Engineering with Computers (2005) 20: 323–332
DOI 10.1007/s00366-004-0301-7

ORIGINAL ARTICLE

A. Plaza · J. P. Suárez · M. A. Padrón

Fractality of refined triangular grids and space-filling curves

Received: 13 June 2003 / Accepted: 30 January 2004 / Published online: 8 October 2004
© Springer-Verlag London Limited 2004

Abstract In this paper we explain the fractal geometry of refined and derefined triangular and tetrahedral meshes by means of the application of iterated function systems (IFS). These meshes feature a remarkable amplifying invariance under changes of scale. The applications of IFS families are shown equivalent to the use of adaptive strategies that combine the refinement procedure with the derefinement procedure. In addition, space-filling curves (SFC) are used to assign a binary code for any 2D triangular refined mesh. SFC are also shown as useful for the problem of automatic domain decomposition.

Keywords Iterated function systems · Local refinement · Simplicial grids · Bisection · Space-filling curves

1 Introduction

It has been shown that fractals arise naturally in the mathematical formulation of various natural phenomena in many fields [1–3]. The iterated function system (IFS) model provides a convenient framework for the description, classification and generation of fractals. An IFS model is generated by a finite number of contractive linear transformations. For a comprehensive introduction to this field refer to Barnsley [1] and Falconer [4].

A. Plaza (✉)
Department of Mathematics,
University of Las Palmas de Gran Canaria,
35017, Las Palmas de Gran Canaria, Spain
E-mail: aplaza@dmat.ulpgc.es

J. P. Suárez
Department of Cartography and Graphic Engineering,
ULPGC, Spain
E-mail: jsuarez@dcegi.ulpgc.es

M. A. Padrón
Department of Civil Engineering,
ULPGC, Spain
E-mail: mpadron@dic.ulpgc.es

On the other hand, numerical grid generation and refinement of a given grid are the main steps in many problems of various areas such as computational geometry, computer graphics, geometric modelling or the finite element method (FEM). Triangles and tetrahedra have been widely used for local adaptive refinement, and several bisection-based algorithms in two [5–7] and three [8, 9] dimensions have been presented in recent years.

In this paper, the fractal geometry of the refinement and derefinement algorithms based on the 4-triangles longest-edge (4T-LE) partition of Rivara [5, 6] are pointed out. We relate the fractal behavior of the triangular meshes obtained by these algorithms to the application of IFS. We also provide an example of the application of space-filling curves (SFC) connecting the barycenters of the triangles to domain decomposition in an L-shaped domain in three dimensions. As a preliminary example see Fig. 1 in which the fractality of the refinement area is observed, since in the refined zones the mesh presents a remarkable amplifying invariance under changes of scale.

The paper is organized as follows: in Sect. 2 we provide some basic definitions and briefly review some of the most used partitions in four triangles. Our latest versions of the refinement and derefinement algorithms in two dimensions are summarized in Sect. 3. The algorithms are based on the the work of M.C. Rivara [5, 6] and, in fact, they can be seen as other versions of her algorithms. However, the change in point of view makes the extensions of similar algorithms to three dimensions possible [9, 10]. In Sect. 4, and based on Plaza [11], the application of IFS families is shown to be equivalent to the use of an adaptive strategy that combines the refinement and derefinement algorithms. Then we focus on SFC following the Hilbert's geometric generation procedure [12] in Sect. 5. The extension of some previous ideas to three dimensions is given in Sect. 6. Some numerical examples both in two and three dimensions are provided in Sect. 7, and we end with some conclusions.

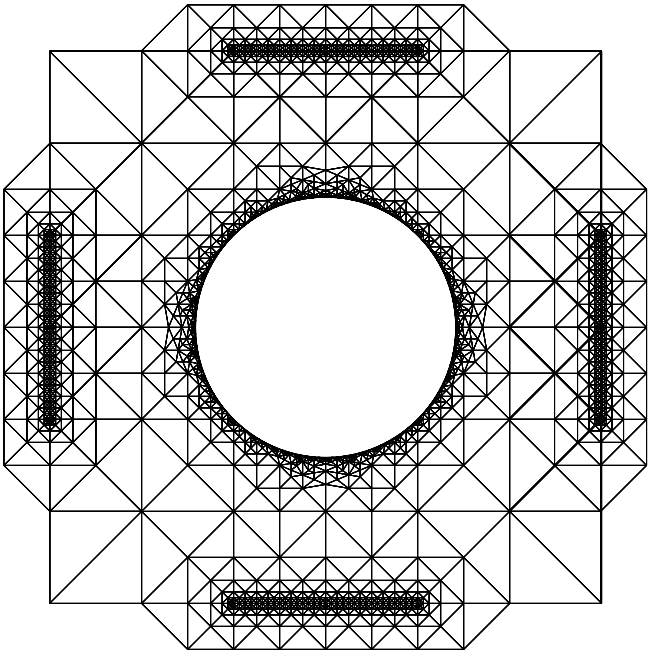

Fig. 1 Example of local refinement and fractal behavior

2 Basic definitions and partitions

Definition 1 A d-triangle is the notation for a triangle (or simplex) of dimension d: a 0-triangle is a point, a 1-triangle is a segment, a 2-triangle is a triangle (default) and a 3-triangle is a tetrahedron. A d-triangle can also be defined as the closed subset $T \subset \mathbf{R}^n$, $0 \le d \le n$ given by the convex linear hull of $d+1$ vertices $x_1, x_2, \ldots, x_{d+1}$ in $\mathbf{R}^n$:

$$T = [x_1, \ldots, x_{d+1}] = \left\{ x = \sum_{i=1}^{d+1} \lambda_i x_i \,\middle|\, \sum_{i=1}^{d+1} \lambda_i = 1; \quad 0 \le \lambda_i \le 1, \ 1 \le i \le d+1 \right\}$$

Definition 2 A triangulation T of a bounded set Ω with polygonal boundary is a partition of Ω into a set of triangles $T = \{t_1, t_2, \ldots, t_n\}$ such that $\Omega = \bigcup_{i=1}^{n} t_i$, $\mathrm{int}(t_i) \neq \emptyset$ for all i, $\mathrm{int}(t_i) \cap \mathrm{int}(t_j) = \emptyset$ if $i \neq j$, and $t_i \cap t_j$ if not empty is a common vertex, or a common edge (or in three dimensions it can also be a common face).

