Signal Processing: *Image Communication* 27 (2012) 140–152

Contents lists available at SciVerse ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

The chordlet transform with an application to shape compression ☆

Z. He [a], M. Bystrom [b,*]

[a] *InfraReDx Inc., Burlington, MA 01803, USA*
[b] *OneCodec Inc., Belmont, MA 02478, USA*

ARTICLE INFO

Article history:
Received 16 May 2011
Accepted 7 September 2011
Available online 8 October 2011

Keywords:
Beamlet
Chordlet
Contourlet
JBIG

ABSTRACT

Due to their abilities to succinctly capture features at different scales and directions, wavelet-based decomposition or representation methods have found wide use in image analysis, restoration, and compression. While there has been a drive to increase the representation ability of these methods via directional filters or elongated basis functions, they still have been focused on essentially piecewise linear representation of curves in images. We propose to extend the line-based dictionary of the beamlet framework to one that includes sets of arcs that are quantized in height. The proposed chordlet dictionary has elements that are constrained at their endpoints and limited in curvature by system rate or distortion constraints. This provides a more visually natural representation of curves in images and, furthermore, it is shown that for a class of images the chordlet representation is more efficient than the beamlet representation under tight distortion constraints. A data structure, the fat quadtree and an algorithm for determining an optimal chordlet representation of an image are proposed. Codecs have been implemented to illustrate applications to both lossy and lossless low bitrate compressions of binary edge images, and better rate or rate–distortion performance over the JBIG2 standard and a beamlet-based compression method are demonstrated.

1. Introduction

Concise and intuitive representations of data are the foundations of many image analysis and processing approaches, including image compression. Need for efficient transforms has led to the development of toolsets from global frequency-based approaches, such as Fourier analysis, to multiscale space–frequency approaches, such as wavelet analysis [1]. In the past few decades sophisticated methods for localized directional analysis of images have been developed; these include the steerable pyramid [2], the brushlet [3], the curvelet [4], the contourlet [5] and the beamlet [6]. These approaches have found successful application in such varied realms as image retrieval, compression, denoising, and inverse problems.

However, we note that for two-dimensional images, while these approaches may involve directionally filtered images or employ elongated basis functions, there is still essentially a piecewise linear representation of curves in images. A specific example of this class of approach is the beamlet framework of [6], which employs a line-segment-based dictionary. Any edge in an image can be represented to within a desired degree of distortion using a chain of dictionary elements, since elements down to pixel size can be employed. However, the efficiency in representation arises when some distortion is permitted, since the endpoints of the elements are constrained to predetermined locations so that not all possible line segments in an $N \times N$ image are contained in the dictionary.

In this work we propose to extend the line-segment-based beamlet dictionary to a chordlet dictionary that includes elements that are arcs. As with the lines in the

☆ This work was supported in part by NSF EEC-9986821 and was presented in part at EUSIPCO 2007.
* Corresponding author.
E-mail addresses: zhihua@ieee.org (Z. He), bystrom@ieee.org (M. Bystrom).

0923-5965/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2011.09.001

beamlet framework, the arcs' endpoints are constrained to pre-determined locations on the boundaries of squares formed by recursive dyadic partitioning of an $N \times N$ image. The arcs' curvatures are constrained by a quantization factor that is a function of distortion or rate constraints in the image processing application. The beamlet dictionary is then a subset of the new dictionary, and there is now the potential for a more visually natural representation of curves in images.

In the following we first provide an overview of the beamlet framework, which forms the motivation for the current work. In Section 3 we then propose extension of the beamlet dictionary to include arc elements and discuss construction of these elements. A primary contribution of this work is an analysis of the representation efficiency of the chordlet as compared with the beamlet for a class of images, which is given in Section 4. While the chordlet framework will never exceed beamlet performance on images composed primarily of straight or short edge segments, for classes of images consisting of primarily curved segments it is shown that the chordlet will outperform the beamlet.

A secondary contribution of this work is a new representation method for the chordlet framework that is a variant of the quadtree structure of the beamlet-based JBEAM [7] algorithm as discussed in Section 5. The new fat quadtree representation permits succinct description of multiple curves in an image at the same scale. To employ this structure a tree-pruning algorithm is introduced.

In Section 6 we review beamlet element selection using the JBEAM algorithm and, as a tertiary contribution, propose a multi-scale selection algorithm for the chordlet framework. Finally, to illustrate the chordlet performance over a certain class of images, we present an application to low-rate compression using the proposed chordlet framework. Significant rate and/or distortion savings over both JBEAM [7] and the JBIG2 standard [8] will be shown for some binary edge images.

2. Overview of the beamlet framework

The beamlet framework, as described in detail in [6], consists of: the beamlet dictionary, which is a set of line segments that will be used to approximate structures in images; a transform, which is the set of integrals of the image along the elements of the dictionary; the beamlet pyramid which is an organized collection of transform coefficients; the beamlet graph, which connects vertices representing pixel corners by edges representing beamlet dictionary elements; a collection of beamlet algorithms for data analysis via the beamlet transform and associated bookkeeping structures. As in [6] we proceed by giving an overview of the beamlet elements, transform, and pyramid structure. A discussion of a beamlet algorithm will be reserved for Section 7 and interested readers are referred to [6] for an overview of the beamlet graph.

Using the notation of [6] an array of $N \times N$ pixels can be modeled as a continuum square $[0,1]^2$, where each pixel is then a $1/N \times 1/N$ non-overlapping square of a grid in $[0,1]^2$. A key to the beamlet framework is that it allows for elegant multiscale decomposition. To achieve this, the image must be partitioned via, for instance, recursive dyadic partitioning, so that beamlet dictionary elements and the beamlet transform can be defined at different scales. For an image of size $N \times N$, assuming $N=2^J$ with J a positive integer, a dyadic square at scale $0 \le j \le J$, for j an integer, is the collection of the points $\{(x_1,x_2) : [k_1/2^j,(k_1+1)/2^j] \times [k_2/2^j,(k_2+1)/2^j]\}$, with $0 \le k_1, k_2 \le 2^j$. Therefore, $j=0$ represents the coarsest scale, while $j=J$ represents the finest scale or partitioning under consideration.

Since the goal of the beamlet transform is to represent image structures by combinations of elements from the beamlet dictionary, we then examine how the elements are defined. Consider a level of resolution $\gamma = 2^{-J-L}$ for integer $L \ge 0$. The boundary of each dyadic square S at level j is divided into equispaced partitions of length γ forming a total of $M_{j,\gamma}$ permissible vertices for beamlet elements at scale j [6]. Thus, the points on the boundary of the dyadic square at scale $j \in \{0, 1, \ldots J\}$ form the set $V_{S,j,\gamma} = \{v_{i,S,\gamma} : 0 \le i < M_{j,\gamma}\}$, where $M_{j,\gamma} = 2^{L+J-j+2}$. The collection of all vertices at scale j for resolution γ is then $V_{j,\gamma} = \bigcup_S V_{S,j,\gamma}$.

Beamlet elements are then formed by the lines that connect any pair of vertices in $V_{S,j,\gamma}$. This implies that the boundary partitioning constrains locations of beams, thus reducing the set of lines representable in an image given a fixed level of resolution. For resolution γ and scale j, the set of lines or *beams* formed by connecting distinct pairs of points in $V_{S,j,\gamma}$ for all S is then denoted as $B_{\gamma,j}$. The entire beamlet dictionary B_γ^J is then defined as the collection of all beams from $B_{\gamma,j}$; $0 \le j \le J$. For simplicity, the elements in beamlet dictionary B_γ^J are referred to as beamlets. Illustrations of the recursive dyadic partitioning of a unit square and the partitioning of the squares' boundaries via a fixed γ, along with selected elements in a beamlet dictionary at coarser and finer scales, are given in Fig. 1(a) and (b), respectively.

The beamlet transform is defined as the collection of line integrals over all beams to scale J taken of a continuous function defined on the unit square. Considering a continuous image $I(\cdot)$ formed from a discrete image $I[m,n]$ through interpolation [6], the beamlet transform of this image is given as

$$R_i(b_i) = \int_{b_i} I(x(l))\, dl, \quad b_i \in B_\gamma^J, \tag{1}$$

which is the integral over the line segment b_i over a unit-speed path, $x(l)$. Thus, the beamlet transform of an image I of size $N \times N$ ($N=2^J$), yields a set of beamlet coefficients

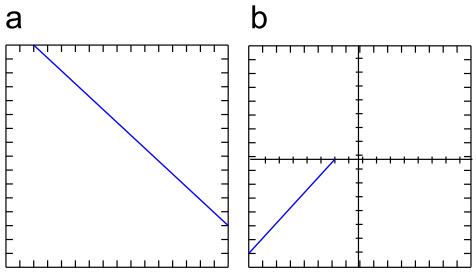

Fig. 1. Illustration of beamlet elements at two scales with fixed resolution, γ: (a) $j=0$ and (b) $j=1$.

