JBEAM: Coding Lines and Curves via Digital Beamlets

Xiaoming Huo[*], Jihong Chen
School of ISyE, 765 Ferst Dr.
Georgia Institute of Technology
Atlanta, GA 30332-0205

David L. Donoho
Department of Statistics
Stanford University
Stanford, CA 94305

Abstract

Beamlets are combined with zero-tree coding algorithm to create a new coding method, particularly suitable for lines and curves. Beamlets are a multiscale collection of line segments at a range of scales, location, having a variety of lengths and orientations. The new coding scheme—named JBEAM—is more efficient—in terms of bit rates—in coding binary curvy images in simulations.

Shape coding is an important technique in multimedia signal processing. A new approach based on combining beamlets—a hierarchical data structure for lines—and zero-tree coding is proposed. In simulations, the new method is proven to outperform existing methods implemented in industry standards. Our method is carefully designed to optimize the rate-distortion curve.

There are existing literatures on both multiscale (or hierarchical) representation of shapes, and coding by using line segments (or polynomial curvy segments). Some examples are Strip Trees, and Chain Coding. Due to space, we can not review them fully. The novelty of our work is the combination of the zero-tree coding with a multiscale representation. Moreover, an optimal coding for curves via hierarchical representations based on a rate-distortion sense has not been reported in the literature.

1 Coding a Single Digital Beamlet

We present the definition, representation and coding of a single digital beamlet.

Continuous Beamlet. Beamlets are introduced as a tool to analyze linear objects in 2-D [1]. In summary, beamlets are multiscale line segments; compared to the original set of all possible line segments, the dictionary of beamlets has low order of

[*]Corresponding author, 404 385 0354 (office), 404 894 2301 (Fax), xiaoming@isye.gatech.edu.
This work has been partially supported by National Science Foundation grants DMS 00-77261, DMS 01-40587 and DMS 01-40698.

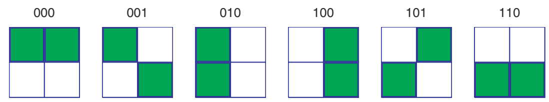

Figure 1: Six possible beamlet in a 2 by 2 image. The coded bits are in the titles.

complexity; and for an arbitrary line segment, it takes a small number of beamlets to approximately superpose the line segment within a given precision. All beamlets form a pyramid. In a Ph.D thesis [4], a first version of beamlet is presented, together with some of its application in image analysis.

Digital Beamlet. In a given dyadic square, a discrete beamlet is determined by its two end pixels, which lie on the boundary of this square. Given the two end pixels, the discrete beamlet is determined by interpolation. Without loss of generality, we assume that (x_1, y_1) and (x_2, y_2) are the indices of two end pixels, $0 < x_1, x_2 \le n, 0 < y_1, y_2 \le n$, where the size of the image is n by n. A digital beamlet is a set of pixels that intersect with the line segments connecting points $(x_1 - 1/2, y_1 - 1/2)$ and $(x_2 - 1/2, y_2 - 1/2)$. Bresemham's line drawing algorithm will produce the same set of pixels.

Counting Possible Configurations for a Single Beamlet. We count the number of possible positions of a digital beamlet. This will give an upper bound of the number of bits that are required for coding a single beamlet. A digital beamlet is determined by its two end pixels. Consider a 2^s by 2^s image. The number of boundary pixels is $4 \cdot (2^s - 1) = 2^{s+2} - 4$. Hence the number of beamlets is

$$\#(\text{beamlets}) = \binom{2^{s+2} - 4}{2} = (2^{s+1} - 2)(2^{s+2} - 5).$$

The number of bits that are necessary to code the above configurations is

$$\#(\text{bits}) = \log_2(\#\text{beamlets}) \le 1 + s + (s + 2) = 2s + 3. \tag{1.1}$$

The upper bound given in (1.1) gives an fairly good estimate. However, when the size of the square (2^s) is small, the number of bits can be significantly reduced. This prompts us to treat the beamlet coding for small squares differently than for large squares.

A Generic Coding Scheme. Given a 2^s by 2^s image, recall that we would like to code all the possible beamlets progressively. Let us first consider some simple cases. For a 1 by 1 image, it is trivial. For a 2 by 2 image, there are four pixels, all of which are boundary pixels. Hence there are six possible beamlets, which are depicted in Figure 1.

It is more interesting to consider the case when $s \ge 2$. Note that when the size of an image is 2^s by 2^s, there are $2^{s+2} - 4$ boundary pixels, which should be coded by at most $(2s + 3) = 2(s + 2) - 1$ bits. The following lemma is a stronger result.

