A MIXED TRIANGULAR AND QUADRILATERAL PARTITION FOR FRACTAL IMAGE CODING

Franck Davoine, John Svensson and Jean-Marc Chassery

Laboratory TIMC-IMAG, IAB, Domaine de la Merci
38706 LA TRONCHE Cedex, FRANCE
e-mail: Franck.Davoine@imag.fr

ABSTRACT

This paper presents a new partitioning scheme for fractal image coding, based on triangles and quadrilaterals. The aim is to have the advantage of the triangles over the square and rectangular blocks, in terms of adaptivity to the image (reduction of the block effect), and to reduce the blocks number by grouping neighboring triangles into quadrilaterals. Quadrilaterals permit a reduction of the number of local contractive affine transformations composing the fractal transform, and thus to increase the compression ratio, while preserving the visual quality of the decoded image.

1. INTRODUCTION

The partitioning scheme is very important for fractal image compression. Y. Fisher in [3] proposes the quadtree-based partitioning and a more flexible one: the H-V partition, based on rectangles [4].This last one is arranged so that edges in the image tend to run diagonally through the blocks. In this paper we propose the Delaunay triangulation satisfying the following properties:

- it is fully flexible because it can be computed on a set of points well localized on the image support, depending on its grey level content
- the storage of the partition is efficient: one bit is used to code the addition or the suppression of a point (triangle vertex) in the triangulation
- triangles can have any orientation, and reduce the block effect in the decoded image.

2. THE FRACTAL TRANSFORM

Let us consider an *eventually contractive operator* W defined from the metric space (X, d) of digital images to itself, where d is a given distance. W is eventually contractive if it is contractive at the n^{th} iterate. W has a unique *fixed point* $\mathcal{A}_t$ given by:

$$\lim_{k\to\infty} W^{\circ k}(B) = \mathcal{A}_t = W(\mathcal{A}_t), \quad \forall B \in X, \qquad (1)$$

where $W^{\circ k}$ means the k^{th} iterate of W. In this case, the image $\mathcal{A}_t$ is completely defined by the operator W. The principle of fractal image compression is to construct an operator W for which the fixed point $\mathcal{A}_t$ is a close approximation to a given image A to encode. The operator W has to be eventually contractive and to minimize the distance $d(W(A), A)$. This requirement is given by the generalized collage theorem [3]:

$$d(A, \mathcal{A}_t) \leq \frac{1}{1-s}\,\frac{1-\sigma^n}{1-\sigma}\, d(A, W(A)) \qquad (2)$$

where s is the contractivity of $W^{\circ n}$ and σ is the Lipschitz factor of W (if W is contractive, $\sigma < 1$). The formula (2) shows that the minimization of $d(A, W(A))$ is suboptimal but much easier to realize than the minimization of $d(A, \mathcal{A}_t)$, as $\mathcal{A}_t$ is unknown during the encoding step. One solution, originaly proposed by Jacquin [5], is to partition the image A into N non-overlapping squares R_i,

$$A = \bigcup_{i=1}^{N} R_i \quad \text{with} \quad R_i \cap R_j = \emptyset,$$

and to map on each square R_i the most similar transformed block $\omega_i(D_{\alpha(i)})$. α is an application from $[1 \dots N]$ to $[1 \dots M]$ where $M \leq N$: the block $D_{\alpha(i)}$, different and larger than R_i, can be choosen anywhere in the image and can be mapped onto one or more different blocks of the partition R. In this case, the transformed image A with the operator W discussed above is expressed as:

$$W(A) = \bigcup_{i=1}^{N} \omega_i(D_{\alpha(i)}). \qquad (3)$$

W is composed of local transforms ω_i and thus returns the union of transformed parts of the image A, in order to have A very close to $W(A)$. This is possible if the image A is sufficiently *piecewise self-similar* and if the partition R is computed in a grey level dependent way. The compressed representation of the image A typically contains the partition construction rule, the N local contractive transforms w_i coefficients associated to each block R_i and the coding of the corresponding block D_i shapes and locations in the image. We show in section 5 that the triangles D_i are searched in a partition D, (not anywhere in the image) in order to make the encoding step possible, in terms of computing times.