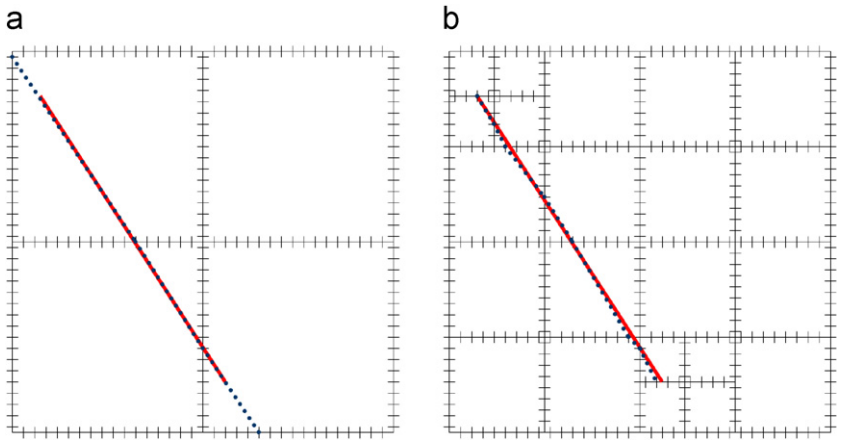

Fig. 2. An example of two potential representations of an edge (solid line) by selected beamlet elements (dotted lines) in a beamlet pyramid.

$\{R_i(b_i)\}$. For brevity, the coefficient associated with the ith beamlet, b_i, will be referred to as R_i.

Through a beamlet transform, an arbitrary curve in an $N \times N$ pixel image can be represented by a collection of beamlet elements at different scales in different dyadic squares. Thus, the beamlet representation effectively approximates curves in an image via a set of line elements constrained spatially by pre-determined endpoints. Fig. 2(a) provides an illustration of the representation of an image edge using three elements from the beamlet transform of the curve. Although three elements were used for this approximation, it is clear that the edge could be approximated with different degrees of quality using other combinations of beamlet elements. For instance, Fig. 2(b) illustrates the same curve approximated by seven elements at a combination of finer scales. We further note that similar quality to the three-element approximation of Fig. 2(a) could be achieved with a single element at the coarsest scale. As suggested in Fig. 2(b), perfect representation of curves in images is possible, thus allowing for lossless representation. This is more readily apparent when considering a discrete representation of the same image edge. If the elements in the beamlet dictionary are permitted to be one pixel in size, perfect reconstruction of image edges is obtainable, although at significant rate cost.

As described in [6], as $\gamma \to 0$, a beamlet at one scale "can be decomposed into the union of at most three beamlets at the next finer scale." Thus, beamlets can be organized in a pyramid form, with a beam at a coarse scale being the parent of some number of beams at finer scales. Although this infinitely fine partitioning of the boundary will certainly not be performed in practice for discritized images, it is still useful in beamlet algorithms to organize beamlets in a pyramidal fashion particularly when choosing the best representation of images from different possible scales.

3. Extension of the beamlet dictionary

The proposed extension to the beamlet dictionary is illustrated in Fig. 3(a) and (b); the addition of arc elements results in a dictionary that is a superset of the beamlet dictionary. Each beamlet element is then a chord subtending a set of arcs, with the quantized height of each arc, that is, the quantized distance between the midpoint of the chord and the midpoint of the arc, denoted by $k_i \in \{1, 1/\Delta, \ldots, K\}$; $K \le r$ where $2r$ is the chord length. The total number of extended-beamlet elements in the new dictionary is then dictated by $1/\Delta$, namely, the curve approximation that is acceptable. Thus, it is possible to choose any quality representation, and even a lossless representation, as with the beamlet. In the continuous case, for perfect reconstruction, $r \to 0$ and Δ is selected so that there is ϵ ($\epsilon \to 0$) difference between the heights of neighboring arcs. In the discrete case, for perfect reconstruction, an element can be of size as small as a single pixel. Naturally, representation or reconstruction quality comes with the penalty of compression rate. Following definition of the new dictionary an analysis of dictionary size and representation performance is given in Section 4.

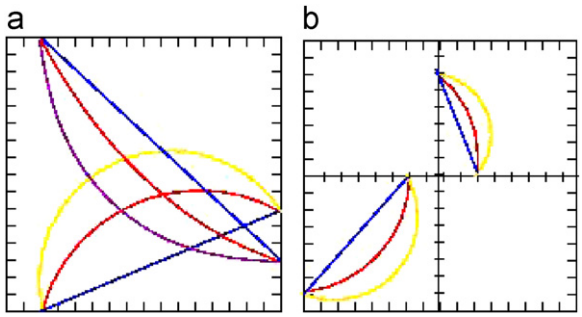

Fig. 3. Illustration of chordlet elements at two scales for selected arc heights and fixed resolution γ: (a) $j=0$ and (b) $j=1$.

The ith element in the extended beamlet or *chordlet* dictionary is denoted as c_i. Analogous to the beamlet transform of Eq. (1), the continuous chordlet transform is given as

$$R_i^+(c_i) = \int_{c_i} I(x(l))\, dl, \quad c_i \in C_\gamma^{J,\Delta}, \tag{2}$$

where $C_\gamma^{J,\Delta}$ is the set of elements in the extended-beamlet dictionary up to scale J, resolution γ, and pre-selected arc

height quantization factor, Δ. For brevity the chordlet coefficient for element c_i will be referred to as R_i^+.

To introduce the additional element dimension over the beamlet we first review characteristics of curves. Note that the distance adopted for this discussion is the Euclidean distance, although other metrics could easily be used in chordlet construction and transform computation.

Definition 1 (*Chord length 2r*). The chord length $2r$ of an arc Γ with two endpoints ρ_0 and ρ_1 is defined as the Euclidean distance between ρ_0 and ρ_1.

Definition 2 (*Arc height k*). For an arc Γ with two endpoints ρ_0 and ρ_1, the height, k, of the arc is defined as the distance between the mid-point of the chord connecting ρ_0 and ρ_1 and the mid-point of arc Γ.

Definition 3 (*SmoothCurve C_v*). A curve with chord length $2r$ and a constant curvature $\kappa < \frac{1}{r}$.

Definition 4 (*The radius R and angle θ of SmoothCurve C_v*). A *SmoothCurve* with curvature κ corresponds to an arc segment from a circle with radius $R = 1/\kappa$ and defining angle θ, the angle formed between the radius to one end of the arc and radius to the midpoint of the arc.

Definition 5 (*SmoothCurveSegment C_s*). A *SmoothCurveSegment*, C_s, with chord length $2r$ is a curve segment with constant curvature κ, whose two endpoints fall on the boundaries of the same dyadic partition.

For purposes of performance analysis, the set of chordlet elements used in Eq. (2) can now be described using these curve characteristics, namely,

$$C_\gamma^{J,\Delta} = \{C_s = \overline{v_{e_1,j}, v_{e_2,j}}, k : k = 0, 1/\Delta, \ldots, \lfloor r_{e_1,e_2} \rfloor, \\ 0 \le e_1, e_2 \le M_{j,\gamma},\ e_1 \ne e_2,\ 0 \le j \le J\}, \tag{3}$$

where C_s is a SmoothCurveSegment of arc height, k, connecting two points, $v_{e_1,j}$ and $v_{e_2,j}$, on the boundary of a dyadic square at scale j. The quantization step size, $1/\Delta$, can be selected for desired height resolution. The set of $M_{j,\gamma}$ element edge points is as for the beamlet framework and the chord $2r_{e_1,e_2}$ is restricted to these endpoints.

The continuous chordlet transform of Eq. (2) is translated for practical use to a discrete transform of image f as

$$R_i^+[c_i] = \sum_{m,n} f[m,n]\phi_{m,n}(c_i), \quad c_i \in C_\gamma^{J,\Delta}, \tag{4}$$

where γ is typically set to unity and i is the chordlet element index. The curve weight, $\phi_{m,n}$, is defined as

$$\phi_{m,n}(c) = \int_c \int_{x(l)} \delta_{m,n}\, dp\, dl, \tag{5}$$

where for the unit-size pixel

$$\delta_{m,n} = \begin{cases} 1, & l \in [m-0.5, n-0.5] \times [m+0.5, n+0.5], \\ 0 & \text{else.} \end{cases}$$

For applications to binary images $\phi_{m,n}(\cdot)$ can be quantized. In the results in Section 7 the mapping

$$\phi_{m,n}^q(c) = \begin{cases} 1, & \phi_{m,n}(c) \ge 0.5, \\ 0, & \phi_{m,n}(c) < 0.5 \end{cases} \tag{6}$$

is employed.