The last condition is frequently referred to as the conformity (or consistency) condition and is usually required in finite element codes. In some longest-edge (LE) refinement algorithms in 2D [5–7] and in 3D [8, 9] the edges are bisected by following the bisected edge and the longest edge of the neighbor triangle.

2.1 The newest vertex bisection

This method was presented by Mitchell [7].

Definition 3 The newest vertex bisection of a triangle t_0 is obtained by connecting one of the vertices, called the peak, to the midpoint of the opposite edge, called the base, as in Fig. 2.

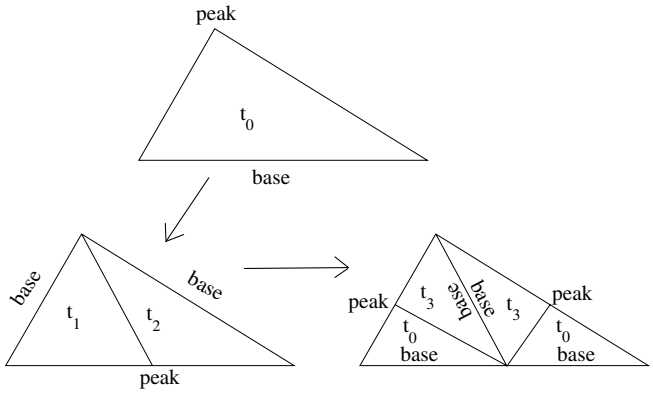

Fig. 2 Newest vertex bisection and propagation of the peak

It is easy to show that only four similarity classes of triangles and only eight distinct angles are created by this method, so the angles satisfy the important condition of being bounded away from 0 and π. Moreover, if this partition is applied an even number of times to an initial triangle, then, only two similarity classes of triangles are obtained, and half of the generated triangles are similar to the initial one.

2.2 The 4-triangles longest-edge partition

The 4T-LE partition was introduced and has been studied by Rivara [5] and Rivara and Iribarren [13] for the last 15 years.

Definition 4 The LE partition of a triangle t_0 is obtained by joining the midpoint of the longest edge of t_0 with the opposite vertex (Fig. 3a). The 4T-LE partition is obtained by joining the midpoint of the longest edge to the opposite vertex and to the midpoints of the two remaining edges (see Fig. 3b).

As the first 4T-LE partition of any triangle t_0 introduces two new edges parallel to the edges of t_0, the following result holds [13]:

Proposition 1 (a) The first 4T-LE partition of t_0 produces two triangles similar to t_0 and two (potentially) new similar triangles t_1. (b) The iterative 4T-LE partition of any triangle t_0 introduces (at most) one new dissimilar triangle per iteration.

Also in [13], it is demonstrated that the number of similarity classes generated by the 4T-LE partition is finite. It should be noted here, that although the Mitchell partition and the 4T-LE partition may yield the same sequence of nested meshes, this is certainly not the general case. As it has been proved in [13, 14] the 4T-LE

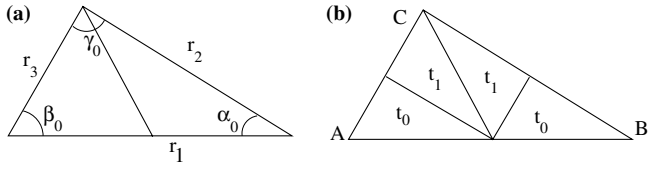

Fig. 3 **a** LE partition of triangle t_0, **b** 4T-LE partition of triangle t_0

partition shows a remarkable improvement property in the case of an initial obtuse triangle.

We focus in this paper on the fractal geometry shown by the triangular grids arising in the finite element calculations. These meshes are also related to some SFC as we shall demonstrate in this work.

3 The refinement and derefinement algorithms

3.1 The refinement algorithm

Frequently, refinement is classified into two types: *uniform* and *local*. Uniform refinement is said to be the refinement of all triangles in a mesh following a fixed subdivision pattern. If the refinement is made locally for only a triangle or a group of them, then the refinement is local. The local refinement of triangular meshes involves two main tasks. The first is the partition of the target triangles and the second is the propagation to successive neighbor triangles to preserve conformity. In order to ensure the conformity of the arising mesh, the refinement must be extended to additional triangles. This is done for the 4T-LE partition by using partial division patterns that are dependent on the bisected edges. These partition patterns always bisect the triangle at the midpoint of the longest-edge and then, if necessary, one or two of the resulting subtriangles are also bisected. To illustrate the 4T-LE refinement of a given triangle t in a given mesh, the next algorithm is provided.

4T-LE-refinement (τ, t_0)
/* Input variables: t_0, triangle to be refined, and τ, triangular mesh.
/* Output variables: new mesh τ
Perform the bisection of three edges of t_0
For each new point P generated **do**
{
While $P \neq$ the longest-edge midpoint of the neighbor triangle t^* **do**
{
Perform the bisection of the longest-edge of t^*
Let $P = Q$, where Q is the point generated
}
For each triangle t with bisected edges **do**
{
Subdivision of t
}
}

The local refinement algorithm works first on the *skeleton* of the 2D triangulation (the wireframe comprised of the edges of the triangulation). Then, the edge refinement propagates to successive neighbor triangles to preserve conformity, and finally it subdivides the interior of the triangles to obtain the new refined mesh. An efficient implementation of the previous algorithm using a graph-based data structure has been presented in Suárez et al. [15].

The example in Fig. 4 illustrates how the algorithm works: (a) triangle t is refined and then the edges of t are bisected; (b) the conformity process for non-conforming points are performed and (c) new triangle subdivisions are built.

3.2 The derefinement algorithm

The derefinement algorithm is the inverse counterpart of the refinement algorithm. Its goal is to obtain a new sequence of nested meshes, by eliminating the less significative vertices from an input sequence of meshes, following some *derefinement condition*. It should be noted that the algorithm itself does not depend on the derefinement condition. The derefinement condition usually involves the interpolated approximated solution of the extreme vertices of an edge and the approximated solution at the midpoint of the edge.