3. DELAUNAY TRIANGULATION

Let S be a set of points in the plane. The Delaunay triangulation $DT(S)$ associated to S is the unique triangulation

The authors thank the GDR 134 TDSI CNRS

with *empty circles*. More formally, $DT(S) = \{(p,q,r) \in S^3, C(p,q,r) \cap S - (p,q,r) = \emptyset\}$, where $C(p,q,r)$ is the circle circumscribed by the three points p, q, and r. An important property of the Delaunay triangulation for our compression application is the *local max-min angle* criteria which maximizes the smallest interior angle of the triangulation. This property reduces the error in the localization of each pixel in the triangles.
We choose an incremental approach to compute the Delaunay triangulation because of its two main advantages: the *dynamic* and *optimal* aspects [2]. The computation of the set of points adapted to the image content is done with a *split and merge* approach [1] as it is detailed in the following algorithm:

1. Initialization of the process with a lattice on the image support.
2. Repeat until convergence: homogeneity of all the triangles (standard deviation of each triangle is less than a fixed threshold E) and triangle sizes greater than a fixed threshold T.
   (a) Compute the Delaunay triangulation (dynamic management with the incremental algorithm). In practice 10000 triangles are computed in less than 1 second on a Silicon Graphics Indigo workstation.
   (b) Compute the grey level mean value, standard deviation and size of each triangle.
   (c) Split each non-homogeneous triangle (standard deviation is greater than E) by adding a point on its barycenter.
3. Merge the triangles under the *star polygon* criteria. If all the triangles issued from a point p have the same grey level according a threshold, then p is suppressed. This is equivalent to triangulate only the star polygon which is defined by the set of neighbors of p in the Delaunay triangulation (see Figure 1).

The step (3) consists in suppressing "useless" triangle vertices from the set of points. With such an algorithm, the image is adaptively divided into small triangles that cover edged and textured regions, and into larger triangles on homogeneous regions. Figure 2 presents a result computed on the well known Lena image.

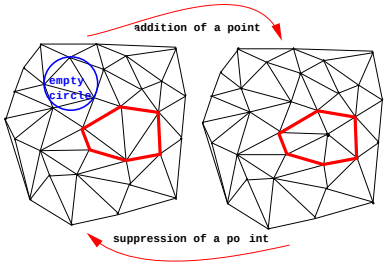

Figure 1: Addition and suppression of a point in Delaunay triangulation

There are many other ways to construct the partition. A possibility that can be used for the computation of the set of points consists in starting from a dense and regular set of points, and in operating only one merge step on the initial triangulation. Small triangles stay on the textured regions of the image and large triangles appear on the homogeneous regions.

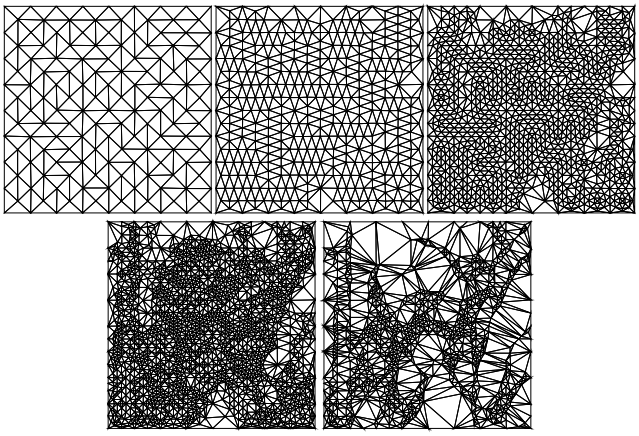

Figure 2: Split and Merge algorithm computed on the Lena image and initialized on a lattice. First row: initialization (255 triangles), split 1 (734) and split 2 (971). Second row: split 3 (3720) and merge step (1867 triangles)

4. QUADRILATERALS

This step consists in extracting from the triangulation neighboring triangles that form convex quadrilaterals. The aim is to significantly reduce the number of blocks, and consequently the number of transformations to code.