Given the discrete transform of Eq. (4), a coefficient R_i^+ captures the correlation between the ith element and a line or curve in the image, I, under consideration. To select which combination of elements at various scales best represent an image, we first quantify the contribution of a single element to the quality of the representation, $\hat{I}$, of the image. This contribution is termed the relative significance and is given as

$$S_i^+ = (R_i^+ - (l_i - R_i^+)) = 2R_i^+ - l_i, \tag{7}$$

where l_i is the arc length of element c_i, while the terms R_i^+ and $(l_i - R_i^+)$ correspond to the positive and negative contributions of the associated element, respectively.

However, since an image curve can be represented by combinations of one or more elements at the same scale or at different scales, additional characteristics are required. First, we define the contribution of a representation, $\hat{I}(\xi)$, where ξ is the set of transform elements for the given representation, to an image I as

$$C(\hat{I}(\xi)) = \sum_{m,n} (I - \max(I - \hat{I}(\xi), \underline{0}) + \min(I - \hat{I}(\xi), \underline{0})), \tag{8}$$

where $I - \max(I - \hat{I}(\xi), \underline{0})$ captures the positive contribution, $\min(I - \hat{I}(\xi), \underline{0})$ captures the negative contribution and $\underline{0}$ is a null matrix. Then, we define the gain provided by a single element to an existing representation. Denoting the representation using both ξ and an additional element c_i as $\hat{I}(\xi \cup c_i)$, the contribution gain of c_i relative to $\hat{I}(\xi)$ is given as

$$G_c^+(\hat{I}(\xi), c_i) = C(\hat{I}(\xi \cup c_i)) - C(\hat{I}(\xi)). \tag{9}$$

These contributions to image representation are used in the proposed algorithm of Section 6.2 to determine an optimal representation of images.

4. Chordlet performance analysis

While a beamlet element b_i is completely specified by its endpoints, a chordlet element, c_i, must also be specified by an additional parameter k_i. Considering a square of size $2^J \times 2^J$, a total of $2J+3$ bits is required for element specification using the beamlet-based JBEAM binary-image compression method [7]. On the other hand, binary representation of the arc height k_i for $\Delta = 1$ requires J bits, so that a total of $3J+3$ bits is needed to specify a chordlet element at the same scale for this height resolution. While it would appear that a chordlet representation is less efficient than a beamlet representation, it can be shown that for an arbitrary curve partitioned into segments of constant curvature with endpoints falling on dyadic square boundaries that the chordlet is a more efficient representation than the beamlet. Limiting analysis to this class of curves is restricting, but as shown below, as long as the distortion constraint is tight and the partitioning can be performed into dyadic squares of size $N > 4$ the

chordlet representation is more efficient. Furthermore, as will be demonstrated in Section 7, binary edge images can often be more efficiently represented by a chordlet-based rather than a beamlet-based scheme.

The focus of the evaluation of the efficiencies of beamlet and chordlet frameworks is to find the average numbers of elements required for the representation of an arbitrary curve using each of these transforms. To make the problem tractable, but yet meaningful, we restrict our attention to a subset of curves. The introduction of the *SmoothCurve* in Definition 3 yields a bridge between arbitrary curves and the chordlet elements. An arbitrary curve with varying curvatures can be partitioned into a chain of *SmoothCurves* with different curvatures. For analysis, the *SmoothCurves* in an image field are assumed to be uniformly distributed. We then approximate the image's *SmoothCurves* by a series of chordlet elements.

Lemma 1. *A SmoothCurve, C_v, with endpoints within the grid range $N \times N$, $N = 2^J$, can be approximated within Hausdorff distance $\frac{1}{2}$ for $\Delta = 1$ by a continuous chain of chordlet elements $c_i \in C_J^+$, where the number of chordlet elements required for approximation is bounded by $8 \log_2(N) = 8J$ for $N > 2$.*

The proof of this lemma is given in Appendix A. It corresponds to Lemma 2.2 in Donoho and Huo [6] with proof in [9] which can be restated as

Lemma 2. *Any line segment with endpoints within the grid range $N \times N$ can be approximated within Hausdorff distance $\frac{3}{2}$ by a continuous chain of beamlets $b_i \in B$, where the number of beamlets required is bounded by $8 \log_2(N)$ for $N > 2$.*

In the proofs of the above two lemmas, a *SmoothCurve* (correspondingly, line segment) is first broken into a continuous chain of *SmoothCurveSegments* (line segments) whose endpoint-pairs fall on the boundaries of common dyadic square; these C_s (line segments) can be then approximated by a set of chordlet (beamlet) elements. Thus, in the proof of Lemma 1, *SmoothCurve* approximation has been converted to the approximation of a *SmoothCurveSegment* C_s. In Appendix A it is shown that a C_s can be approximated by a single chordlet when the quantization of chordlet height is chosen properly to meet the approximation error requirement. A lower bound to the average number of beamlet elements required for approximation of a C_s at a fixed distortion is also given in Appendix A.

Table 1 gives results for the average number of beamlet elements, $\overline{N}_B$, required for representation of a *SmoothCurveSegment* for selected values of N and ϵ_0. For $N \leq 16$ and when the approximation error ϵ_0 is relatively large, only one beamlet is required for a *SmoothCurveSegment*. For 75% of the cases examined, Table 1 shows that $\overline{N}_B > 1.5$. Recalling the number of bits required to represent each beamlet or chordlet element, namely $2J+3$ versus $3J+3$ where $J = \log_2 N$, implies that for images with edges that can be divided into few or large smooth curve segments, the chordlet representation is significantly more efficient than the beamlet representation. Conversely, when there is fine texture in an image, as indicated by short curves or lines, or when distortion constraints are loose, the beamlet framework would be significantly more efficient than the chordlet, c.f. $N = 4$, $\epsilon_0 = 1.5$. As an example of where the beamlet would trivially outperform the chordlet, consider the canonical brick wall image. Since at a large scale the image is composed of straight lines outlining the bricks there is no need for the added curvature dimension. Indeed, using the chordlet would result in a significant loss in rate even if the distortion of reconstructed image were permitted to be high. Furthermore, the texture of each brick is fine, and at small grid sizes, e.g., 4×4, a beamlet can approximate a curve sufficiently well enough that a curvature dimension is extraneous. A rough rule of thumb for the selection of chordlet versus beamlet framework can be derived from the ratio of bits required for element representation, namely, $(3 \log_2 N + 3)/(2 \log_2 N + 3) > 1.3$, and the experimental result for the $N = 4$, $\epsilon_0 = 0.5$ case. For this parameter selection there are two curves for each line, and the number of beamlet elements required was found to be ≈ 1.4. This suggests that, since $1.4 > 1.3$, if there are twice as many curves as lines in the image, the chordlet framework is more efficient than the beamlet framework.

5. Beamlet and chordlet representation structure

Given a selected distortion metric, such as the Hausdorff distance or the number of mis-matched pixels between curves, and a fixed dictionary size, the goal of optimal or near-optimal image representation is to select the best combination of dictionary elements for an image, subject to rate, and/or distortion constraints. To do so a calculation of the distance, that is, distortion between each potential dictionary element and each edge in an image, is made. Thus, an efficient bookkeeping measure must be employed to track the candidate elements and assist with determining the best beamlet or chordlet representation for the image. A natural selection, arising from the recursive dyadic image partitioning strategy, is the quadtree; this was the method selected for the JBEAM algorithm [7]. To summarize the approach in [7], each node in the quadtree (QT) structure represents a single dyadic partition of the image and is

Table 1
Average number of beamlet elements required for approximation of a *SmoothCurveSegment* in an $N \times N$ grid with approximation error ϵ_0.

