The J_1^a Triangulation: an adaptive triangulation in any dimension

A. Castelo[1], L.G. Nonato[1], M.F. Siqueira[2], R. Minghim[1] and G. Tavares[3]

[1] Instituto de Ciências Matemáticas e de Computação
Departamento de Computação e Estatística
Universidade de São Paulo - USP - São Carlos
{castelo,gnonato,rminghim}@icmc.usp.br

[2] Departamento de Computação e Estatística
Universidade Federal do Mato Grosso do Sul - Campo Grande
marcelos@seas.upenn.edu

[3] Departamento de Matemática
Pontifícia Universidade Católica do Rio de Janeiro - Rio de Janeiro
tavares@mat.puc-rio.br

June 7, 2004

Abstract

Spatial sampling methods have acquired great popularity due to the number of applications that need to triangulate portions of space in various dimensions. One limitation of the current techniques is the handling of the final models, which are large, complex and need to register neighborhood relationships explicitly. Additionally, most techniques are limited to Euclidean bidimensional or tridimensional spaces and many do not handle well adaptive refinement. This work presents a novel method for spatial decomposition based on simplicial meshes (the J_1^a triangulation) that is generally defined for Euclidean spaces of any dimension and is intrinsically adaptive. Additionally it offers algebraic mechanisms for the decomposition itself and for definition of neighbrs that allow to recover all the information on the resulting mesh via a set of rules. This way it is possible to balance the cost of storage and manipulation by calculating the needed information instead of storing it. Results additionally show good quality meshes with efficient calculation.

1 Introduction

The iso-surface extraction problem has achieved large popularity over the last two decades. The interest in this subject is motivated by the large number of applications that rely on iso-surfaces as a mechanism to investigate surface reconstruction, solid modelling, and visualization from images.

Different strategies have been proposed to generate iso-surfaces from volumetric data, which can be grouped in three main categories [1]: surface fitting, surface tracking, and spatial sampling. Surface fitting methods aim at progressively adapting an initial approximation to the desired iso-surface [8, 10]. Surface tracking techniques make use of continuation methods to approximate the iso-surface from a seed [13]. The most popular iso-surface extraction methods belong to the spatial sampling category, which generate an approximation of the iso-surface by decomposing the domain in a finite number of cells [14].

One of the most famous spatial sampling method is marching cubes [14], which works on a regular hexahedral grid decomposition of a region of interest and builds a triangulation by computing the intersection between the grid and the iso-surface of interest. Although very popular, the marching cubes algorithm presents some of the intrinsic problems of spatial sampling techniques, particularly ambiguity in the triangulation and large number of triangles.

Ambiguity appears because certain configurations of cubes allow more than one choice of triangulation, producing holes in the final surface. Several methods have been proposed to overcome the ambiguity problem [17], as for example to decompose the hexahedral cells in tetrahedra [12]. The main problem with tetrahedral decomposition is the even larger number of triangles produced [9].

The number of triangles is usually not critical for surface fitting and surface tracking techniques, as they handle this problem through adaptive mechanisms [11, 1]. Adaptive schemes allow to keep the density of the mesh under control. However, they are not so common in spatial sampling methods. Memory requirements, the data structures involved and their complex implementation are some reasons for avoiding adaptive schemes in spatial sampling methods.

In this paper we introduce a new adaptive spatial sampling method, called the J_1^a triangulation. Based on an algebraic framework, the J_1^a handles an adaptive tetrahedral decomposition in any euclidean space, with low cost of memory and very simple data structures.

Another important contribution of this work is the criterion used to adapt the triangulation. Our approach provides a refinement mechanism based on submanifolds, i.e., the piecewise linear manifold that approximates an implicity manifold can be refined around implicitly defined submanifolds. The methodology proposed here is efficient for modelling objects obtained by user-defined mathematical functions as well as for the reconstruction from real data.