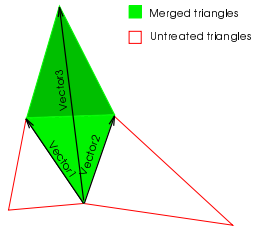

Figure 3: Extraction rule of quadrilaterals.

The first step is to test whether or not a quadrilateral is concave (or almost degenerated). This is done by comparing the signs of the two vector products $vector1 \times vector3$ and $vector2 \times vector3$ as it is shown in Figure 3. If they have opposite signs the two triangles will form a convex quadrilateral. Otherwise they will result in a concave quadrilateral and we redo the test for another neighboring triangle. Thereafter, we use a multiple test regarding the differences of mean values and standard deviations between the two triangles. If the values are less than a given threshold we merge the two triangles. This algorithm returns many quadrilaterals on the homogeneous and textured regions. The contours stay covered by small triangles (Figure 4).

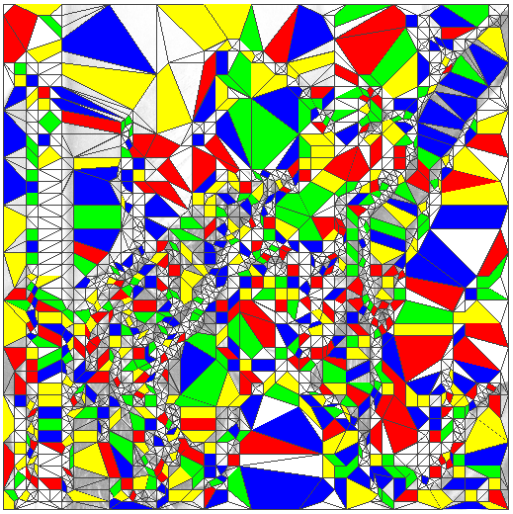

Figure 4: Mixed quadrilateral and triangular partition of the image Lena: 1706 triangles + 715 quadrilaterals.

5. ENCODING ALGORITHM

The encoding algorithm is based on a two-level Delaunay triangulation of the image support. One of the two levels returns the blocks D_j (partition D) discussed in section 2. The partition D is composed of regular blocks and is not adapted to the image content. It provides a pool of blocks available for the encoding process. Moreover, the size of the pool is increased by considering the 6 or 8 isometries (rotations and flips) applied on triangular or quadrilateral blocks. The second partition (partition R) returns blocks R_i of smaller or equal "mean area" and is computed with the algorithms proposed in section 3. It is important that the two partitions R and D share a large number of self-similar blocks. Neighboring triangles that form convex quadrilaterals are then grouped; the other triangles are left unchanged.
Transformations ω_i composing the operator W determine the way to map a block $D_{\alpha(i)}$ onto a block R_i and is written as:

$$\begin{aligned}\omega_i(D_{\alpha(i)}) &\equiv \omega_i(x_m, y_m, f(x_m, y_m)) \quad \forall\ (x_m, y_m) \in D_{\alpha(i)} \\ &= (x'_m, y'_m, f(x'_m, y'_m)) \\ &= (v_i(x_m, y_m), s_i\ f(x_m, y_m) + o_i),\end{aligned}$$

where $f(x_m, y_m)$ is the luminance of the pixel at coordinates (x_m, y_m) in the block $D_{\alpha(i)}$, $f(x'_m, y'_m)$ is the luminance of the pixel at coordinates (x'_m, y'_m) in the block R_i, and s_i, o_i respectively control the contrast and brightness of the grey level transformation. The spatial transformation v_i maps a block $D_{\alpha(i)}$ onto a block R_i ($v_i(D_{\alpha(i)}) = R_i$). In general, two planar quadrangles are not affine images of each other. Consequently, we use *projective* tranformations to map quadrilaterals :

$$v_i(x_m, y_m) = (x'_m, y'_m) =$$

$$\left[\left(\frac{a_i x_m + b_i y_m + e_i}{g_i x_m + h_i y_m + 1}\right), \left(\frac{c_i x_m + d_i y_m + f_i}{g_i x_m + h_i y_m + 1}\right)\right] \qquad (4)$$