Lemma 1 *Consider a length-*$(2^K - 1)$ *consecutive sequence,* $1, 2, 3, \ldots, 2^K - 1$, *in which there is a substring,*

$$s(i, j) = i, i + 1, \ldots, j - 1, j,$$

Proceedings of the Data Compression Conference (DCC'04)
1068-0314/04 $ 20.00 © 2004 IEEE

(a) (b)

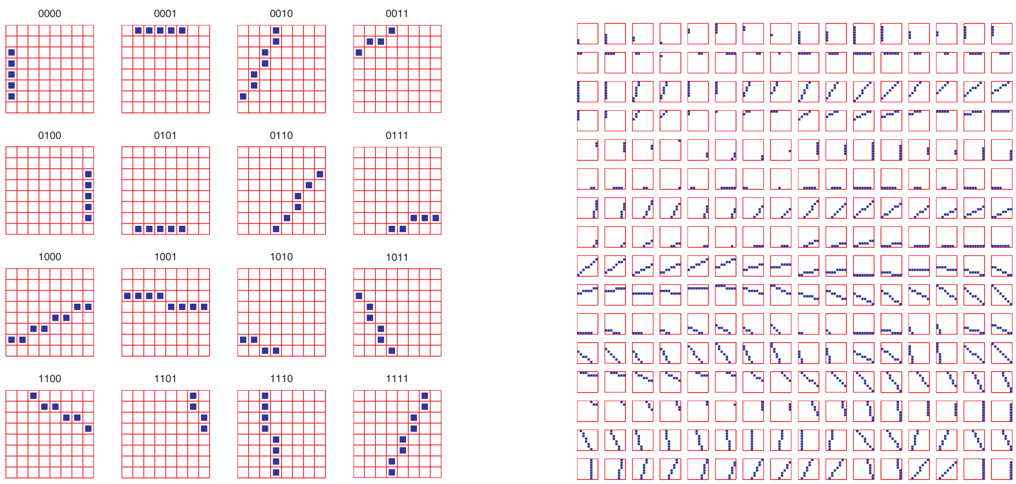

Figure 2: Representable digital beamlets in a 8 by 8 image when four (resp. eight) bits are available.

where $1 \leq i < j \leq 2^K - 1$*. There is a coding scheme for all substrings* $s(i,j)$ *such that (1) the coder is progressive and (2) it takes at most* $(2K-1)$ *bits.*

Note that if we align all the boundary pixels in a straight line, the two end points of a digital beamlet determines a substring; hence we obtain the above coding problem with a shorter sequence. The proof of the above lemma is constructive. Due to space, the proof is skipped.

Progressive Representation of a Single Beamlet. In the above coding scheme, when more bits com in, the position of a beamlet can be approximated better. To illustrate this phenomenon, we present the representable beamlets for a 8 by 8 image while 4 and 8 bits are available: Figure 2.

A *representable beamlet* is picked as follows. Consider there is only one bit available. Suppose the upper right pixel is chosen as the *central pixel* of the coding. The first bit will tell if the beamlet 'covers' this pixel or not. If not (the first bit is '0'), a representative beamlet goes from the central pixel to a last pixel. If yes (the first bit is '1'), a representative beamlet goes from the middle of the first half to the middle of the second half.

When more bits are coming in, the representation of the beamlet becomes more accurate. When there are 8 bits, the representative beamlets are illustrated in Figure 2 (b). Note that 9 bits is sufficient to code all digital beamlets in an 8 by 8 image. Hence the beamlets in Figure 2 (b) can approximate arbitrary beamlets with high accuracy.

2 Distortion

Given a binary image in a square, a distortion function is introduced. Since the distortion will be used to assess the goodness-of-fit of a beamlet representation, which will be described in the next section, some additivity condition must be satisfied. Moreover, since curve representation is considered, a small distortion should lead to better curve fitting by visual inspection. It will be shown that the proposed distortion function satisfies these requirements.