The input of the 4T-LE-derefinement algorithm is a sequence of k nested meshes $\{\tau_1, \tau_2, ..., \tau_k\}$. These meshes have been obtained, for example, by the 4T-LE refinement algorithm. During the inspection within a mesh level, a vertex is temporally marked to be removed if the absolute difference between the values in this vertex of the numerical solution and its corresponding interpolated function is less than a sufficiently small tolerance error $\epsilon > 0$. Note that the conformity is ensured locally by modifying the neighboring elements of the surrounding edge of each vertex v_j. After that process, the algorithm progressively reconstructs the k approximations from the coarsest level to the finest mesh. At the end, the result is a sequence of new meshes $\{\tau'_1, \tau'_2, ..., \tau'_k\}$.

Algorithm 4T-LE-derefinement (T, ϵ)
/* Input: $T = \{\tau_1, \tau_2, ..., \tau_k\}$, ϵ = tolerance
/* Output: $T' = \{\tau'_1, \tau'_2, ..., \tau'_k\}$
For $i = k$ to 2 **do**
{
For each vertex $v_j \in \tau_i$ **do**
{
Let a and b be the surrounding vertices of vertex v_j
If $\left|\frac{h(a)+h(b)}{2} - h(v_j)\right| < \varepsilon$ **then**
{
mark(v_j)
Check conformity
If v_j gets unmarked **then** unmark(a) & unmark(b)
}
}
}
For $j = 2$ to k **do**
{
Re-triangulate mesh τ_j
}

Figure 5 shows an example of derefinement of a sequence of four nested triangular meshes. Mesh level τ_3

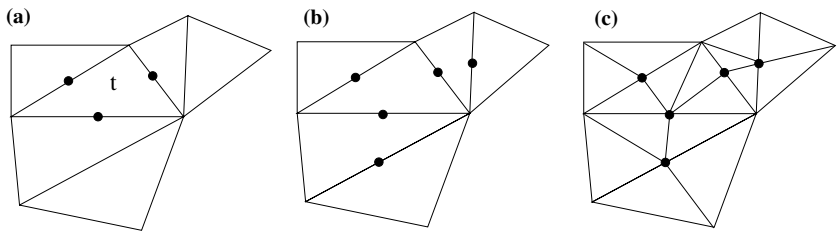

Fig. 4 **a** Edge bisection for refining triangle t, **b** edge subdivision in neighboring elements, **c** subdivision of involved triangles

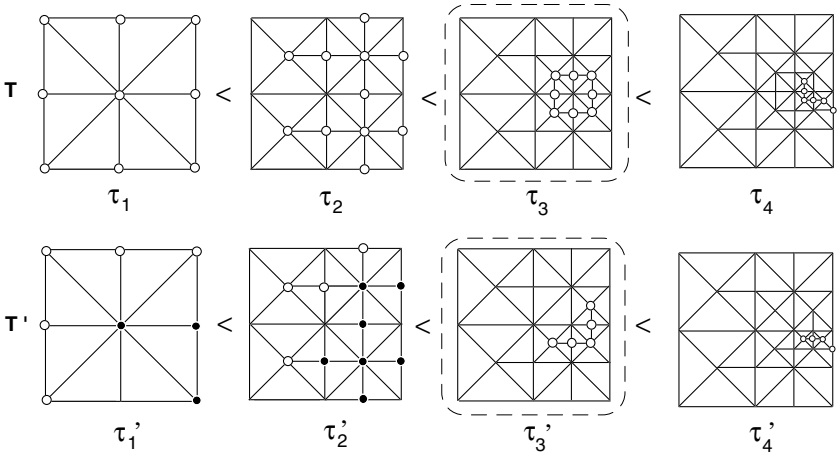

Fig. 5 Input and output sequences of nested meshes for the derefinement algorithm

and its derefined counterpart mesh τ'_3 are indicated with a dotted line. Note that the condition for some nodes to be maintained in the mesh (white nodes) implies directly that other nodes of the previous levels must also remain in the mesh (black nodes in mesh τ'_2). For details of the derefinement algorithm see [10, 15].

4 Fractality and iterated function systems

The fractal appearance of the adapted meshes commonly used in the FEM area has been explained in different ways. For instance, Rivara and Inostroza [16] define the concept of a stable molecule associated to a node P as follows:

Definition 5 For any conforming triangulation τ and any vertex P of τ, the stable molecule associated with vertex P is the partition of the plane around vertex P, induced by the use of the 4T-LE refinement of each triangle around the vertex P, such that further 4T-LE refinements of the triangles around P do not change the number of triangles sharing P.

Figure 6 illustrates the concept of a stable molecule associated to vertex P. Note that the number of angles converging in P are the same once the stable molecule of P has been achieved (Fig. 6b). Note that, after the stable molecule associated to vertex P has been achieved, further refinement around P does not change the shape of the triangles sharing P but only their size, with 1/2 as a scaling factor. Rivara and Inostroza [16, Theorem 2, p. 586] established the following result in the context of longest-side refinement, also valid for the 4T-LE refinement:

Theorem 1 Let τ be any conforming triangulation and consider any vertex P of τ. The use of the longest-side refinement algorithm to refine the triangulation around the vertex P produces triangulations that have the following characteristics:

(a) After a finite number of iterations, the algorithm produces a triangulation τ^* such that, the stable molecule associated with vertex P is obtained.
(b) The next iterations of the algorithm do not partition the angles of the stable molecule, but only introduce a set of new vertices distributed in geometric progression along the sides of the stable molecule of P.

The fractal behavior has also been explained via IFS in Plaza [11]. As is well-known, an IFS consists of a set of n contractive mappings, $W=\{w_i, i=1, \ldots, n\}$ defined on a metric space (X, d). Then the attractor associated to

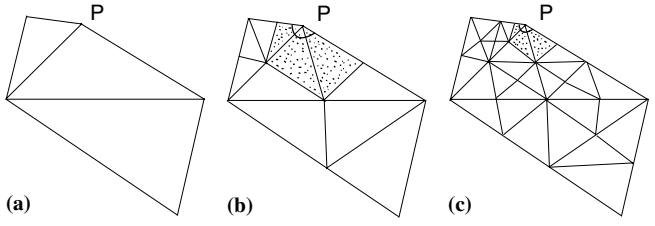

Fig. 6 4T-LE refinement around vertex P and the stable molecule

W, also called a *deterministic fractal* [1] is the unique fixed point A of a contractive mapping from metric space $(\mathscr{H}(X)), h(d))$ into itself, where $\mathscr{H}(X)$ is the set of nonempty compact subsets of X, and $h(d)$ is the Hausdorff metric associated with d. That is, for every pair of points $A, B \in \mathscr{H}(X)$ (A and B are compact subsects of X),

$$h(d)(A,B) = \max\left\{\max_{x\in A} d(x,B), \quad \max_{x\in B} d(A,x)\right\},$$

where $d(x,B)$ is defined by $d(x,B) = \min_{b\in B}\{d(x,b)\}$.

