BEAMLET CODER: A TREE-BASED, HIERARCHICAL CONTOUR REPRESENTATION AND CODING METHOD

Xiaoming Huo and Jihong Chen

School of ISyE, Georgia Institute of Technology
Atlanta, GA 30332-0205
xiaoming, chenjh@isye.gatech.edu

ABSTRACT

A quad-tree-based hierarchical contour representation and coding method is studied. This method is based on multiscale line segments—beamlets. Simulations are reported to evaluate the effectiveness of such an approach. This is a proof-of-concept study. The reported compression ratios are not the "best". However, the idea of tree-based coding is novel; and this idea has good potential to realize a progressive contour coding, which is important in applications such as content-based video transmission.

1. INTRODUCTION

Coding shapes has many important applications in modern multimedia signal processing. One example is the content based video coding. (See [4] and the papers included in the same issue.) An efficient method to code the boundary of an object subsequently facilitates efficient image and video coding. The other example is that in image compression, some artificial images (such as line drawing, maps, some cartoons, blue prints, etc) are made by lines. The special properties of these images requires a representation and coding scheme that is particular suited for linear features.

Compressing and coding linear features has been a long existing subject. They are usually under the topics of *contour coding* and *shape coding*. Back in 60's, runlength coding first emerged [5] in the literature. The basic idea of a runlength codec is to treat an binary image as a nearest neighbor graph—a pixel is connected to its four or eight nearest neighbors—and code the relative positions of the neighboring pixels. Soon after, a chain coding method is proposed [6]. The key idea of a chain coder is to code, at each step, multiple and aligned pixels (that form a line segment) instead of one nearest neighbor as a runlength coder does. Many improvements have been introduced for chain coding. For example, in [10], statistical modeling and arithmetic coding is added as a post-processing tool to fully take advantage of the statistical structure in a chain of line segments. Alternatives other than straight line segment have been explored as basic elements. These works include discrete arcs [1], polygonal curves [8], and a lot more. Researchers have noticed that it is not necessary to exactly follow a curve given by a discrete image. Methods taking into account of the trade-off between the fidelity and efficiency of a contour codec are studied in [13]. More interestingly, the idea of using multiscale structure (i.e. pyramid) to code a contour has been studied by several researchers, e.g. [9, 11]. Despite all these research efforts, relative to generic image compression, the work of representing and coding linear features is still under-developed.

Before describing our methods, let's review what are ideal properties of a contour coder. A *good* contour coder should be *progressive, fast* and *efficient*. "Progressiveness" means that a fraction of the coded stream should allow an approximate reconstruction of the original contour. This property is important for image and video transmission. "Fast" means the coder requires small number of numerical or computer operations. Being fast is an obvious requirement for any coding method. "Efficiency" means that for a given shape, the number of bits is as small as possible.

We report a study on a contour coder that is based on a newly advocated data structure—beamlet [3, 2], which is a collection of multiscale line segments and is designed for analyzing linear features. Similar to 2-D wavelets, the beamlet data structure has an embedded quad-tree structure. A tree based coding strategy, which is analogous to the one in EZW [14] and SPIHT [12], is deployed. Our method is proven to be progressive and fast, but *not* necessarily efficient. Simulational study of this method is reported. Due to the tree structure that is embedded, it is believed that this method has the potential to bring a fully progressive contour coder. However, this work is still at a proof-of-concept stage. Refining the coding method and developing a fully progressive coder are our future research topics.

The contribution of the present paper is on the feasibility of combining a tree-based coding and beamlets data structure. This the first time to have a tree-based coder for multiscale line segments. We consider the reported simulation results *preliminary.* There will be amble space for improvement. This paper may inspire new researches in this area.

In Section 2, we describe a beamlet coder. Simulations on country borders are reported in Section 3. Some final marks are provided in Section 4.

2. BEAMLET CODEC

We first describe related literature on beamlets, then describe our beamlet codec.

2.1. Beamlet

Beamlets are introduced as a tool to analyze linear objects in 2-D [3]. In summary, beamlets are multiscale line segments; comparing to the original set of all possible line segments, the dictionary of beamlets has low order of 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. Due to the space limitation, the details on this data structure are omitted and readers are referred to the reference.

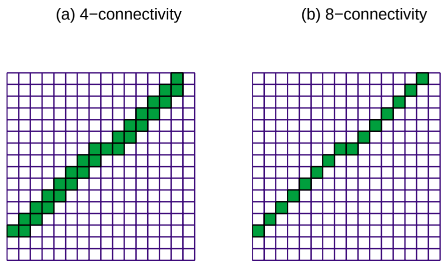

Figure 1: Two discrete beamlets. One is generated by allowing connecting to four neighboring pixels (a); the other eight neighboring pixels (b).