Distortion Function. Let S denote a square, and y denote a subset of pixels in the square S. Implicitly, y also represents the set of pixels that 'determine' the image content. Let $\mathcal{B}_S$ denote all the digital beamlets residing on square S: the beamlets with end-pixels on the boundary of square S. Let b represent a digital beamlet, which equivalently represents the set of pixels that this digital beamlet traverses. The distortion between the image y and a beamlet $b \in \mathcal{B}_S$ is defined as:

$$d(y, b) = L_1(y, b) + \lambda \cdot L_2(y, b), \tag{2.2}$$

where λ is a prescribed positive constant, and

$$L_1(y, b) = \sum_{x \in y} \rho_1^2(x, b), \tag{2.3}$$

where $\rho_1^2(x, b)$ denote the distance between a pixel x and a digital beamlet b; and

$$L_2(y, b) = \sum_{x \in y} I\{x \notin b\} + \sum_{x \in b} I\{x \notin y\}, \tag{2.4}$$

where $I\{\cdot\}$ is an indicator function. Some discussions on the above proposed distortion function are skipped.

Sub-additivity. In beamlet approximation to a binary image, when one moves from a fine scale to a coarse scale, the minimum possible distortion should be non-decreasing, because during this procedure, more restrictions are imposed on the beamlet representation. To be more specific, let us define the minimum possible distortion of a beamlet fit within a given square S as

$$D(y, S) = \min_{b \in \mathcal{B}_S} d(y \cap S, b), \tag{2.5}$$

where y is a subset of pixels in a binary image, and $\mathcal{B}_S$ denotes the set of beamlets in square S. Suppose square S can be equally partitioned into 2 by 2 subsquares:

$$S = S_1 \cup S_2 \cup S_3 \cup S_4. \tag{2.6}$$

A 'reasonable' distortion function should render:

$$D(y, S) \geq \sum_{i=1}^{4} D(y, S_i). \tag{2.7}$$

Using the distortion function that is defined previously, one can prove the following result.

Lemma 2 *Given the distortion function that is defined in (2.2), for the minimum distortion (which is defined in (2.5)) and an equal partition of a square S (which is described in (2.6)), the following inequality is true:*

$$D(y, S) + \varepsilon(S) \geq \sum_{i=1}^{4} D(y, S_i), \tag{2.8}$$

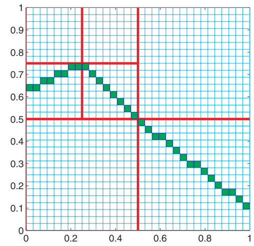

Figure 3: An example of binary image. Note this image is not a shape. It is made for illustrating the idea of beamlet representation.

where $\varepsilon(S)$ is a constant that only depends on the size of square S. In most of the cases, the inequality in (2.7) will be observed.

Due to the space, the proof is omitted. Quantity $\varepsilon(S)$ is a constant that only depends on the size of square S.

3 Beamlet Representation.

Given a upper limit of distortion, a *beamlet representation* of a binary image can be efficiently computed. More details will be presented in the section of Rate-Distortion-Optimized Representation.

As an example, we consider a binary image in Figure 3. At the coarsest level, no beamlet can fit well; hence a quadratic split is assigned. At the next scale, two boxes contains no dark pixels. They are labeled as 'No Split'. One box contains a beamlet. The last remaining box has no appropriate fitting; hence it is split into four. In the end, the four smallest boxes either contains no dark pixels or are fitted by beamlets. The corresponding partitioning and the vertices of beamlets are given as part (a) of the following table:

(a)

level 0	level 1		level 2			
			N	N	–	–
Q	Q	N	B_2	B_3	–	–
	N	B_1	–	–	–	–
			–	–	–	–

(b)

	1st vertex	2nd vertex	binary represent
B_1	16	45	10011111011
B_2	5	16	100100000
B_3	8	24	101101010

(3.9)

where Q, N, and B denote respectively 'Quadratic Split', 'No Split', and 'Beamlet'. Symbol "–" means no content. The coordinates of beamlets and their corresponding bit representations are in the part (b) of the above table. Note that from the above two tables, we can reconstruct a binary image.

It is possible that the beamlets in a beamlet representation are *not* connected. Such a feature has both advantage and disadvantage. Due to space, some details are omitted here. The alternate procedure that can address the disadvantage is not the theme of this paper, hence will be studied and reported elsewhere.

Proceedings of the Data Compression Conference (DCC'04)
1068-0314/04 $ 20.00 © 2004 IEEE

4 Rate-Distortion-Optimized Representation

Beamlet Representation (BR) has been described earlier. Each BR is associated with a Recursive Dyadic Partitioning (RDP) of a square. Let $\mathcal{P}$ denote a RDP and $S \in \mathcal{P}$ indicates that a square S is a *terminal* square in RDP $\mathcal{P}$.