The set W induces a contractive mapping in $(\mathscr{H}(X), h(d))$, say $\mathscr{W}$, defined by

$$\mathscr{W}(B) = \bigcup_{i=1}^{n} w_i(B) \tag{1}$$

for all $B \in \mathscr{H}(X)$. In our case, $X = \mathbb{R}^{\not\vDash}$, $\mathscr{H}(\mathbb{X})$ is the space of nonempty compact subsets of $\mathbb{R}^{\not\vDash}$ and d is the Euclidean distance in the plane. So the attractor A satisfies $A = \mathscr{W}(A) = \bigcup_{i=1}^{n} w_i(A)$, and of course A is determined only by the set of mappings of the IFS, W, and not by the particular initial point $B_0 \in \mathscr{H}(X)$, because according to the fixed point theorem of complete metric spaces:

$$A = \mathscr{W}(A) = \lim_{k\to\infty}(B_k),$$

where $B_k = \mathscr{W}(B_{k-1}) = \bigcup_{i=1}^{n} w_i(B_{k-1})$.

In this way, to refine the reference triangle of Fig. 7 (a) toward vertex 1, we can apply the IFS given, in complex form, by $w(z) = (1/2)z$ and the initial input of Fig. 7b. The result is equivalent to applying global refinements followed by the derefinement procedure with vertex 1 as singularity. By iterating this process the resulting mesh (see Fig. 8) can be viewed as the iterate application of the contractive mapping to the previous mesh level:

$$B_0 \cup \mathscr{W}(B_0) \cup \mathscr{W}^2(B_0) \cup \cdots$$

Similarly, to refine the reference triangle toward the side connecting nodes 2 and 3, the corresponding IFS is given by the contractive maps $w_1(z) = (1/2)(z+i)$, $w_2(z) = (1/2)(z+1)$, and $w_3(z) = (-1/2)z + \sqrt{2}$ (see Fig. 9).

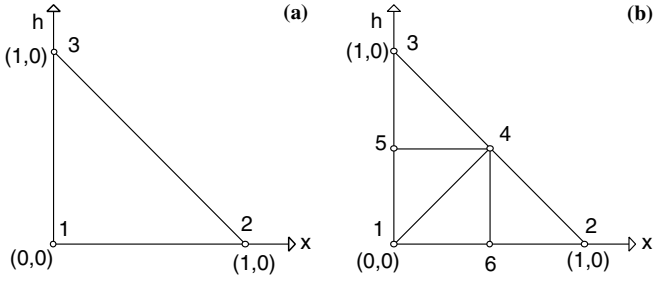

Fig. 7 **a** Reference triangle, **b** 4T-LE partition of the reference triangle

Finally, if the objective refined area is the whole initial triangle, the IFS to be applied to the initial triangle is given by $\{w_i;\ i=1,\ \ldots,\ 4\ \}$, where $w_1(z) = (1/2)(z+i)$, $w_2(z) = (1/2)(z+1)$, $w_3(z) = (-1/2)z + \sqrt{2}$, and $w_4(z) = (1/2)z$. In this case, mesh $\tau_k = \mathscr{W}^k(\tau_0)$ and the attractor given by the whole initial triangle is obtained as $A = \mathscr{W}(A) = \lim_{k\to\infty}(\tau_k)$.

5 Space-filling curves

By connecting the centers of the triangles of successive refined meshes in a specific order, in a natural way, the concept of SFC arises in the context of refined triangulations.

As noted by Sagan [12], Peano [17] discovered the first SFC, but Hilbert was the first who recognized a general geometrical generating procedure that allowed the construction of an entire class of SFC. Hilbert promulgated the following heuristic principle: If the interval $\mathscr{I}$ can be mapped continuously onto the square $\mathscr{Q}$, then after partitioning $\mathscr{I}$ into four congruent subintervals and $\mathscr{Q}$, into four congruent subsquares, each subinterval can be mapped continuously onto one of the subsquares. Next, each subinterval is, in turn, partitioned into four congruent subintervals and each subsquare into congruent subsquares, and the procedure is repeated. If this is carried on ad infinitum, $\mathscr{I}$ and $\mathscr{Q}$ are partitioned into 2^{2n} congruent replicas for $n = 1,2,3$ .... Hilbert demonstrated that the subsquares can be arranged so that adjacent subintervals correspond to adjacent subsquares with a common edge, and so that the inclusion rela-

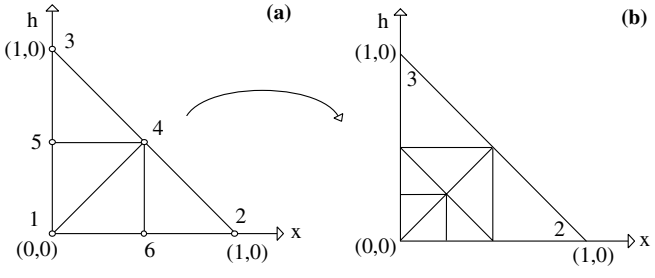

Fig. 8 Vertex 1 as attractor: **a** initial mesh $\tau_0 = B_0$, **b** mesh $\tau_1 = B_0 \cup \mathscr{W}(B_0)$

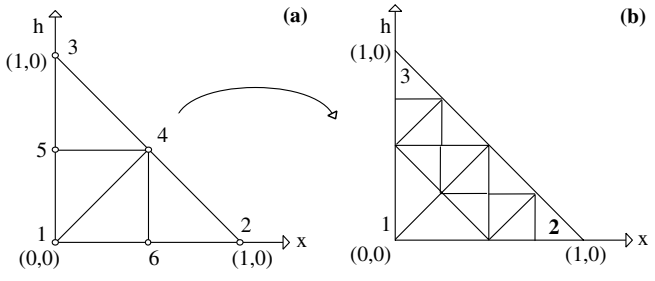

Fig. 9 Side 1–2 as attractor: **a** initial mesh $\tau_0 = B_0$, **b** mesh $\tau_1 = B_0 \cup \mathscr{W}(B_0)$

tionships are preserved, i.e., if a square corresponds to an interval, then its subsquares correspond to the subintervals of that interval. In Fig. 10, this process is shown for the first three steps. This geometric idea is in the background of some of the latest proposed data structures [18] in the area of digital terrain representations [19] and image compression [20].