Error	$N=4$	$N=8$	$N=16$	$N=32$	$N=64$	$N=128$	$N=256$	$N=512$
$\epsilon_0 = 0.5$	1.4576	2.1656	2.9597	4.0738	5.6676	7.8803	10.9933	15.3837
$\epsilon_0 = 1.0$	1.0847	1.4902	2.2096	3.0325	4.1540	5.7224	7.9235	11.0278
$\epsilon_0 = 1.5$	1.0169	1.4248	1.8869	2.5665	3.4825	4.7545	6.5568	9.0906
$\epsilon_0 = 2.0$	1.0000	1.2549	1.6770	2.2572	3.0784	4.1871	5.7436	7.9411

labeled with information about the beamlet choice for its associated partition. A sample image with recursive dyadic partitioning is illustrated in Fig. 4(a). Each partition is labeled and is associated with a node in the quadtree, as illustrated in Fig. 4(b) and (c). The children of a particular node represent the dyadic partitioning of the coarse scale partition associated with that parent node. Using notation similar to that of [7] parent partitions are labeled with a "Q", child partitions that contain no image pixels are labeled with an "N" to denote empty, while the leaf nodes corresponding to partitions containing beamlets are labeled as "R" and have associated with them all necessary information for the associated beamlet element, namely, its vertices.

We also introduce a related bookkeeping structure, the fat quadtree (FQT), as illustrated in Fig. 5 based on the original image of Fig. 4(a). In this case, each large-scale image curve is associated with its own quadtree structure and labeling is performed as described previously. This permits the use of multiple elements in the same dyadic square, but results in a more complex representation than the QT structure, since more than one curve can be considered in each dyadic square at the same scale. While it requires a more sophisticated element selection method than the lone quadtree, but as will be shown in Section 7 it can often provide for better compression.

For applications such as compression, once a quadtree structure has been determined and populated for a particular image, it must be encoded to form a bitstream. In [7] the authors use a progressive coding algorithm based on that of [10,11] to map the *Q*, *N* and *R* symbols together with the binary vertex labels for each of the beamlet-associated leaf nodes to a symbol stream. At the end of each stream, an EOF marker denotes the end of the data. The key to the algorithm is that high-level nodes and the most significant bits of leaf nodes at high levels are mapped before lower-level nodes and lower-significance bits; the algorithm and an example can be found in [7].

Due to the introduction and adoption of the new fat quadtree structure, this algorithm must be adapted. Given an example fat quadtree structure with labels as in Table 2 a symbol-stream, which includes the bits, $Bs_i(k)$, for the ith chordlet-associated leaf node, is as follows:

$$Q,Q,EOB,Q,R,R,N,N,Q,N,N,Bs_1(1),Bs_1(2),Bs_1(3),$$
$$Bs_2(1),Bs_2(2),Bs_2(3),N,N,N,N,R,R,N,N,N,Bs_1(4),Bs_1(5),Bs_1(6),$$
$$Bs_2(4),Bs_2(5),Bs_2(6),Bs_3(1),Bs_3(2),Bs_3(3),Bs_4(1),Bs_4(2),Bs_4(3),$$
$$Bs_1(7),Bs_1(8),Bs_1(9),\ldots,Bs_4(3J-5),Bs_4(3J-4),Bs_4(3J-3),EOF.$$

To be consistent with JBEAM construction, the order of symbols is based on a raster scan. In this particular example, as can be seen from Fig. 5 and Table 2, all the leaf nodes are from the $J-1$ and $J-2$ levels; therefore, a total of $3(J-1)+3$ and $3(J-2)+3$ bits are required for the corresponding chordlet element. As compared to the EZW-like JBEAM encoding, three bits are represented in

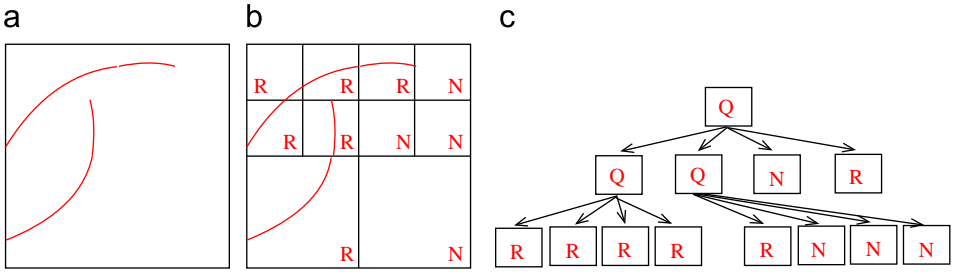

Fig. 4. Generation of a three-level labeled quadtree: (a) original image, (b) recursive dyadic partitioning and labeling, and (c) generated quadtree.

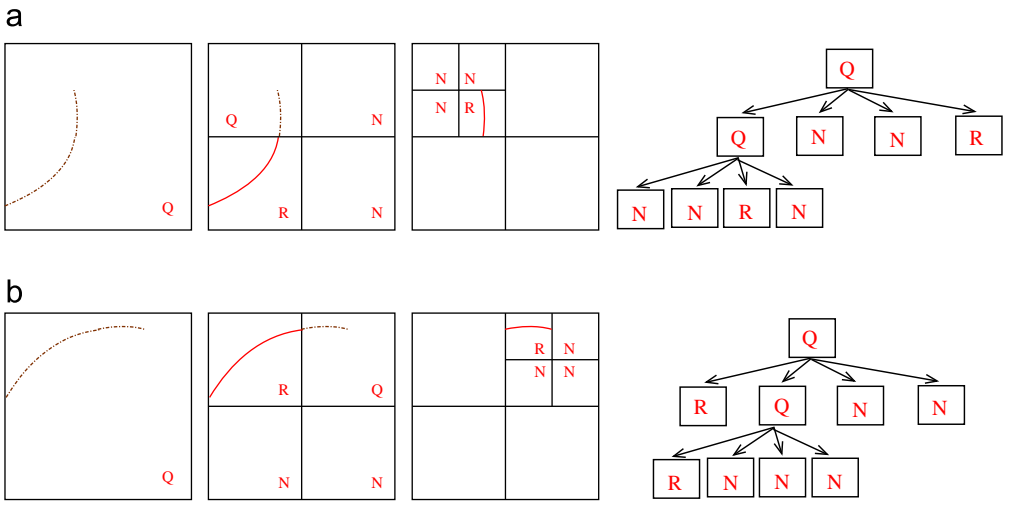

Fig. 5. Generation of a three-level labeled fat quadtree consisting of two sub-quadtrees: (a) partitioning and labeling of the first curve with associated quadtree and (b) partitioning and labeling of the second curve with associated quadtree.

Table 2
Fat quadtree structure with labels corresponding to the chordlet representation given in Fig. 5.

Level 1	Level 2		Level 3			
Q	Q	N	N	N	–	–
			N	R	–	–
	R	N	–	–	–	–
			–	–	–	–
Q	R	Q	–	–	R	N
			–	–	N	N
	N	N	–	–	–	–
			–	–	–	–

each level instead of two bits, due to the fact that the bit budget for each chordlet element has been increased from $2J+3$ to $3J+3$ when compared to a beamlet element.

The remainder of the adaptation is straightforward from [7] and thus is not discussed in detail here. However, we note that an additional marker, EOB, must be inserted at the end of the string of symbols for the coarsest scale in order to indicate the number of quadtrees forming the fat quadtree.

6. Element selection under rate or distortion constraints

Both the beamlet and chordlet transforms result in overcomplete representations of images. Note that while the curve in Fig. 2(a) is approximated by three beamlet elements, it could be approximated by either fewer or more elements depending on the desired approximation quality. Let $\hat{I}(\xi)$ denote the beamlet or chordlet approximation to an edge image, I, using the set, ξ, of weighted elements b_i or c_i with weights given by the transform coefficients R_i or R_i^+, respectively. Thus, a significant challenge is to pare down the entire set of coefficients for a succinct, yet sufficiently accurate representation of image edges given a distortion metric $D(I,\hat{I})$. Although the Hausdorff distance was used in the comparison of the different frameworks above, the distortion metric, $D(\cdot,\cdot)$, used in the algorithm below can be selected according to specific application. For the low-rate compression application discussed in Section 7 the distortion is the number of mis-matched pixels. We first review the JBEAM [7] algorithm and then present an algorithm for the FQT.

6.1. Overview JBEAM algorithm

For application of the beamlet transform to compression, the beamlets are first digitized as discussed in [7]. Following this step, it is necessary to determine which beamlets from the dictionary at selected, or all, scales best represent the image in a rate–distortion sense. For purposes of this discussion, we limit consideration to only binary edge images, and consider only a single set of image pixels, y, in an image partition. This method is readily extended to multiple curves and non-binary images.

The rate, $\mathcal{R}(\mathcal{P})$, for a particular beamlet quadtree or fat quadtree representation, $\mathcal{P}$, is given as the total number of bits required for each element-bearing R node in the tree plus $\log_2 3$ times the number of symbols in the stream, including the EOF and EOB symbols as described in the previous section.