The coefficients $a_i \ldots h_i$ are computed considering the four vertices of the block $D_{\alpha(i)}$, and the four vertices of the block R_i. The transformation v_i of a triangle onto another triangle is *affine* and is given by (4) with g_i and $h_i = 0$.
The similarity (in terms of grey levels) between a block R_i and the transformed block $D_{\alpha(i)}$ is measured with the square error (SE) given by:

$$d(R_i, \omega_i(D_{\alpha(i)})) =$$

$$\sum_{m=1}^{n_r} \left[f(x'_m, y'_m) - s_i f(v_i^{-1}(x'_m, y'_m)) - o_i\right]^2$$

for any $(x'_m, y'_m) \in R_i$, where n_r is the number of pixels included in R_i. We use the inverse transformation v_i^{-1} in order to consider all the "discret" points in the block R_i. If the block R_i is smaller than $D_{\alpha(i)}$, the block $D_{\alpha(i)}$ is subsampled. However a non uniform subsampling is done when we compare two different shape quadrilaterals. A better solution that we use is described below (Fig. 5):
The transformation of the four vertices is given by:

$$v_i \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x'_k \\ y'_k \end{bmatrix} = \begin{bmatrix} a_k\ x_k \\ b_k\ y_k \end{bmatrix} \quad \forall k = 1 \ldots 4$$

The transformation of a point (x_m, y_m) in the block $D_{\alpha(i)}$ is given by:

$$\begin{bmatrix} x'_m \\ y'_m \end{bmatrix} = \begin{bmatrix} a_m\ x_m \\ b_m\ y_m \end{bmatrix}$$

with:

$$\begin{bmatrix} a_m \\ b_m \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \end{bmatrix} \begin{bmatrix} (1-\alpha)(1-\beta) \\ \alpha(1-\beta) \\ \beta(1-\alpha) \\ \alpha\beta \end{bmatrix}$$

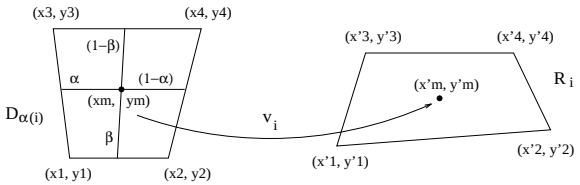

Figure 5: Mapping of a quadrilateral onto another.

The point (x'_m, y'_m) is generally not on the discrete grid of the image support. A supplementary bilinear interpolation scheme can be used to compute its grey level value. We choose the simplest approximation to its nearest "discret" point.
The contractivity of the transform W is controlled by the parameters s_i. If $s = Max_i\{s_i\}\ \forall\ i = 1 \ldots N$ is less than 1, W is insured to be eventually contractive, considering the L_2 distance. Moreover, it is experimentaly verified that W stays eventually contractive if $s < 1.5$ [3].
The computation time for a 256×256 image coding is about 2 to 10 minutes, depending on the desired compression ratio and consequently the number of blocks in the partition R. This long computing time is mainly due to the scanning of pixels in triangular blocks.