Figure 11 shows the first steps in the generation of the so-called Sierpinski's SFC. This SFC has been used to create hierarchical generalized triangle strips [21] with applications in geometric compression and rendering.

Based on Evans et al. [19], Fig. 11a shows the possible orientations of the triangles appearing by iterative bisection from an initial isosceles right triangle. This orientation is compatible with the Sierpinski's SFC in the sense that by assigning the binary number to the center of the obtained triangles by successive bisection, the Sierpinski's SFC goes through all the points following the algebraic order of the numbers. The first steps in the formation of the Sierpinski's SFC for an initial triangle and the corresponding binary numbers of each triangle are shown in Fig. 11.

6 The 3D case

In this section, we briefly introduce some extensions of the previously presented ideas to the 3D case. However, there are some differences between the 2D and 3D spaces. For example, there is no tetrahedron whose edge bisection produces two tetrahedra similar to the former one. This fact is in clear contrast to the 2D case, in which all right triangles produce four similar right triangles when the 4T-LE partition is applied.

As it has already been noted, some 3D refinement algorithms and partitions have been studied in recent years [8, 9, 22, 24]. One of the approaches based on 3D bisection is that of Liu and Joe [22]. This partition can be understood as the 3D version of the Mitchell partition (see Sect. 2.1). The edges for bisection are chosen without any computation following a rule between the edge types involved and their relative position [22] to automatically assign the types to the new edges. The partition can also be explained by means of a mapping between the original tetrahedron T and a canonical tetrahedron P with the same volume as T [22]:

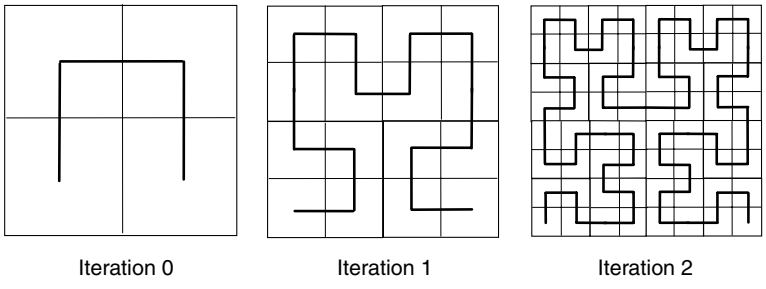

Fig. 10 First steps in the generation of Hilbert's SFC

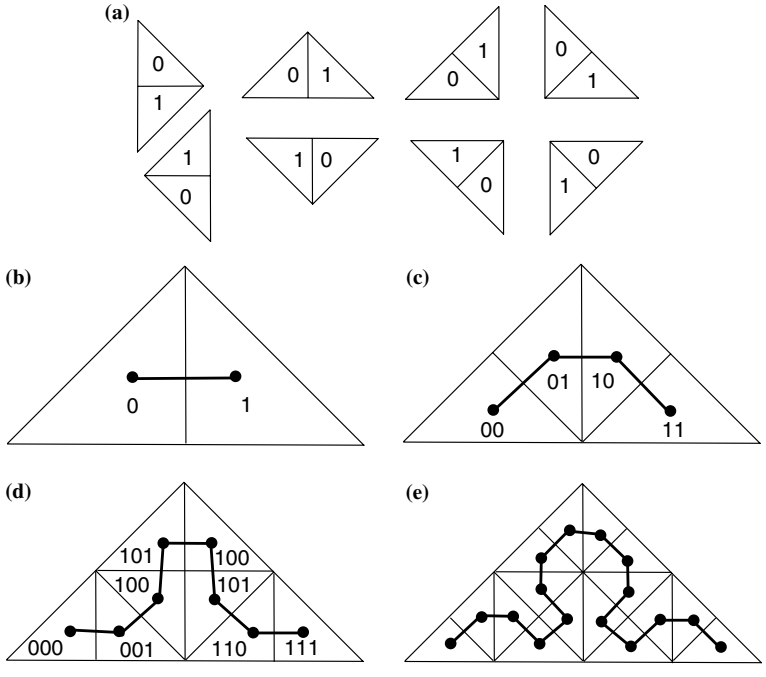

Fig. 11 Triangle orientations and their children, **b**–**e** Notation for the children and generation of the Sierpinski's SFC on a triangle

Definition 6 (Liu–Joe partition) Let T be any tetrahedron and let P be the special tetrahedron of Fig. 12 with the same volume as T. Then

(a) Transform T to P by the affine transformation $y = \mathbf{M}^{-1}(P,T)x + b_0$.
(b) Iteratively bisect P to three levels, always bisecting the longest edge.
(c) Transform the 8 subtetrahedra P_i of P back to subtetrahedra T_i of T using the inverse affine transformation $y = \mathbf{M}(P,T)x - \mathbf{M}(P,T)b_0$.

Figure 12 shows the partition in 8 tetrahedra of the canonical Liu–Joe tetrahedron. This tetrahedron has the interesting property in that its partition in 8 subtetrahedra yields 8 tetrahedra similar to the former one.

It should be noted that in the subtetrahedra of T, the longest edge may not be the one that is bisected, so this partition, in general, is not equivalent to three levels of longest-edge partition.

Recently, another partition in 8 tetrahedra based on the length of the edges, the 8-tetrahedra longest-edge (8T-LE) partition, has been investigated and used for refining and coarsening tetrahedral meshes [9, 10].