At a particular scale, which yields a dyadic square S, the measure of the fit of a beamlet to an edge y that falls within this square is given by [7]

$$d(y,b,S) = d_f(y,b,S) + \lambda d_m(y,b,S).$$

The first term is a measure of the distance from the beamlet element, b, to the edge, y, while the second term is a measure of the mismatch between the beamlet element and the curve. The parameter λ is fixed for the particular application and trades off the influence of the two measures.[1] The best representation of the curve in square S is then given by

$$D(y,S) = \min_{b \in B_S^\gamma} d(y,b,S)$$

for a set of permissible beamlets, B_S^γ, in square S and, implicitly, for a selected λ and $\gamma = 1$. Note that there are multiple approaches to selecting the best representation for a given curve. First, all possible beamlet elements can comprise B_S^γ and decisions later made about the scale at which to best represent an image curve. Alternatively, for efficiency, the number of elements can be restricted prior to forming the set B_S^γ on the basis of the number of bits available. We consider the first option for optimality and assume that computational complexity is not a significant constraint in performing the analysis, although it would typically be a severe constraint in applications.

Thus, the challenge then becomes to solve the usual constrained optimization problem, namely,

$$\min_{\mathcal{P}} R(\mathcal{P}), \quad \text{subject to } \sum_{S \in \mathcal{P}} D(y,S) \le \mathcal{D},$$

where $\mathcal{D}$ is an overall distortion constraint. Clearly, the variables can be exchanged to minimize distortion in the face of rate constraints. In [7], this is solved via bottom-up pruning of the quadtree. To summarize the algorithm of [7], the rate and associated distortion are calculated for each tree node, assuming it is marked as first, Q, N, and then R, with the rates and distortions for each Q-node being the sum of rates and distortions for its children. These candidate rate and distortion calculations are stored as the tree is traversed. At each level, for a fixed Lagrange multiplier, the cost is calculated for each node and for each node labeling, noting that nodes labeled N have no distortion. A choice is made between the three labelings and the result is stored. This procedure is repeated for different values of the Lagrange multiplier and the appropriate resulting quadtree representation is chosen given the overall distortion constraint.

6.2. Proposed chordlet element selection algorithm

In this section we present an algorithm for selection of the best chordlet representation of an image under rate or distortion constraints using the fat quadtree structure. Since the FQT permits multiple curves to be represented at each scale, this algorithm must differ from that of Huo and Chen [7], but employs, in one of its steps, the bottom-up tree-pruning algorithm of [7], which was briefly reviewed above.

[1] However, it is shown in [7] for a selected example that the influence of λ is negligible.

We first note that in-scale inhibition, that is, element selection at a single scale, must be performed. This is an extension beyond that of [7]. More specifically, since a number of elements can be used to represent a curve or curves at a given scale, a set of the best representations with respect to a selected distortion measure must first be chosen for each curve and each scale. Consider an image with multiple curves at a given scale in a single partition. Since there are multiple curves in the partition, we have no single reference set of pixels in the partition from which to measure the distance to each element. Thus, we first pick the best representative element by selecting the chordlet element with the greatest coefficient. This then serves as a reference element. All other candidate elements at this scale and in this partition are tested in combination with this element. If the candidate returns a positive gain, that is, if it represents another curve in the partition, then it can potentially be added to the set. If the candidate's gains are negative with respect to the reference, then it is simply deemed to be too similar to the reference. More specifically, it is deemed a neighbor in chordlet space, for example, an element with identical endpoints and similar curvature to the reference, and thus is attempting to represent the same curve in the partition as the reference element. This process is iterated until all curves in the partition are represented sufficiently. The size of the initial set, and therefore the computational complexity of the resulting pruning algorithm, is controlled by a threshold.

Next, the best multi-scale representation can be selected out of the permitted single-scale representations. This results in curves that are represented by combinations of weighted chordlet dictionary elements potentially selected from among many scales to meet distortion or rate constraints. We first propose an in-scale selection algorithm and then discuss a multi-scale selection algorithm.

6.2.1. Element selection within a single scale

Assume a scale $j<J$, and consider a single curve in an image whose endpoints fall on or near the boundary of one of the dyadic squares at scale j. After employing the chordlet transform, a set of significant coefficients, S_i^+, can be extracted from the entire set $\{R_i^+(c_i)\}$; these elements can be thought of as sensing the curve and forming a cluster in the space of elements. As noted above, more than one chordlet element could be used to represent the curve; however, only the most significant of the elements, that is, those that alone or in combination with other would provide the best representation at scale j, should be preserved.

This selection between elements at a single scale is performed as follows:

1. Set $j=0$ and $n=2$.
2. Perform the discrete chordlet transform at scale j.
3. The elements with positive significance level, S_i^+, are put into a candidate stack, S_{cc}. If none of the elements has positive significance level, stop, increment j and return to Step 2. No cluster is formed at this scale. Otherwise proceed to Step 4.
4. The element with the highest significance level is determined to be the first cluster "center" θ_1; this constitutes the initial representation, $\hat{I}(\xi_1)$, where $\xi_1=\theta_1$. Remove θ_1 from S_{cc}.
5. For each iteration n, $n>2$, compute the contribution gain $G_c^+(\hat{I}(\xi_{n-1}\cup c_j))$ of each element $c_j\in S_{cc}$. The element which yields the largest contribution gain is declared the nth cluster center, θ_n. This new cluster center is combined with the previously-determined cluster centers to form the new set $\xi_n=\xi_{n-1}\cup\theta_n$. The cluster center, θ_n, and the elements with non-positive contribution gain are removed from S_{cc}.
6. Increment n and repeat Step 5 until S_{cc} is empty or a pre-determined number of cluster centers have been reached.
   The distortion $Dm(I,\hat{I}(\xi_i))$ between the image I and the chordlet representation $\hat{I}(\xi_i)$ using the current permissible number of chordlet cluster centers is determined as
$$Dm(I,\hat{I}(\xi_i))=D(\hat{I}(\xi_0),I)-\sum_{j=1}^{i}G^+(\hat{I}(\xi_j)), \tag{10}$$
   where $D(\hat{I}(\xi_0),I)$ is the distortion for a null representation $\hat{I}(\xi_0)$. For a given Λ compare the cost $L(\hat{I}(\xi_i))=R(\hat{I}(\xi_i))+\Lambda Dm(I,\hat{I}(\xi_i))$ with $L(\hat{I}(\xi_{i-1}))=R(\hat{I}(\xi_{i-1}))+\Lambda Dm(I,\hat{I}(\xi_{i-1}))$ where $R(\hat{I}(\xi_i))$ is the bit count associated with the representation $\hat{I}(\xi_i)$ for a particular coding system.
7. If $L(\hat{I}(\xi_i))\geq L(\hat{I}(\xi_{i-1}))$, stop the iteration. The chordlet elements $\{C_s^+(I)\}$ that correspond to cluster centers $\{\theta_k\}_{k=1}^{i-1}$ have survived and now provide the mono-scale representation of image I. If $L(\hat{I}(\xi_{i-1}))\geq L(\hat{I}(\xi_i))$, preserve the cluster centers $\{\theta_k\}_{k=1}^{i}$ and cost $L(\hat{I}(\xi_i))$, and perform the clustering assuming $i+1$ cluster centers in the chordlet space. Repeat from Step 3 until the iterative process meets the stop criterion.
8. Increment j and set $n=2$. If $j\leq J$ return to Step 2.

Using the above algorithm in-scale inhibition is performed through the determination of the cluster centers, which are the elements with high contribution gains, thereby restricting the use of multiple similar elements, that is, neighbors in element space, to represent the same edge. The rate–distortion problem is trivially solved by incrementing the number of permissible clusters and comparing the total rate–distortion cost assuming fixed Λ for the new and previous number of clusters. When either the rate constraint is exceeded or the cost increases, results are recorded and iteration is stopped. The algorithm is repeated for a new Λ thereby populating rate–distortion space.