Figure 6: Associations at the end of the encoding step: the bold-lined square is mapped onto each fine quadrilateral, and the hatched triangle onto each fine triangle

6. DECODING ALGORITHM

The decoding consists, after reconstruction of the two partitions R and D, in iterating the operator W, starting with any initial image B. The reconstructed image is given by $A \simeq \lim_{n\to\infty} W^{o_n}(B)$. One iteration consists in scanning each block R_i and in applying the transformation ω_i (affine or projective, depending on the block shape) on their corresponding domain block $D_{\alpha(i)}$. The grey level $f_n(x'_m, y'_m)$ of the pixel m in the block R_i at the n^{th} iteration of W is given by :

$$f_n(x'_m, y'_m) = s_i \; f_{n-1}(v_i^{-1}(x'_m, y'_m)) + o_i \quad \forall\, (x'_m, y'_m) \in R_i.$$

The algorithm needs 5 to 10 iterations to converge. We observe that the number of iterations is mainly dependent on the blocks size differences between the two partitions. The computation time for a 256×256 image is less than 5 seconds. Studies on fast decoder convergence have been done in [6].

7. COMPRESSION

To encode an image, we need to store the coefficients of the N mappings ω_i, composing the operator W. One mapping is needed for each block R_i in the partition R. The Delaunay tessellation is coded in a graph environment.
Thus, for one transformation ω_i, it is only necessary to code:

- the position of the triangle D_i in Delaunay graph with M_T bits ($M_T = \lfloor \log_2(\text{nb. of blocks in partition } D) \rfloor$.
- the orientation for the collage with 3 bits (6 possibilities to map a triangle onto another, and 8 possibilities to map a quadrilateral onto another)
- the scale (s_i) coefficient quantized with 6 bits
- the offset (o_i) coefficient quantized with 6 bits

In addition, the method of computing the image adapted partition R must be coded. Since we always start with a known regular distribution of points to construct the Delaunay triangulation, it is only usefull to code the split and merge processes. At each stage, one bit codes a triangle vertex (non-)addition or (non-)suppression. For example, the final partition in figure 2 is coded with 3460 bits. The string of 0 and 1 can, moreover, be entropy compressed. The remaining coefficients $a_i \ldots h_i$ are not stored because they can be computed by the decoder, knowing the two partitions R and D. The total number of bits necessary to code the operator W is given by: $N_{tot} = N(6+6+3+M_T)+X$ with N = total number of blocks R_i in the partition R and X is the number of bits to code the partition R.

8. RESULTS AND CONCLUSION

We present decoding results computed on the grey level image *Lena* (512×512). The use of quadrilaterals allows:

- to increase the compression ratio by reducing the number of transformations ω_i to code
- or to increase the quality of the decoded image, while preserving a constant compression ratio, by computing smaller triangles on the contours of the image, before the quadrilaterals extraction

	triangles	triangles + quadrilaterals
$T_c = 24.7{:}1$	30.14 dB	29.95 dB
$T_c = 31.6{:}1$	29.16 dB	29.16 dB
$T_c = 34.9{:}1$	28.80 dB	29.00 dB

Table 1: Decoding results: Peak signal to noise ratios ($PSNR$) at different compression ratios (T_c).

Table 1 confirms that if the compression ratio increases, the decoded image quality stays higher with a mixed triangular and quadrilateral partition R than with a triangular partition. The reconstructed images contain fewer artifacts and block effects on the diagonal edges, than methods based on the quadtree partitioning. This new partitioning scheme is convenient for fractal image compression and could be more optimized. We are working on using a gradient image to more constrain the triangulation.

9. REFERENCES

[1] E. Bertin, F. Parazza, and J. M. Chassery. Segmentation and measurement based on 3D Voronoi diagram: Application to confocal microscopy. *Computerized Medical Imaging and Graphics*, 17:0–8, 1993.

[2] A. Bowyer. Computing dirichlet tessellations. *The computer J.*, 24(2):162–166, 1981.

[3] Y. Fisher, editor. *Fractal Image Compression: Theory and Application to Digital Images*. Springer Verlag, New York, 1995.

[4] Y. Fisher and S. Menlove. Fractal encoding with HV partitions. In *Fractal Image Compression: Theory and Application to Digital Images*, pages 119–136. Y. Fisher (Ed.), 1995.

[5] A. E. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1992.

[6] G. E. Øien and S. Lepsøy. Fractal-based image coding with fast decoder convergence. *Signal Processing*, 40:105–117, October 1994.