Definition 7 (8-tetrahedra longest-edge partition) For any tetrahedron t, the (8T-LE) of t produces 8 subtetrahedra by performing the 4T-LE partition of the faces of t, and by subdividing the interior of the tetrahedron t consistently with the division of the faces (see Figs. 3, 13).

The 8T-LE partition can be achieved by performing a sequence of bisections by the midpoints of the edges of the original tetrahedron, taking into account the length of the edges as follows [23]:

Theorem 2 For any tetrahedron t of unique longest edge, the (8T-LE partition) of t is obtained as follows:

1. Longest edge bisection of t producing tetrahedra t_1, t_2.
2. Bisection of t_i, for $i = 1,2$, by the longest edge of the common face of t_i with the original tetrahedron t, producing tetrahedra t_{ij}, for $j = 1,2$.
3. Bisection of each t_{ij} by the midpoint of the unique edge equal to an edge of the original tetrahedron.

Figure 13 shows the four different refinement patterns given by the relative positions of the longest edges, highlighted in bold, of the faces of the tetrahedron. It should be noted that the 8T-LE partition is the extension to 3D of the 4T-LE partition.

In the same way that in two dimensions the Mitchell partition, the longest-edge partition, and the 4T-LE partition are equivalent in some cases [7], in three dimensions, the longest-edge partition [24], the Liu–Joe partition [22] and the 8T-LE partition also yield an equal partition of 8 sub-tetrahedra. In this sense, it is worth noting that the 8T-LE partition, which in turn is equivalent to the Liu–Joe partition or even to the LE partition of the Liu–Joe tetrahedron can be explained by means of IFS with the following contractive mappings:

$$
\begin{aligned}
w_1(\mathbf{X}) &= \tfrac{1}{2}\mathbf{X}, \quad \text{for subtetrahedron } t_1 \text{ in Fig. 12,}\\
w_2(\mathbf{X}) &= \tfrac{1}{2}\mathbf{M}_2\mathbf{X} + \left(a, \tfrac{a}{\sqrt{2}}, 0\right),\\
&\quad \text{for subtetrahedron } t_2 \text{ in Fig. 12,}\\
w_3(\mathbf{X}) &= \tfrac{1}{2}\mathbf{M}_3\mathbf{X} + \left(\tfrac{a}{2}, 0, \tfrac{a}{2}\right), \quad \text{for subtetrahedron } t_3,\\
w_4(\mathbf{X}) &= \tfrac{1}{2}\mathbf{M}_4\mathbf{X} + (a, 0, a), \quad \text{for subtetrahedron } t_4,\\
w_5(\mathbf{X}) &= \tfrac{1}{2}\mathbf{M}_5\mathbf{X} + \left(\tfrac{3a}{2}, 0, \tfrac{a}{2}\right), \quad \text{for subtetrahedron } t_5,\\
w_6(\mathbf{X}) &= \tfrac{1}{2}\mathbf{M}_6\mathbf{X} + \left(a, \tfrac{a}{\sqrt{2}}, 0\right), \quad \text{for subtetrahedron } t_6,\\
w_7(\mathbf{X}) &= \tfrac{1}{2}\mathbf{M}_7\mathbf{X} + (0, 0, a), \quad \text{for subtetrahedron } t_7,\\
w_8(\mathbf{X}) &= \tfrac{1}{2}\mathbf{X} + (a, 0, 0), \quad \text{for subtetrahedron } t_8 \text{ in Fig. 12,}
\end{aligned}
$$

where M_i, for $i = 2, \ldots, 7$ are the corresponding orthogonal matrices:

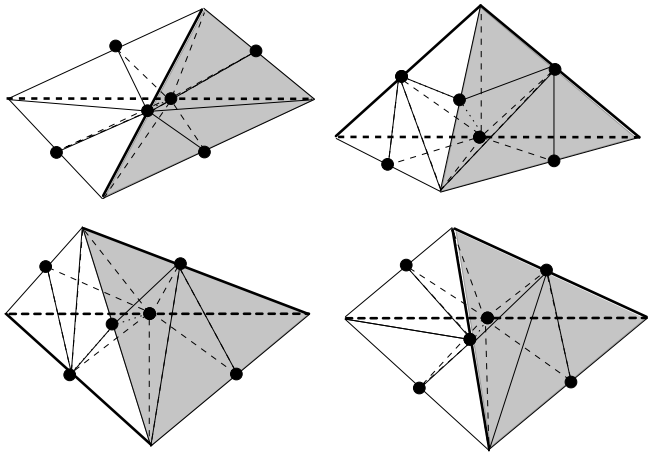

Fig. 13 Four different refinement patterns for the 8T-LE partition depending on the relative positions of the longest edges (*in bold*)

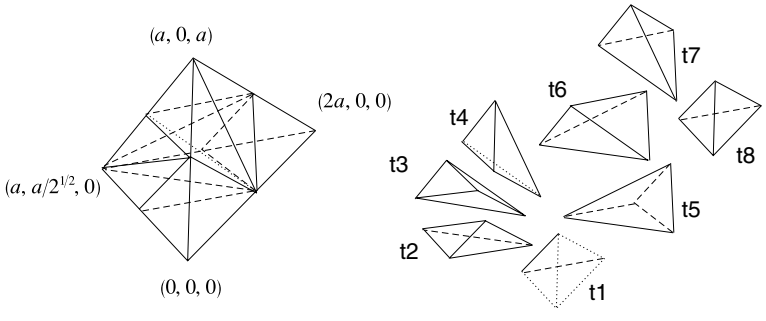

Fig. 12 Canonical Liu–Joe tetrahedron, and 8 subtetrahedra division

$$\mathbf{M_2} = \begin{pmatrix} \frac{-1}{2} & -\frac{\sqrt{2}}{2} & \frac{1}{2} \\ -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ \frac{1}{2} & -\frac{\sqrt{2}}{2} & -\frac{1}{2} \end{pmatrix}; \quad \mathbf{M_3} = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{2}}{2} & \frac{1}{2} \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ -\frac{1}{2} & \frac{\sqrt{2}}{2} & -\frac{1}{2} \end{pmatrix}$$

$$\mathbf{M_4} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}; \quad \mathbf{M_5} = \begin{pmatrix} -\frac{1}{2} & \frac{\sqrt{2}}{2} & -\frac{1}{2} \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{2}}{2} & -\frac{1}{2} \end{pmatrix}$$

$$\mathbf{M_6} = \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{2}}{2} & -\frac{1}{2} \\ -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ \frac{1}{2} & \frac{\sqrt{2}}{2} & -\frac{1}{2} \end{pmatrix}; \quad \mathbf{M_7} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

However, as is well known, the 3D refinement of a general tetrahedral mesh cannot be explained as the application of a set of contractive mappings, since the tetrahedra cannot be divided into a set of similar tetrahedra.