6.2.2. Element selection across scales

We next consider the challenge of choosing the best representation across multiple scales. The proposed multi-scale fat quadtree-based representation algorithm for an image, I, of size $N\times N$ pixels, solves the constrained rate–distortion problem described above by generating a full FQT through associating parent and child representations of curves. Each set of preserved elements on each level of a particular quadtree in the FQT structure is then compared with the representation on a neighboring scale that corresponds to the same curve; those elements that yield the lowest rate–distortion costs are retained while

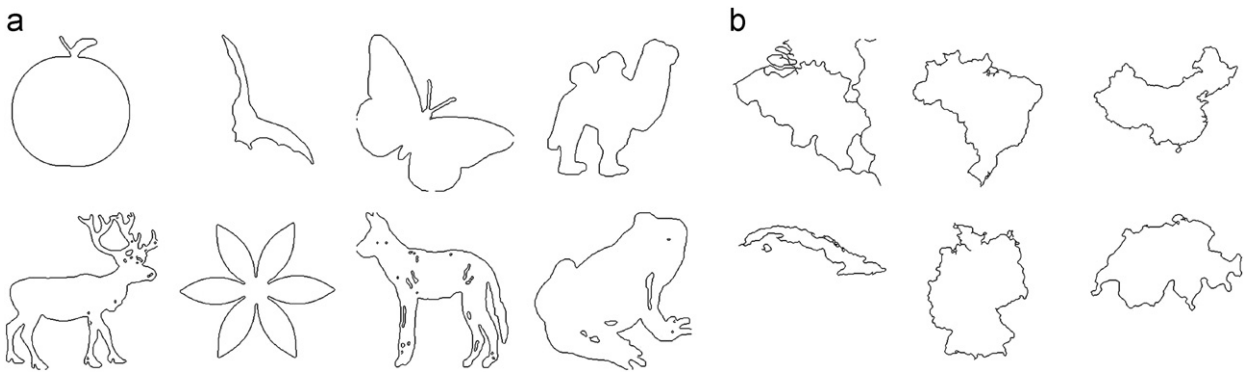

Fig. 6. Sample images used in the compression application, from top left to bottom right: (a) *Apple*, *Bat*, *Butterfly*, *Camel*, *Deer*, *Device*, *Dog*, and *Frog* (*Shape* images) and (b) *Belgium*, *Brazil*, *China*, *Cuba*, *Germany* and *Switzerland* (*Map* images).

the remainder is pruned from the FQT structure. The proposed cross-scale algorithm is as follows:

1. Starting at the finest scale $j=\log_2 N$, perform the transform, and determine the mono-scale representation for each 1×1 dyadic square, that is, each pixel, using a given Λ. The elements preserved as significant, the cluster "centers" from the single-scale selection process performed on four neighbors,[2] are added to the child cluster center stack S_c. Create leaf nodes labeled with either N or R according to the content of the associated dyadic square.
2. Perform mono-scale selection for scale $j-1$. The elements surviving this stage are put into a parent stack S_p. The associated curves corresponding to the preserved elements for each set of four neighboring child dyadic squares are determined and are put into the test stack S_t for later comparison of the best representation among levels.
3. Starting from $i=1$, for the ith element in the parent stack, $S_p(i)$, determine the clusters at scale j to which those child elements in S_t corresponding to $S_p(i)$ belong. The associated cluster centers at scale j are then preserved. The representation $\hat{I}^i_{j-1}$ for $S_p(i)$ is defined as the representation corresponding to the first test subtree T^i_{j-1} where the ith node is labeled with the cost of the ith cluster center. The second test representation $\check{I}^i_{j-1}$ is defined as the representation corresponding to subtree T^i_{j-1} where the node is labeled with the sum of the costs of the children elements corresponding to $S_p(i)$. The rate–distortion cost for representation $\hat{I}^i_{j-1}$ of the first test tree is computed as
$$\hat{L}^i_{j-1}=R(\hat{I}^i_{j-1})+\Lambda D_m(I,\hat{I}^i_{j-1}), \tag{11}$$
where $D_m(I,\hat{I}(\xi_i))$ is defined as in Eq. (10). Cost $\check{L}^i_{j-1}$ is similarly defined. Thus, representation cost $\hat{L}^i_{j-1}$ determines the cost for including $S_p(i)$ as a terminal node in the tree, while the cost $\check{L}^i_{j-1}$ captures the cost for choosing $S_p(i)$ as an intermediate node in the tree.

[2] Here the term neighbors denotes the four dyadic squares $m=1,\ldots,4$ that are children of a single parent.

Table 3
Number of non-zero image pixels and the bit count required for lossless coding of *Map* images using different representations.

	Belgium	*Brazil*	*China*	*Cuba*	*Germany*	*Switzerland*
Non-zero pixels	1233	822	813	658	871	845
JBIG2	7888	5768	5792	4360	5792	6088
JBEAM	6740	4584	4792	3392	5048	4512
QT	6432	4440	4624	3264	4744	4296
FQT	6428	4168	4592	3144	4368	4168

4. Using a bottom-up tree pruning algorithm as in [7] we determine the cost associated with each node in the FQT at scale $j-1$.
5. Repeat Steps 3 and 4 for each element in stack S_p, and remove the elements from S_c, which have been used in the process. If the resultant S_c is empty, go to Step 6. Otherwise, start from the next remaining candidate element in S_c, determine (as in Step 3) the associated subtrees at the current node scale and the child node scale, compute and compare their respective rate–distortion costs. If the cost for the candidate element is greater than the cost for the previously–computed element, prune the subtree of this candidate element, otherwise, push the corresponding element onto S_p.
6. If $j>1$, update S_c with S_p, decrement j, and repeat Steps 2 through 5, otherwise stop.

7. Applications to low-rate shape compression

In this section we present results of the application of the chordlet framework and the algorithm of the previous section to compression of binary shape images. Shape coding has received wide attention over the decades, ranging from basic chain coding to the more sophisticated block-based methods proposed for MPEG-4 such as reviewed in [12–15]. We note that there are similarities in between contour-based shape coding mode and our approach in that vertices and curve segments are selected; however, our method uses further constraints on endpoints to simplify the approach.

Table 4
Number of non-zero image pixels and the bit count required for lossless coding of *Shape* images using different representations.

	Apple	*Bat*	*Butterfly*	*Camel*	*Deer*	*Device*	*Dog*	*Frog*	*Guitar*
Non-zero pixels	658	557	968	957	1884	1228	1744	1221	776
JBEAM	3032	2912	4600	4906	9072	5888	8280	6080	3960
QT	2680	2936	4528	4816	8744	5800	7248	4920	4008
FQT	2680	2936	4424	3824	8316	5744	7248	5920	3952

Table 5
Lossy compression of *Map* images. Distortion is given as *D* and bit count is given as *R*.

Method	*Belgium*	*Brazil*	*China*	*Cuba*	*Germany*	*Switzerland*
JBEAM						
D	44	20	256	204	257	289
R	5984	4352	2792	2016	3056	2544
QT						
D	36	13	228	167	169	213
R	4632	4296	2208	1680	2824	2056
FQT						
D	28	11	213	157	211	179
R	5624	4248	2152	1360	2272	2016

We compare the performance of the chordlet-based framework to that of JBEAM [7,16], which is the inspiration for the proposed algorithm, as well as to the JBIG2 [8] industry standard. The binary shape images employed in this comparison are 256×256 images from two image databases: the *Map* database [7]; the MPEG-7 Core Experiment CE-Shape-1 part B *Shape* database [17]. Sample images from these databases are illustrated in Fig. 6.

The discrete chordlet transform with quantized elements, as described in Eq. (6) is employed, since the images under consideration are binary. Performance of the three algorithms will be compared on the basis of rate, *R*, and distortion, *D*. Here, rate is measured in the total number of bits required to represent the binary image, while distortion is defined as the number of mis-matched pixels in a representation $\hat{I}$ of image *I*.[3] For purposes of fair comparison, no entropy coding of the bit streams is employed.

We first consider lossless compression of the binary images. The first rows in Tables 3 and 4 give the number of non-zero pixels in each of the sample images and the remaining rows give the bitrates for compressing via a selection of methods, where QT and FQT represent the chordlet-based methods using the two representations. We observe that in all cases but three, *Bat*, *Frog* and *Guitar*, the FQT-based chordlet method has equal or better compression performance when compared with JBEAM and the QT-based chordlet method.

Lossy compression results are given in Tables 5 and 6 for sample *Map* and *Shape* images, respectively. While effort was made to hold the distortion constant, it is not possible to have equivalent distortion among the three approaches compared. However, we observe that in all cases, a chordlet-based method outperforms the beamlet-based method and, in general, the FQT chordlet representation outperforms the QT representation.

These results are further emphasized in Figs. 7 and 8. In Fig. 7(a) a portion of the original *Deer* image is shown. The corresponding portions after lossy compression and reconstruction via the JBEAM, QT-based-chordlet and FQT-based-chordlet approaches are shown in Fig. 7(b)–(d), respectively. In these latter figures, the dark pixels indicate mis-matched or missing pixels between the reconstructed and original images. Similarly, in Fig. 8 compression results using the three different schemes are demonstrated. Fig. 8(a) shows the original binary image *Apple*, while Fig. 8(b)–(d) shows the results of compression via the JBEAM, QT-based-chordlet and FQT-based-chordlet methods. The rate was deliberately kept very low in this example to emphasize the perceptually natural characteristics of the chordlet-based approach.

