition, using the same optimization parameter λ as for the SFQ tree-pruning.

Once encoded for a node n_i, a wedgelet decomposition can be used to predict the wavelet coefficients for all descendants U_i. This is due to the approximate support of each wavelet basis function within its corresponding dyadic block. One way to obtain a prediction for these coefficients is to create an temporary image containing the coded wedgelet decomposition at the appropriate location, take its wavelet transform, and extract the appropriate coefficients. For each j such that $n_j \in U_i$, we denote the predicted wavelet coefficient as $w^*_{i,j}$. The collection $\{w^*_{i,j} : n_j \in U_i\}$ is called a *curve tree*: a subtree of wavelet coefficients derived from the wedgelet decomposition encoded at its root n_i.

A curve tree provides an alternative method for representing a subtree of wavelet coefficients. W-SFQ adds curve trees as a third option (symbol 2) to the SFQ tree-pruning; the modification to SFQ is straightforward. In the case of a curve tree, $R_i^{(2)}$ is simply the rate required to encode the wedgelet decomposition, although in practice we may add an extra penalty because the likelihood of symbol 2 is generally much lower than symbols 0 and 1. The distortion for the curve tree option is given by

$$D_i^{(2)} = \sum_{j:\ n_j \in U_i} (w_j - w^*_{i,j})^2.$$

As with SFQ, the option with the lowest Lagrangian cost is chosen.

Curve trees offer some of the same benefits as zerotrees – large collections of wavelet coefficients can be described using very few bits – but curve trees can do so in the high-energy regions near edges. Scalar quantization can be reserved for more complicated texture regions. Moreover, because curve trees are obtained by taking the wavelet transform of an image with sharp edges, they implicitly model the joint coherency of the wavelet coefficients. This should minimize visual artifacts at low bitrates.

4.3. Residual compression in W-SFQ

When a curve tree is transmitted at a node, we are assured that it improves the R/D performance of the coder (otherwise one of the two standard SFQ symbols would have been chosen). Facing the challenges described in Section 3, we are left with the question of whether to attempt compression of the residual wavelet errors on that subtree. Rather than ignore these residual errors, we implement standard SFQ compression on each residual subtree resulting from a curve tree. SFQ allows a spatially adaptive approach, so that textures away from the edges may still be encoded. In addition, the zerotree symbol allows the coder the option of ignoring any geometric artifacts which it cannot efficiently compress. This residual SFQ may destroy some of the wavelet coherency among geometric features, but it is also guaranteed only to improve the coder's R/D performance.

4.4. Compression performance

For efficiency when implementing W-SFQ, we restrict each wedgelet decomposition to a pair of common grayscale values m_1 and m_2. (Indeed it is only necessary to transmit $m_2 - m_1$ in order to compute the wavelet coefficients). Thus, the parameter set Θ for each node contains only the wedgelet index k, with the quantity $m_2 - m_1$ transmitted only once.

For our implementation, we also expand our wedgelet dictionary to accommodate smooth edges. For each curve tree, we transmit as a single parameter the width T (in pixels) of the transition from m_1 to m_2. With this added parameter, we are able to better approximate blurred edges.

For an example of the effectiveness of curve tree representations, we compress the 512×512 *Peppers* image using both SFQ and W-SFQ at a bitrate of 0.07bpp. For a point of reference, JPEG-2000 compression yields a PSNR of 28.57dB at this bitrate.

Fig. 3 shows a portion of the SFQ-compressed image. SFQ compression gives a PSNR of 29.08dB, and the SFQ tree-pruning leaves a total of 5512 wavelet coefficients described by scalar quantization.

At the same bitrate, Fig. 4 shows the *Peppers* image compressed using W-SFQ. A PSNR of 29.25dB is attained, an improvement of 0.17dB over the standard SFQ technique. In regions described by curve trees, ringing artifacts are noticeably reduced compared to the SFQ result. In this case, 44 separate curve trees are encoded, leaving 4436 wavelet coefficients described by scalar quantization. The compression for curve tree prediction errors encodes 160 residual coefficients using scalar quantization.