7 Numerical examples

Figure 14 shows two instances of the SFC associated to a local refined triangular grid.

Note that, for the (local) refinement, the curve goes through the centers of the triangles, and then we get a *unique* sequence of points associated to this (local) refined grid. So, by means of a suitable binary tree data structure, each possible local refined mesh may be associated with a unique binary tree or the corresponding unique sequence, given by the list of the numbers of the triangles in increasing order. For instance, following the notation for the children of Fig. 11a, to the meshes of Fig. 14, the two following sequences of binary numbers are associated, respectively: $\{0^4, 0^31, 0^210, 0^21^2, 010^3, 010^21, 010^2, 0101, 01^20^2, 01^201, 10^5, \ldots\}$, $\{0^4, 0^31, 0^210, 0^21^2, 010^2, 0101, 01^20, 01^20^2, 01^201, 10^4, \ldots\}$

It should be noted that in the previous notation the exponents are for the number of times the respective base appears. For example, 0^31 means 0001.

These integer sequences represent the triangles appearing in the mesh from the left to the right following the curve shown in Fig. 14. Note that the number of digits in each integer is equal to the depth of the binary tree, or the level of division of the sequence of meshes. Then, the structure of the binary tree allows us to calculate all the geometric information associated with a given triangular mesh with remarkable advantages in terms of memory usage and speed. For example, determining the coordinates of the three vertices of a triangle from its label is straightforward. The label describes a path in the binary tree representing the triangulation. This kind of data structure has been used for representing a digital terrain (height field) containing approximations of the terrain at different levels of detail by Evans et al. [19].

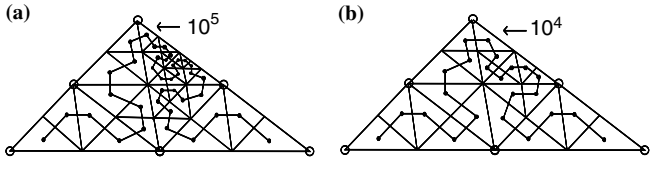

Fig. 14 Fractal geometry on **a** a local refinement and **b** derefinement. The notation for the upper right triangle is pointed out in each mesh

Space-filling curves have also been applied to the problem of automatic domain decomposition [25–27] both in 2D and 3D. Pilkington and Baden [25] describe a partitioning strategy called inverse spacefilling partitioning (ISP). ISP maps the higher dimensional space to a line, called a space-filling curve, and partitions the line into logically contiguous segments that correspond to physically irregular partitions. This idea has also been applied to a parallel *hp* adaptive finite element code [26].

Figure 15 shows a 3D L-shaped domain with the initial mesh. In this initial triangulation, some local tetrahedral refinements have been applied. We have used the software developed by Edwards [28] to obtain the 3D Hilbert SFC connecting the centers of the tetrahedra. This curve gives us an order into the set of tetrahedra. The partitioning of the domain into k subdomains is given by the corresponding partitioning of the SFC k subcurves.

Figure 16 shows the domain decomposition into two subdomains. Both subdomains present the same number of tetrahedral elements. The first subdomain corresponds to the first part of the SFC and the second one to the last part of the curve.

There are, however, some issues to be investigated; for example, what is the best SFC to be employed for the automatic domain decomposition, Hilbert type, Morton type, or another? Since, in general, the subdomains obtained by using the SFC are not connected, this leads us to ask for additional conditions for assuring the connectivity of the subdomains and for the structure of the interface between the subdomains, which are of interest in the context of the parallel implementation of FEMs.