8. Conclusions

In this work we have proposed an arc-based extension to the beamlet element dictionary and have suggested methods of defining elements and performing a chordlet transform. It has been demonstrated that, for a particular class of images, the new transform and fat-quadtree representation is more efficient than the beamlet-based framework. Results were demonstrated for low-rate lossy and lossless compression of binary shape images with significant numbers of curved edges, and it is seen that, in general the FQT-based chordlet approach is the best approach, resulting in rate reductions of up to 31% and 22% over JBIG2 and JBEAM, respectively.

We note, however, that as demonstrated in Section 4 the chordlet framework will only be useful in compression when there is no fine texture. More specifically, if an image contains edges that are best represented under a given distortion constraint on small dyadic squares, then the beamlet-based framework will certainly outperform the chordlet-based framework. Furthermore, the chordlet framework's performance can naturally not meet that of beamlet's when images are composed primarily of non-curved edges. Nevertheless, while significantly more computationally complex than the beamlet framework, due to the number of dictionary elements that must be considered, the chordlet framework is promising for some compression applications. These applications include not only shape-compression for images but region-coding or

[3] Note that a variety of distortion metrics could be utilized in any particular application; see, e.g., [13] for an alternative. The metrics should be carefully selected to maximize either objective or subjective performance for the application, since different metrics could yield significantly different results.

Table 6
Lossy compression of *Shape* images. Distortion is given as D and bit count is given as R.

Method	*Apple*	*Bat*	*Butterfly*	*Camel*	*Deer*	*Device*	*Dog*	*Frog*	*Guitar*
JBEAM									
D	99	92	125	130	182	206	197	26	106
R	1808	1704	3056	3096	7008	3160	6096	5576	2552
QT									
D	88	80	120	130	169	199	185	31	98
R	1000	1608	2696	2752	6424	2904	5480	5288	2392
FQT									
D	92	83	103	118	150	167	141	26	104
R	968	1528	2856	2832	6184	2720	5520	5256	2256

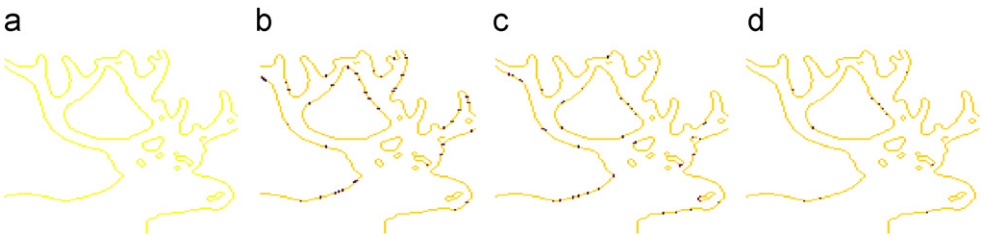

Fig. 7. An illustration of compression of a portion of image *Deer*. Dark pixels indicate distortion contributions, the mismatch between the original and reconstructed image. (a) Original image, (b) JBEAM representation, $D=182$ and $R=7008$, (c) quadtree-based chordlet representation, $D=169$ and $R=6424$, and (d) fat quadtree-based chordlet representation, $D=150$ and $R=6184$.

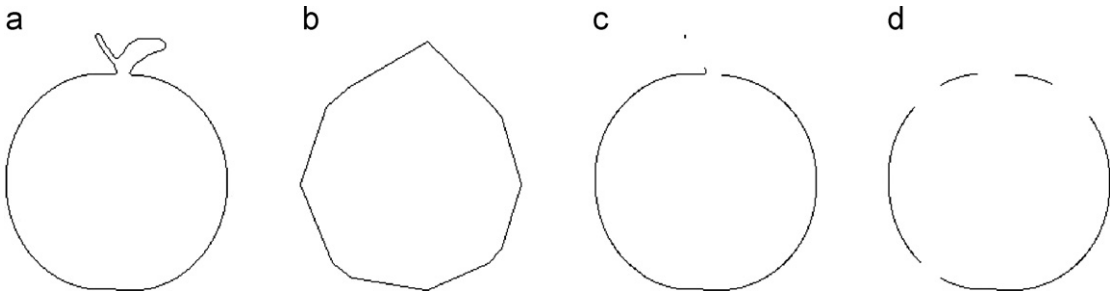

Fig. 8. An illustration of extremely low-rate representation of shape image *Apple* using different representation schemes, JBEAM and the two proposed methods. The rate, R, is the number of bits required to represent the image using the associated scheme: (a) original, (b) JBEAM, $R=312$, (c) QT, $R=312$ and (d) FQT, $R=184$.

contour-encoding applications in video and images, e.g., [18–20]. In practice, it would be useful to be able to switch between the beamlet and chordlet representations using a context-dependent indicator in the header.

Efforts continue in developing fast chordlet transform methods and expansion to additional applications such as image and video retrieval. The chordlet framework has the potential to meet most of the desired properties of shape descriptors as listed in [21]: ability to capture important shape characteristics, while reflecting properties of the human visual system; robustness to partial occlusion; robustness to rotation, scaling and zooming. While additional efforts will have to be dedicated to making this framework adaptable to non-rigid motion and non-affine transformations, there is the potential for use in non-annotated retrieval applications.

Another potential extension of this work is to denoising and inverse problems, extending the approaches of [6] using the curve-based framework. There are two concerns with using the chordlet framework. The first is computational complexity and the second is stability. Both these arise from the addition of the curve dimension, particularly with a requirement for fine granularity of curves as would be required for small structures. We anticipate this proposed framework would only find use in problems involving large-scale structures for the same reasons it is only beneficial over the beamlet for coarse structures or textures.

Appendix A

A.1. Proof of Lemma 1

The proof of Lemma 1 follows similar procedure as the proof of Lemma 2.2 of [9] which employs proofs of [9]. Assume a *SmoothCurve*, C_ν, with two endpoints ρ_0 and ρ_1, in an $N\times N$ image grid. There are two cases to consider based on those two endpoints. First, when the two endpoints fall

on the boundary of a single dyadic square, since the curve heights of the chordlet elements are quantized with step-size $\Delta = 1$, the maximum Hausdorff distance between C_v and its nearest chordlet element is bounded by $\epsilon_0 \frac{1}{2}$ and only one chordlet element is required for the approximation of C_v. This is proved in Section A.2.

For the second case, the two endpoints fall on the boundaries of two separate dyadic squares S_0 and S_1. Using Lemma 5.1 in [9], the coarsest recursive dyadic partition containing S_0 and S_1 that partitions the *SmoothCurve* into a chain of *SmoothCurveSegments* can be constructed. The number of dyadic squares in this RDP is bounded by $4 \log_2 N + 1$. From the definition of a *SmoothCurve*, the curvature κ of the C_v with chord length $2r$ is $\kappa \le \frac{1}{r}$. Thus, for each *SmoothCurveSegment*, C_s, with length $2r_s$ in the partition, the arc height is $k \le r_s$. By the definition and construction of chordlet dictionary as in Eq. (3), each C_s is within the chordlet approximation range, therefore it can be approximated within the Hausdorff distance $\epsilon_0 = \frac{1}{2}$ as shown below. Finally, the count of the number of *SmoothCurveSegments* for S_0 and S_1 yields at most $2(4 \log_2 N + 1)$ and refining this count for $N > 2$ as in [9] yields the bound of $8 \log_2 N$.

A.2. C_s approximation by chordlet elements

Given two endpoints (ρ_0, ρ_1) separated by distance $2r$, circles whose centers are of distance R; $R \ge r$ from both endpoints define the curve space for a *SmoothCurveSegment* with the same two endpoints. The set of chordlet elements $\{c_i\}$ with the same two endpoints offers a quantized/discrete curve space for corresponding *SmoothCurveSegment*. If the Hausdorff distance is adopted as the approximation error $\epsilon_0(c1,c2)$ between two curves $c1$ and $c2$, it is given as

$$\epsilon_0(c1,c2) = \max\left\{\max_{a\in c1}\left\{\min_{b\in c2}\{D(a,b)\}\right\}, \max_{b\in c2}\left\{\min_{a\in c1}\{D(a,b)\}\right\}\right\}, \tag{12}$$

where $D(a, b)$ can be the Euclidean distance between two points, a and b. If the height quantization is assumed to be $\Delta = 1$ then the maximum approximation error is by definition $\epsilon_0 \le \frac{1}{2}$.