The overall distortion of a RDP $\mathcal{P}$ can be defined as

$$D(y, \mathcal{P}) = \sum_{S \in \mathcal{P}} D(y, S),$$

where y is a subset of pixel in a binary image and $D(y, S)$ is the square limited distortion function that is defined in (2.5). By the above definition, it is implied that inequality (2.7) holds.

The overall rate function that is associated with RDP $\mathcal{P}$ can be defined as follows

$$\begin{aligned} R(\mathcal{P}) &= \#\text{ of bits that are required to code } \mathcal{P} \\ &= [\log_2(3)](\#\text{ of symbols in a BR }) + \sum_{S \in \mathcal{P}} R(S), \end{aligned}$$

where the "# of symbols in a BR" is the numbers of 'N', 'Q', and 'B' in a beamlet representation, the constant $\log_2(3)$ is the average cost to code a symbol, and $R(S) = 2j + 3$, if square S is a 2^j by 2^j square. In other words, $R(S)$ is the number of bits for coding a digital beamlet in this square.

We consider a beamlet representation that is rate-distortion-optimized (r-d-o). The true spirit of a r-d-o beamlet representation solves the following:

$$\min_{\mathcal{P}} R(\mathcal{P}), \qquad \text{subject to } D(y, \mathcal{P}) \leq \overline{D},$$

where constant $\bar{D}$ is prescribed and y again denotes a subset of pixels in a binary image. However, it is not an easy task to solve the above problem directly. Instead, we consider a Lagrangian Multiplier version of the above problem: for $\tau > 0$, consider

$$\min_{\mathcal{P}} R(\mathcal{P}) + \tau D(y, \mathcal{P}), \tag{4.10}$$

where y denotes a subset of pixels. Different values of the Lagrangian multiplier τ are examined. Hence the frontier of the following set—(which sometimes is called the *feasible set*)

$$\{(D(y, \mathcal{P}), R(\mathcal{P})) : \mathcal{P} \text{ is a RDP}\}$$

—is explored.

Note that the problem (4.10) can be solved efficiently by using a *tree pruning* idea that has been developed in the literature, for example the Best-Orthonormal-Basis (BOB) in the wavelet packets. We skip the details on the fast algorithm. In summary, we developed a bottom-up tree pruning algorithm. The resulting RDP is the solution to the problem in (4.10); this can be proved by induction. By running the algorithm for different values of τ, one can find a set of r-d-o beamlet representations.

Proceedings of the Data Compression Conference (DCC'04)
1068-0314/04 $ 20.00 © 2004 IEEE

5 Beamlet Codec

Overview of a Beamlet Codec. There are three steps in a beamlet codec. Only the encoding part is described. Since each step of a beamlet encoder is invertible, so the decoding part is not hard to derive. A binary image that contains contours is the starting point. From the binary image, a beamlet based representation is calculated; this step is called *Beamlet transform.* A Beamlet Representation (BR) is generated. Based on the BR, a stream made by symbols and bits is derived. We call this stream a *symbol stream.* Finally, an arithmetic coder or an entropy coder is applied. The Table 1 describes the entire procedure.

Table 1: Overview of A Beamlet Coder

Binary Shape Image,	
$\uparrow \quad \downarrow$	Beamlet Transform
Beamlet Representation,	
$\uparrow \quad \downarrow$	BR to symbol stream conversion
Symbol Stream,	
$\uparrow \quad \downarrow$	Coding
Bit Stream.	

A BR to Symbol Stream Conversion. Now from the two tables that are generated in the last section, we consider how to generate a stream that is made only by symbols (Q, N, and B) and bits (0 and 1). For clarity, we describe it in two steps.

In the first step, we generate a L by $2J+4$ array, which is made by symbols and binary numbers. Here L denote the number of symbols in Table (3.9) part (a). We first list all the symbols on the first column, in an order that the symbols at coarser scales coming before the symbols at finer scales. The binary representation of *beamlets* are listed thereafter. For a beamlet representation given by Table (3.9) (a) and (b), the array is as follows.

$$
\begin{array}{ll|lll|lll}
l_0 \rightarrow & \text{Q}; & l_1 \rightarrow & \text{Q} & & l_2 \rightarrow & \text{N} & \\
 & & & \text{N} & & & \text{N} & \\
 & & & \text{N} & & & \text{B}_2 & 100100000 \\
 & & & \text{B}_1 & 10011111011; & & \text{B}_3 & 101101010.
\end{array}
\tag{5.11}
$$

Note a notation l_j is used to denote the starting point of symbols coming from scale j. Also note that at the same scale, the order of symbols is not critical. We can choose any reasonable ordering scheme, e.g. raster scan [3].