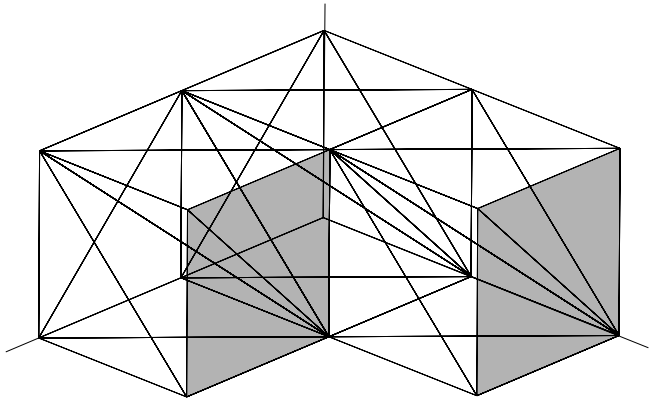

Fig. 15 L-shaped domain with the initial tetrahedral mesh

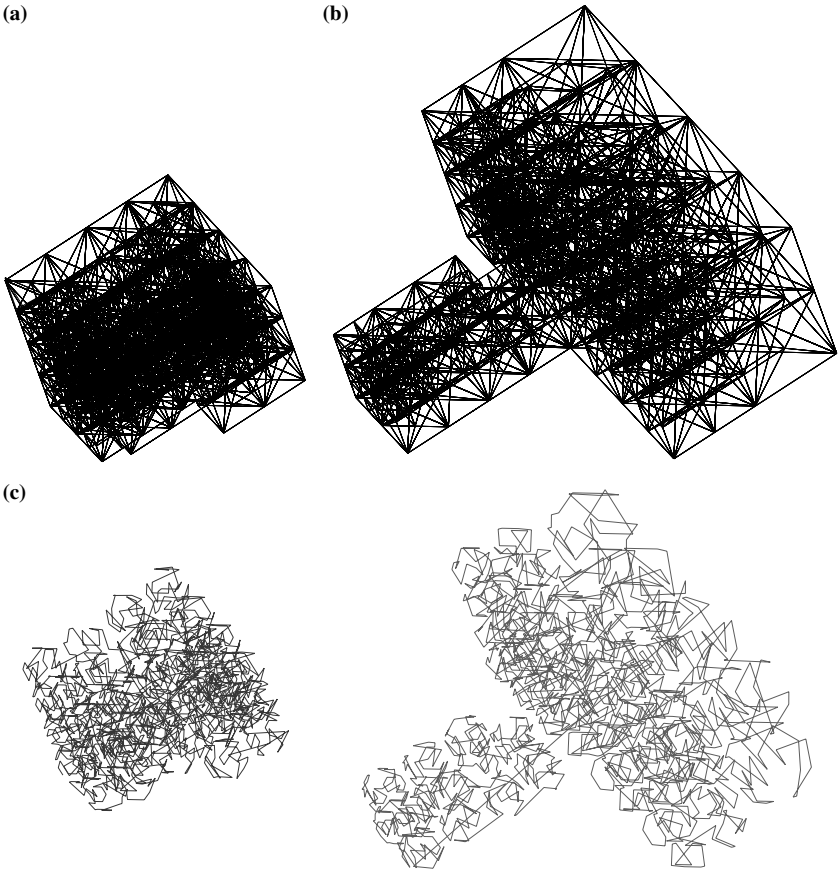

Fig. 16 Domain decomposition of L-shaped domain of Fig. 15 and associated SFC

8 Conclusions

Although using the concepts of IFS to derive a set of intersection-based refinement rules is not common in the area of FEMs, they are equivalent to using a combination of refinements and derefinements.

In this paper, we explained the fractal geometry of refined and derefined triangular and tetrahedral meshes by applying IFS. The application of IFS families is equivalent to the use of adaptive strategies that combine the refinement procedure with the derefinement one.

In addition, SFC were used to assign a binary code for any 2D triangular refined mesh. It was also demonstrated that SFC are useful for the problem of automatic domain decomposition.

Acknowledgements This work has been partially supported by Project UNI-2003/16 of the University of Las Palmas de Gran Canaria and by Project PI2003/35 of the Gobierno de Canarias.

References

1. Barnsley MF (1991) Fractals everywhere. Academic, New York
2. Gutiérrez JM, Iglesias A, Rodríguez MA (1996) A multifractal analysis of IFSP invariant measures with application to fractal image generation. Fractals 4(1):17–27
3. Wu Z-B (2003) Self-similarity limits of genomic signatures. Fractals 11(1):19–25
4. Falconer KJ (2003) Fractal geometry: mathematical foundations and applications, 2nd edn. Wiley, New York
5. Rivara M-C (1984) Mesh refinement based on the generalized bisection of simplices. SIAM J Numer Anal 2:604–613
6. Rivara M-C (1989) Selective refinement/derefinement algorithm for sequences of nested triangulations. Int J Numer Meth Eng 28:2889–2906
7. Mitchell WF (1992) Optimal multilevel iterative methods for adaptive grids. SIAM J Sci Stat Comp 13:146–167
8. Bänsch E (1991) Local mesh refinement in 2 and 3 dimensions. IMPACT Comp Sci Eng 3:181–191
9. Plaza A, Carey GF (2000) Local refinement of simplicial grids based on the skeleton. Appl Numer Math 32(2):195–218
10. Plaza A, Padrón MA, Carey GF (2000) A 3d refinement derefinement combination for solving evolution problems. Appl Numer Math 32(4):401–418
11. Plaza A (1996) The fractal behavior of triangular refined/derefined meshes. Commun Num Meth Eng 12:295–302
12. Sagan H (1994) Space-filling curves. Springer, Berlin Heidelberg New York
13. Rivara M-C, Iribarren G (1996) The 4-triangles longest-side partition of triangles and linear refinement algorithms. Math Comput 65(216):1485–1502
14. Plaza A, Suárez JP, Padrón MA, Falcón S, Amieiro D (2004) Mesh quality improvement and other properties in the four-triangles longest-edge partition. Comput Aided Geom Des 21(4):353–369
15. Suárez JP, Carey GF, Plaza A (2001) Graph-based data structures for skeleton based refinement algorithms. Commun Numer Methods Eng 17(12):903–910
16. Rivara M-C, Inostroza P (1997) Using longest-side bisection techniques for the automatic refinement of Delaunay triangulations. Int J Numer Methods Eng 40:581–597
17. Peano G (1896) Sur une curbe qui remplit toute une aire plane. Math Ann 36:157–160

18. Pajarola R, Antonijuan M, Lario R (2002) QuadTIN: quadtree based triangulated irregular networks. In: Proceedings of IEEE Visualization, pp 395–402
19. Evans W, Kirkpatrick D, Townwsend G (2001) Right-triangulated irregular networks. Algorithmica (Special Issue on Algorithms for Geographical Information) 30(2):264–286
20. Pajarola R, Widmayer P (2000) An image compression method for spatial search. IEEE Trans Image Process 9(3):357–365
21. Velho L, Figueiredo de LH, Gomes J (1999) Hierarchical generalized triangle Strips. Vis Comput 15(1):21–35
22. Liu A, Joe B (1995) Quality local refinement of tetrahedral meshes based on bisection. SIAM J Sci Stat Comput 16:1269–1291
23. Plaza A, Rivara M-C (2002) Asymptotic behavior of the average adjacencies for skeleton-regular triangular and tetrahedral partitions. J Comput Appl Math 140(1–2):673–693
24. Rivara M-C, Levin C (1992) A 3-D refinement algorithm siutable for adaptive and multigrid techniques. Commun Appl Numer Methods 8:281–290
25. Pilkington JR, Baden SB (1996) Dynamic partitioning of non-uniform structured workloads with spacefilling curves. IEEE Trans Parallel Distrib Syst 7(3):288–300
26. Edwards HC, Browne JC (1996) Scalable distributed dynamic array and its application to a parallel hp adaptive finite element code. In: TICAM, Texas Institute for Computational and Applied Mathematics, The University of Texas
27. Iqbal S, Carey GF (2002) Neural nets for mesh assessment. TICAM Report 02-02
28. Edwards HC (2001) Zoltan software, Sandia National Laboratories. http://www.cs.sandia.gov/Zoltan/ug_html/ug_alg_hsfc.html