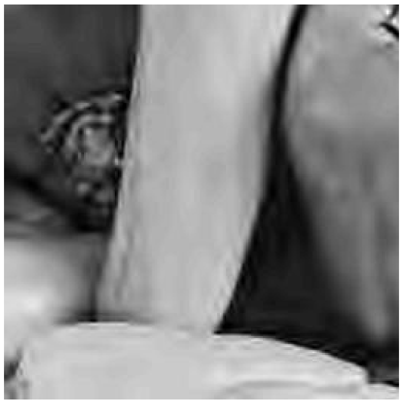

Fig. 3. *Portion of* Peppers *image coded using SFQ optimization. Rate = 0.07 bpp, PSNR = 29.08dB.*

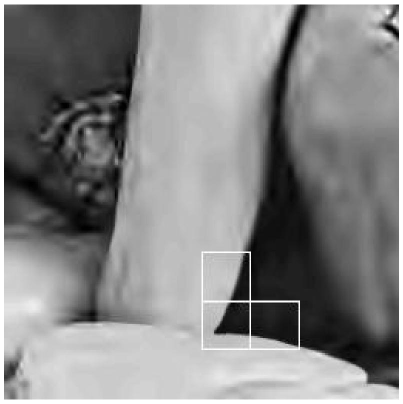

Fig. 4. *Portion of* Peppers *image coded using W-SFQ optimization. Rate = 0.07 bpp, PSNR = 29.25 dB. The three boxes highlight some of the blocks described by curve trees.*

Tests on other natural images perform similarly at a variety of bitrates; for images such as *Cameraman* with isolated, sharp edges, gains may be up to 0.30dB. For synthetic images with sharp edges and very light texture, we can achieve gains several dB above SFQ. For images such as *Lenna* that contain smoother edges with surrounding textures, our gains are currently around 0.05dB.

5. Conclusion

In this paper, we have introduced wedgelets as a tool for parsimonious description of geometry. We have explored some of the practical problems associated with using a geometric tool for natural image compression, and we have discussed W-SFQ as one technique to address these challenges. Despite our relatively modest geometric tool, we notice nontrivial compression gains when geometry is introduced to a powerful wavelet-based coder. This motivates the search for more fully-developed techniques for geometric image compression.

Many questions remain as to the optimal approach for geometric image compression. Even for simple probability models, the information-theoretic optimal coding scheme is often unknown. Several current topics of research may lead to breakthroughs in compression performance for natural images; the best solution may lie in some combination of better geometric tools, alternative compression frameworks, and new harmonic bases motivated by geometry.

Even for an approach based on an explicit geometric tool, several important questions remain. Our technique, for example, involves sending geometry, subtracting it off, and compressing the residual. More generally, perhaps there is another approach where geometric information is used more intelligently. Bandelets meet this description, but have yet to yield a practical compression scheme. Another question relates to how a coder should attempt to correct for geometric artifacts remaining from the geometric tool; it seems logical that geometric features should be represented either by the geometric tool, or not at all. Such questions are currently topics of investigation. [1]

6. References

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

[2] M. N. Do and M. Vetterli, "Contourlets: A directional multiresolution image representation," in *IEEE Int. Conf. on Image Proc. – ICIP '02*, Rochester, New York, Oct. 2002.

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

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

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

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

[7] M. B. Wakin, J. K. Romberg, H. Choi, and R. G. Baraniuk, "Image compression using an efficient edge cartoon + texture model," in *Proc., IEEE Data Compression Conference – DCC '02*, Snowbird, Utah, April 2002, pp. 43–52.

[8] J. Froment, "Image compression through level lines and wavelet packets," in *Wavelets in Signal and Image Analysis*, A. A. Petrosian and F. G. Meyer, Eds. Kluwer Academic, 2001.

[9] M. B. Wakin, J. K. Romberg, H. Choi, and R. G. Baraniuk, "Rate-distortion optimized image compression using wedgelets," in *IEEE Int. Conf. on Image Proc. – ICIP '02*, 2002.

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

[1]Thanks to Mike Orchard for many stimulating discussions.