In the second step, a symbol-and-bit stream is generated by an algorithm that is similar to the one in EZW and SPIHT [6, 5]. Again, the details are omitted. Applying this algorithm to the table in (5.11), the result will look like:

Q, Q, N, N, B, $\text{B}_1(1)$, $\text{B}_1(2)$, $\text{B}_1(3)$, N, N, B, B, $\text{B}_1(4)$, $\text{B}_1(5)$, $\text{B}_2(1)$, $\text{B}_2(2)$, $\text{B}_2(3)$, $\text{B}_3(1)$, $\text{B}_3(2)$, $\text{B}_3(3)$, ... $\text{B}_1(2J)$, $\text{B}_1(2J+1)$, $\text{B}_2(2J-2)$, $\text{B}_2(2J-1)$, $\text{B}_3(2J-2)$, $\text{B}_3^2(2J-1)$.

Here $\text{B}_i(k)$ denotes the kth bit of the ith beamlet.

In *coding,* we reserve the first ten bits to code the value of n. Each symbol (Q, N, B) is coded by two bits, and each binary number (0, 1) is coded by one bit. A binary stream can be generated and further encoded by using an arithmetic coder [7] or a Lempel-ziv coder.

6 Simulations

6.1 Coding Object Shapes in Video Sequences

We present experimental results on a test sequence: *Figure*[1] sequence, QCIF(144×171 pixels per frame). The shape images are cropped to 128 by 128.

Progressivity of a Beamlet Codec. The progressive transmission is illustrated in Figure 4 for the first frame of the video sequence "Figure". In the beamlet representation, the parameters are chosen as $\lambda = 0.2,\ \tau = 0.2$. The coded bit stream is truncated in the tails, and an approximate shape is reconstructed according to the partial stream. From left to the right, when more bits are available, the reconstruction becomes more likely to the original shape.

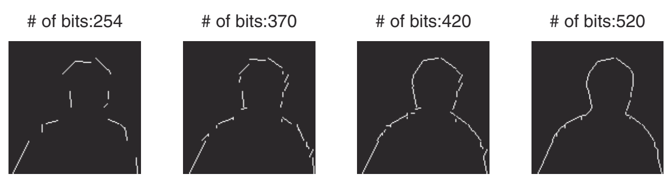

Figure 4: Progressive reconstruction of the sequence *Figure* (frame 0). The number of transmitted bits are shown in the titles.

Rate-Distortion Curve. Figure 5 shows the rate-distortion curves of the beamlet coding averaged over the first 50 frames of the above sequence. It is shown that the rate-distortion curve appears exponential.

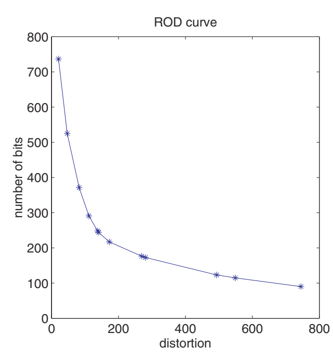

Figure 5: Rate-distortion curves of two sequences.

[1] http://www.ee.columbia.edu/~luoht/research/rvSeg/

Proceedings of the Data Compression Conference (DCC'04)
1068-0314/04 $ 20.00 © 2004 IEEE

6.2 Line Images

We also test the performance of the beamlet coder on some map images. The compression ratios are compared with non-progressive JBIG 2 algorithm [2]. In Figure 6(a), nine maps for country borders are shown. These maps are freely downloadable on the Internet. Each map is a 256 by 256 binary image. We apply lossy coding scenario with the distortion bound 0.4. The reconstructions of these maps are given in Figure 6(b).