2.2. Beamlet Codec

Overview. 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 a binary image, a beamlet based representation is calculated; this step is coded *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 following flow chart describes the entire procedure.

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

In the following, the above three steps are articulated.

Beamlet Transform. In this section, we describe how to generate a beamlet representation from a binary image. In a given box, a discrete beamlet is determined by its two endpionts. Note that an endpoint can only be on the boundary of this box. When the two endpoints are given, the discrete beamlet is given by interpolation, which can be derived by following the idea of Bresenham [15]. Two topological conditions are considered. One only allows a pixel to be connected to its four nearest neighbors. The other allows a pixel to be connected to its eight nearest neighbors—adding four more diagonally connected pixels. Figure 1 gives two discrete beamlets with the same endpoints, satisfying different topological conditions.

For a given binary image, a top-down approach can be applied. A squared image is assumed. Starting from the entire image (the coarsest scale),we exam if there exists a beamlet that *fits* this binary image. The fitness is evaluated by considering two quantities: the fraction of pixels of a beamlet that is on the shape, and the fraction of pixels of the shape that is also on the beamlet. If both fractions are close to one, there is a *good* fit. If no beamlet fit the content of a binary image well, and the image is not blank, then the image is divided into smaller boxes, and the fittings of beamlets in these smaller boxes are considered. This procedure is repeated until the finest level is reached.

As an example, we consider a binary image in Figure 2. At the coarsest level, no beamlet can fit well; hence a quadratic split

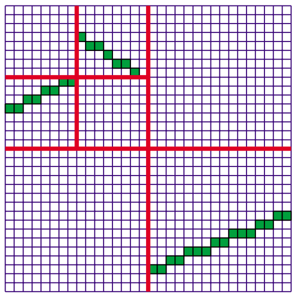

Figure 2: An example of binary image. Note this image is not a shape. It is made for illustrating the idea of beamlet representation.

is assigned. At the next scale, two boxes contains no dark pixels. They are labeled as empty. 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 on dark pixels or are fitted by beamlets. The corresponding partitioning and the vertices of beamlets are given as follows.

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

(1)

where Q, N, and B respectively denote Quadratic Split, No Split, and Beamlet. Symbol "–" means no content. The coordinates of beamlets are:

	first vertex	second vertex
B_1	3	40
B_2	5	16
B_3	5	26

(2)

Note that from the above two tables, we can reconstruct a binary image.

A BR to Symbol Stream Conversion. Now from 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 3 array, which is made by symbols and integral numbers. Here L denote the number of symbols in table (1). In the second step, a symbol-and-bit stream (or simply called a symbol stream) is generated.

To generate the three-column array, we first list all the symbols, the symbols at coarser scales coming before the symbols at finer scales. For the *beamlets*, we use the other two elements on the same row to record the positions of its vertices (endpoints). For a beamlet representation given by table (1) and (2), a three-column array is:

$$\begin{array}{lclll}
l_0 & \rightarrow & \mathrm{Q} & - & - \\
l_1 & \rightarrow & \mathrm{Q} & - & - \\
 & & \mathrm{N} & - & - \\
 & & \mathrm{N} & - & - \\
 & & \mathrm{B}_1 & \mathrm{B}_1^1(=3) & \mathrm{B}_1^2(=40) \\
l_2 & \rightarrow & \mathrm{N} & - & - \\
 & & \mathrm{B}_2 & \mathrm{B}_2^1(=5) & \mathrm{B}_2^2(=16) \\
 & & \mathrm{B}_3 & \mathrm{B}_3^1(=5) & \mathrm{B}_3^2(=26) \\
 & & \mathrm{N} & - & -
\end{array} \quad (3)$$

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 [7].

A symbol and bit stream is made by the following algorithm.

```
Algorithm I

For $j_1 = 0 : J$, where $J = \log_2(n)$ and $n$ is the size of the image,
    1. Enumerate all symbols at level $j_1$, by enumerating first elements of rows between $l_{j_1}$ and $l_{j_1+1} - 1$.
    2. Enumerate bits by following the following procedure.
       For $j_2 = 0 : j_1$,
           enumerate the $(j_1 - j_2 + 1)$st bit of all the binary representation of all beamlet vertices.
       End
End.
Enumerate all remaining bits, by following the order given in the second "for" loop above.
```

Note the above algorithm is similar to the algorithm in EZW and SPIHT [14, 12]. Applying this algorithm to the table in (3), the result look like:

Q, Q, N, N, B, $\mathrm{B}_1^1(1)$, $\mathrm{B}_1^2(1)$, N, B, B, N, $\mathrm{B}_1^1(2)$, $\mathrm{B}_1^2(2)$, $\mathrm{B}_2^1(1)$, $\mathrm{B}_2^2(1)$, $\mathrm{B}_3^1(1)$, $\mathrm{B}_3^2(1)$, ... $\mathrm{B}_1^1(J+1)$, $\mathrm{B}_1^2(J+1)$, $\mathrm{B}_2^1(J)$, $\mathrm{B}_2^2(J)$, $\mathrm{B}_3^1(J)$, $\mathrm{B}_3^2(J)$.

Here $\mathrm{B}_i^j(k)$ denotes the kth bit of the jth vertex of the ith beamlet. We again consider the example in Figure 2, in which we have $n = 32$ and $J = 5$. Note that at scale j, it takes $J + 2 - j$ to code a coordinate of a vertex. So it respectively takes 6 and 5 bits to code a vertex at scale 1 and 2. The following gives the binary representations of all vertices.

$$\text{scale 1} \left\{ \begin{array}{lll} \mathrm{B}_1^1 & 3 & 000010; \\ \mathrm{B}_1^2 & 40 & 100111; \end{array} \right.$$

$$\text{scale 2} \left\{ \begin{array}{lll} \mathrm{B}_2^1 & 5 & 00100; \\ \mathrm{B}_2^2 & 16 & 01111; \\ \mathrm{B}_3^1 & 5 & 00100; \\ \mathrm{B}_3^2 & 26 & 11001. \end{array} \right.$$

Note that in decimal representation, a coordinate of a vertex starts with 1; in a binary representation, to save space, it starts with 0. So coordinate 5 is coded as 00100, not 00101. The final symbol stream based on the BR in (1) and (2) is

Q, Q, N, N, B, 0, 1, N, B, B, N, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, EOF.

Here "EOF" stands for the end of the file.

Coding. The symbol stream that is generated above contains the following elements: Q, N, B, 0, 1, and EOF. We may reserve the first few bits to code the value of n. This stream can be encoded by using an arithmetic coder [16] or an entropy coder.

3. SIMULATIONS

To evaluate the performance of the beamlet coder, maps of borders for thirty countries are taken. These maps are freely downloadable on the Internet. Each map is a 256 by 256 binary image. In Figure 3(a), nine snapshots of these images are given.

A lossy coding scenario is considered. In the lossy beamlet coding, the faithfulness condition of the beamlet transform is set to be $\varepsilon = 0.8$: both fractions in measuring the fitness of a beamlet representation must be at least $\varepsilon = 0.8$. The first nine reconstructions of lossy coding are given in Figure 3(b). It is clear that these reconstructed figures are close to the original images.

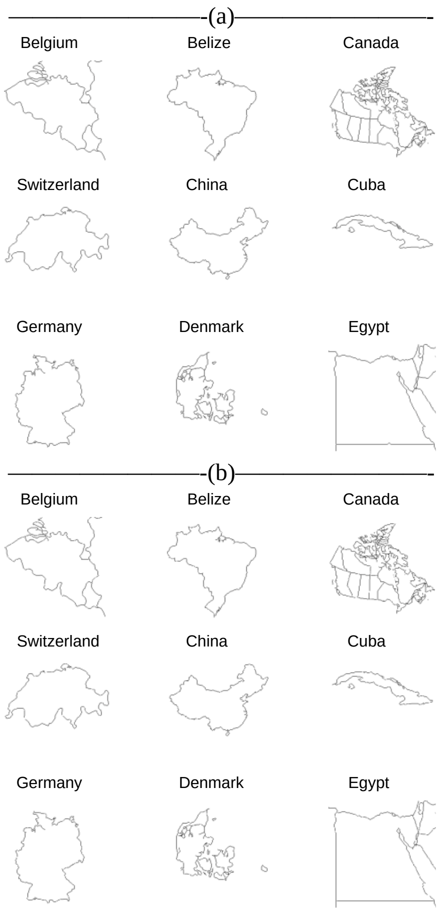

Figure 3: (a) Country borders. We use them as our test images. (b) Reconstruction of maps in the lossy beamlet coding.

In the Table 1, the following quantities are provided:

1. **Cnt.1.** The number of nonzeros in the binary image.
2. **Dev.** The deviation between the original binary image and the beamlet reconstruction. Note even if this number is not

equal to zero, the reconstruction can be visually very close to the original.