A.3. C_s approximation by beamlet elements

The best piecewise linear approximation of a *SmoothCurveSegment* is a uniform partitioning of the curve into sub-curves with each sub-curve approximated by a single line segment as follows: given a curve with chord length $2r$ and height k, divide the curve into m equally spaced sub-curves, and find the smallest m such that the height z of each sub-curve satisfies $z \le \epsilon_0$, the approximation error. Notation is illustrated in Fig. 9, where an arc which is a section of a circle with radius R is uniformly partitioned into m sub-curves, and 2θ and 2β are the angles of the arc and a sub-curve, respectively. Given

$$R^2 = (R-k)^2 + r^2 \Rightarrow R = (k^2+r^2)/(2k)$$

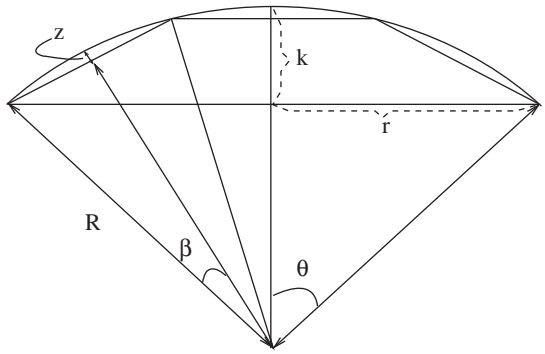

Fig. 9. An illustration of the approximation of a curve by line segments.

then

$$\theta = \arcsin\frac{r}{R} = \arcsin\frac{2kr}{k^2+r^2} \quad \text{while } \beta = \frac{\theta}{m}.$$

The distortion between the sub-curve and the line element is then

$$z = R - R\cos\beta = \frac{k^2+r^2}{2k}\left(1-\cos\frac{\theta}{m}\right).$$

To meet the distortion constraint $z \le \epsilon_0$ choose

$$m(r,k,\epsilon_0) = \left\lceil \frac{\arcsin\frac{2kr}{k^2+r^2}}{\arccos\left(1-\frac{2k\epsilon_0}{k^2+r^2}\right)} \right\rceil.$$

Thus for a desired approximation error ϵ_0, $m(r, k, \epsilon_0)$ is the minimum number of line segments required for approximation of a *SmoothCurveSegment*, C_s. We observe that when $m(r, k, \epsilon_0) > 1$, the endpoints of sub-curves might not fall on the boundary of a dyadic square. Therefore, additional beamlets are required for curve approximation and the above $m(r, k, \epsilon_0)$ serves as a lower bound on the total number of beamlet elements required for approximation. Then the average number of beamlet elements required for representing a *SmoothCurveSegment* is given as

$$\overline{N_B}(\epsilon_0) = \frac{4\sum_{G_1(r,k)} m(r,k,\epsilon_0) + 3\sum_{G_2(r,k)} m(r,k,\epsilon_0) + 2\sum_{G_3(r,k)} m(r,k,\epsilon_0)}{4\sum_{G_1(r,kx)} 1 + 3\sum_{G_2(r,k)} 1 + 2\sum_{G_3(r,k)} 1}, \tag{13}$$

where $G_1(r,k)$, $G_2(r,k)$, and $G_3(r,k)$ denote *SmoothCurveSegments* with co-edge, neighbor-edge, and opposite-edge endpoints, respectively. These are found through counting arguments assuming resolution $\gamma = 1$ as

$$\sum_{G_1(r,k)} m(r,k,\epsilon_0) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\sum_{k=0}^{\lfloor (i-j)/2 \rfloor} m((i-j)/2,k,\epsilon_0),$$

$$\sum_{G_1(r,k)} 1 = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\sum_{k=0}^{\lfloor (i-j)/2 \rfloor} 1,$$

$$\sum_{G_2(r,k)} m(r,k,\epsilon_0) = \sum_{i=1}^{N-1}\sum_{j=1}^{N-1}\sum_{k=0}^{\lfloor \sqrt{(1-j)^2+(i-N)^2}/2 \rfloor} \times m\left(\sqrt{(1-j)^2+(i-N)^2}/2, k, \epsilon_0\right),$$

$$\sum_{G_2(r,k)} 1 = \sum_{i=1}^{N-1}\sum_{j=1}^{N-1}\sum_{k=0}^{\lfloor\sqrt{(1-j)^2+(i-N)^2}/2\rfloor} 1,$$

$$\sum_{G_3(r,k)} m(r,k,\epsilon_0) = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=0}^{\lfloor\sqrt{(1-N)^2+(i-j)^2}/2\rfloor} \times m\left(\sqrt{(1-N)^2+(i-j)^2}/2,k,\epsilon_0\right) - N$$

and

$$\sum_{G_3(r,k)} 1 = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=0}^{\lfloor\sqrt{(1-N)^2+(i-j)^2}/2\rfloor} 1 - N.$$

References

[1] S. Mallat, A Wavelet Tour of Signal Processing, Academic Press, 1999.

[2] E.P. Simoncelli, W.T. Freeman, E.H. Adelson, D.J. Heeger, Shiftable multiscale transform, IEEE Trans. Inf. Theory 38 (March) (1992) 587–607.

[3] F.G. Meyer, R.R. Coifman, Brushlets: a tool for directional image analysis and image compression, Appl. Comput. Harmonic Anal. 4 (1997) 147–187.

[4] E.J. Candes, D.L. Donoho, Curvelets—a surprisingly effective non-adaptive representation for objects with edges, in: Curves and Surfaces, Venderbilt University Press, 1999.

[5] M.N. Do, Directional Multiresolution Image Representations, Ph.D. Dissertation, Swiss Federal Institute of Technology, Lausanne, Switzerland, November 2001.

[6] D.L. Donoho, X. Huo, Beamlets and multiscale image analysis, in: Multiscale and Multiresolution Methods, Springer Lecture Notes in Computational Science and Engineering, vol. 20, 2002, pp. 149–196.

[7] X. Huo, J. Chen, JBEAM: multiscale curve coding via beamlets, IEEE Trans. Image Process. 14 (2005) 1665–1677.

[8] P.G. Howard, F. Kossentini, B. Martins, S. Forchhammer, W.J. Rucklidge, The emerging JBIG2 standard, IEEE Trans. Circuits Syst. Video Technol. 8 (November) (1998) 838–848.

[9] D. Donoho, Wedgelet: nearly minimax estimation of edges, Ann. Stat. 27 (3) (1999) 859–897.

[10] J.M. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, IEEE Trans. Signal Process. 41 (December) (1993) 3445–3462.

[11] A. Said, W.A. Pearlman, A new, fast and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circuits Syst. Video Technol. 6 (June) (1996) 243–250.

[12] A.K. Katsaggelos, L.P. Kondi, F.W. Meier, J. Ostermann, G.M. Schuster, MPEG-4 and rate–distortion-based shape-coding techniques, Proc. IEEE 86 (4, June) (1998) 1126–1154.

[13] N. Brady, MPEG-4 standardized methods for the compression of arbitrarily shaped video objects, IEEE Trans. Circuits Syst. Video Technol. 9 (8, December) (1999) 1170–1189.

[14] J.-B. Lee, J.-S. Cho, A. Eleftheriadis, Optimal buffered compression and coding mode selection for MPEG-4 shape, IEEE Trans. Image Process. 10 (5, May) (2001) 686–700.

[15] Z. Chen, K.N. Ngan, A unified approach of bit-rate control for binary and gray-level shape sequences coding, IEEE Trans. Circuits Syst. Video Technol. 17 (7, July) (2007) 823–832.

[16] JBEAM Software, 〈http://www.isye.gatech.edu/xiaoming/jbeam 2004〉, 2004.

[17] L.J. Latecki, R. Lakamper, U. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in: IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, June 2000, pp. 424–429.

[18] H. Nicolas, D. Pateux, D. Le Guen, Minimum description length criterion for region-based video compression, in: Proceedings of the IEEE International Conference on Image Processing, 1997, pp. 346–349.

[19] A.J. Pinho, A region-based approach to high bit-rate video coding, in: Proceedings of the 11th Portuguese Conference on Pattern Recognition, 2000, pp. 301–306.

[20] V. Makkapati, P. Mahapatra, Contour encoding based on extraction of key points using wavelet transform, in: International Conference on Pattern Recognition, 2006.

[21] M. Ghanbari, Standard Codecs: Image Compression to Advanced Video Coding, IEE, 2003.