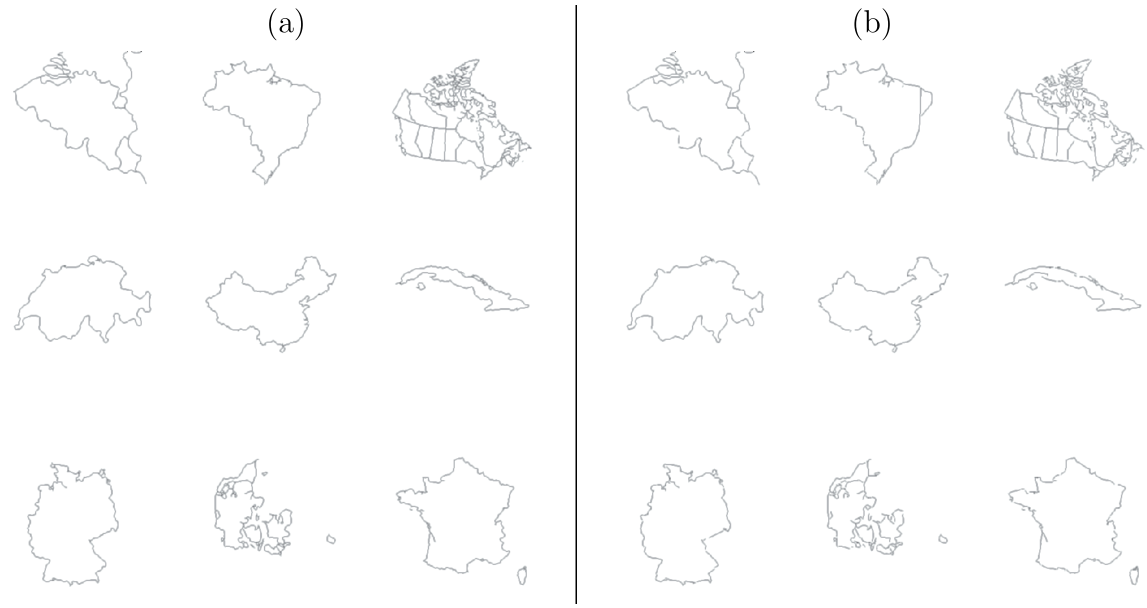

Figure 6: (a) Country borders. We use them as our test images. (b) Reconstruction of maps in the lossy beamlet coding.

In Table 2, the following quantities are provided: (1) **Pix:** the number of nonzero pixels in the binary image; (2) **Dis:** the distortion between the original binary image and the beamlet reconstruction; (Note even if this number is not equal to zero, the reconstruction can be visually nearly identical to the original.) (3) **# bits:** the length of the bit stream; (4) **Ent:** the length of the bit stream after the Lempel-ziv entropy encoding; and (5) **Rat:** the ratio of bit# over the number of nonzeros.

Observations. In lossless coding, BC works slightly better than the JBIG 2. Noticed that if a small distortion is allowed, there is very little visual difference between the coded and original. Hence a lossy BC is implemented. After entropy coding, the numbers of bits are nearly 10% of the number of bits from a lossless JBIG 2 coding. More interestingly, the number of bits after an entropy coding is nearly *half* of the number of nonzero pixels in the original binary image. Note that the number of nonzero pixels is a measure of the complexity of the geometry in a binary image. In this sense, the BC achieves the idea of coding according to geometry.

Table 2: Table of simulation result for lossless coding of border maps of nine countries.

		Lossless (#bits)		Lossy coding			
Country	Pix.	JBIG 2	BC	Dis.	# bits	Ent.	Rat.
Belgium	1233	7888	6114	165	3098	658	0.53
Belize	822	5768	4391	148	2021	474	0.58
Canada	2911	15224	14555	471	7366	1405	0.48
Switzerland	845	6088	4171	118	2135	509	0.60
China	813	5792	4609	141	2041	494	0.61
Cuba	658	4360	3303	108	1542	365	0.55
Germany	871	5792	4916	165	2138	516	0.60
Denmark	1350	7992	6780	170	3735	712	0.52
France	920	6448	5300	139	2339	531	0.58

7 Conclusion

A novel tree-based line coding method is proposed, utilizing a multiscale data structure: beamlets. Its application in coding digital curves (shapes, boundaries) are reported. In simulations, it significantly outperforms existing curve coding standards.

References

[1] D.L. Donoho and X. Huo. Beamlets and multiscale image analysis. In *Multiscale and Multiresolution Methods*, volume 20 of *Lecture Notes in Computational Science and Engineering*. Springer, 2001.

[2] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge. The Emerging JBIG2 Standard. *IEEE Trans. on Circuits and Sys. for Video Tech., 8*(7):838-848 November 1998.

[3] D. Hearn and M.P. Baker. *Computer Graphics: C Version*, 2nd Ed. Prentice Hall, May 1996.

[4] X. Huo. Sparse Image Representation via Combined Transforms. PhD thesis, Stanford, August 1999.

[5] A. Said and W.A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuits and Systems for Video Techn.*, 6:243–250, June 1996.

[6] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 41(12):3445–3462, December 1993.

[7] I.H. Witten, R.M. Neal, and J.G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, vol. 30, no. 6, June 1987.