3. **Cnt.2.** The length of the symbol stream.
4. **Ent.** The entropy (with base 2) of the symbol stream.
5. **Rat.** The ratio of Entropy over the number of nonzeros.

Country	Cnt.1	Dev.	Cnt.2	Ent.	Rat.
Belgium	1233	61	4798	9482	7.7
Belize	822	37	3454	6965	8.5
Canada	2911	106	12248	24050	8.3
Switzerland	845	50	3184	6363	7.5
China	813	30	3628	7389	9.1
Cuba	658	36	2450	4817	7.3
Germany	871	40	3710	7583	8.7
Denmark	1350	58	5436	10641	7.9
Egypt	1336	54	4026	7988	6.0
Finland	594	38	2466	5026	8.5
France	920	35	4036	8359	9.1
Georgia	1137	102	3520	7066	6.2
Greece	1287	43	5792	11703	9.1
Israel	859	82	2758	5403	6.3
India	1020	48	4410	9030	8.9
Iceland	1071	28	4752	9596	9.0
Italy	1054	45	4390	8847	8.4
Jordan	767	107	2138	4269	5.6
Japan	703	19	3384	6867	9.8
Korea(North)	950	38	3978	7984	8.4
Korea(South)	1226	43	5348	10618	8.7
Mexico	828	48	3252	6594	8.0
Porland	705	29	2886	5901	8.4
Russian Federation	1054	34	4666	9476	9.0
Sweden	675	21	2864	5844	8.7
Singapore	812	102	2234	4388	5.4
Turkey	742	21	3192	6395	8.6
United Kingdom	1098	29	4912	9856	9.0
United States	839	50	3504	7147	8.5
Vietnam	761	15	3616	7419	9.7

Table 1: Table of simulation result for border maps of 30 countries.

The ratios in the last column indicate, to some extent, how well a beamlet coder compare to other coder such as runlength. Since a runlength coder roughly takes $O(K)$ bits, where K is the number of nonzeros. We see that the current version of beamlet coder is a little bit worse than a runlength coder. But note that neither a runlength coder, nor a chain coder is hierarchical or progressive. The beamlet coder is based on a quadtree, hence it is hierarchical, and can be progressive.

Note that the current coding method has a lot of space for improvement. We leave it as future research.

In this paper, due to the variations in implementing an entropy coder or an arithmetic coder, the final numbers of these coders are not so informative; so we choose not to report them.

4. CONCLUSION

A Novel, tree-based line coding method is proposed. Its application in coding country borders are reported. This method has good potential to lead to fully adaptive and progressive coders for linear features.

5. REFERENCES

[1] S. Biswas. Contour coding through stretching of discrete circular arcs by affine transformation. *Pattern Recognition*, 34:63–77, 2001.

[2] D. Donoho and X. Huo. Beamlet pyramids: A new form of multiresolution analysis, suited for extracting lines, curves, and objects from very noisy image data. In *Proceedings of SPIE*, volume 4119, July 2000.

[3] 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.

[4] M. Etoh, J. Ostermann, and T. Sikora. Editorial: special issue on shape coding for emerging multimedia applications. *Signal Processing: Image Communication*, 15:581–583, 2000.

[5] H. Freeman. On the coding of arbitrary geometric configurations. *IRE Trans. Electron. Comput.*, vol. EC-10, pp.260-268, June 1961.

[6] H. Freeman. Application of the generalized chain coding scheme to map data processing. *Proc. IEEE Comput. Soc. Conf. Pattern Recog. Image Processing*, Chicago, IL, May 1978, pp. 220-226.

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

[8] J. Kim, A.C. Bovik, and B.L. Evans. Generalized predictive binary shape coding using polygon approximation. *Signal processing: Image Communication*, 15:643–663, 2000.

[9] J. Koplowitz and J. DeLeone. Hierarchical representation of chain-encoded binary image contours. *Computer Vision and Image Understanding*, 63(2):344–352, 1996.

[10] C.C. Lu and J.G. Dunham. Highly efficient coding schemes for contour lines based on chain code representation. *IEEE Trans. Communications*, 39(10):1511–1514, October 1991.

[11] P. Meer, A. Sher, and A. Rosenfeld. The chain pyramid: hierarchical contour processing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(4):363–376, April 1990.

[12] 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.

[13] G.M. Schuster and A.K. Katsaggelos. An optimal polygonal boundary encoding scheme in the rate distortion sense. *IEEE Trans. Image Processing*, 7(1):13–26, January 1998.

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

[15] R. Ulichney. Bresenham-style Scaling. *Proceedings of the IS&T Annual Conference*, pp.101-103, Cambridge, Mass., 1993.

[16] 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.