This work is organized as follows: Section 2 presents a brief description of related works. In Section 3 we introduce some basic concepts necessary to understand the subsequent sections. Section 4 describes the J_1^a triangulation and its properties. The adaptive mechanism is presented in Section 5. Some computational results that illustrate the applicability of the proposed framework are shown in Section 6. Finally, conclusions are presented in Section 7.

2 Related Works

In this section we present previous work on spatial decomposition. For each work we emphasize the strategy used for decomposition, the adaptive mechanisms applied (when present) and how the neighborhood relationship among cells is obtained by them.

In general, iso-surface extraction based on sampling methods employ cubes or tetrahedrons as basic cells and evaluate the sign of a potential field at the vertices of the cells to define a configuration from which the polygonal approximation of the iso-surface is produced [4].

Spatial sampling methods that utilize tetrahedrons generally start from a cubic grid and generate a tessellation of the space either by decomposing each cube in a set of tetrahedrons as in the work by Guéziec and Hummel [12] or by constructing the tetrahedral tessellation from the vertices contained in the cubic grid as presented in other works [21, 7]. An important fact that deserves attention is that the quality of the tetrahedral tessellation of the domain depends on the type of decomposition employed, affecting the polygonization for the target surface.

As pointed out by Chan and Purisima [7], the type of tetrahedral decomposition named body-centered, which has the center of a cubic unit as a vertex of the tetrahedral mesh, has advantages when compared to other types of decompositions, such as the classic cubic subdivision in tetrahedrons (vertices of tetrahedrons are the vertices of the unit cube) and the face-centered subdivision (new vertices are added in the center of each face of the unit cube). A detailed description of these decompositions can be found in the book by Mckie and Mckie [15]. The main advantages of the body-centered decomposition are its degree of regularity and the good ratio between the lengths of the longer edges and the lengths of shorter edges. That feature imposes a better quality for the body-centered tessellation, resulting in a more appropriate polygonal approximation of the iso-surface. These nice properties have motivated the use of body-centered decomposition in several recent works [21, 7].

As already mentioned in the introduction, an adaptive mechanism can be a powerful tool to reduce the number of elements in the polygonal approximation of an iso-surface. It is not very common for strategies based on sampling methods to employ adaptive schemes to extract iso-surfaces. One of the main reasons for this is the fact that it can be difficult to combine elements in different levels of refinement without introducing cracks [20]. Another reason that discourages the use of adaptive schemes with sampling methods is the difficulty to control the neighborhood relationship among elements, an essential feature to allow efficient traversal through the tessellation [2].

In general, the combination of elements in different levels of refinement is done by imposing that adjacent elements stay in consecutive refinement levels [23], i.e., if an element of the decomposition is at level i then its neighbors are at levels $i-1$, i or $i+1$ (this notation only makes sense in decomposition from cubic grids). This restriction on the level of refinement of the neighbor elements allows that the decomposition be done in a more effective way, as shown in various works [16, 5].

The literature has presented several approaches to deal with the neighborhood relationship of elements in a tetrahedral decomposition, namely: labelling schemes [12, 21], topological data structures [2], and spatial partition trees [16]. The last two are commonly employed in adaptive tetrahedral decomposition of complex domains, for instance, those utilized in finite elements problems. In handling regular domains, labelling schemes are more attractive, since they offer a traversal mechanism that saves memory while allowing efficient access to neighboring elements. The main problem with labelling schemes is that they are not easily conceived to adaptive decompositions. This is a shortcoming handled efficiently by the work presented here.

The J_1^a triangulation presented in this paper is an adaptive domain decomposition that starts from an adaptive cubic grid and generates a simplicial tessellation of the domain, which is a hybrid of the body-centered and face-centered decompositions. This hybrid approach produces a decomposition with good quality in any euclidean space $\mathbb{R}^m$, which can be employed in a variety of applications, from implicit manifold reconstruction to extraction of iso-surfaces from images. An important aspect of the J_1^a triangulation is that its simplices can be defined through an algebraic mechanism, resulting in very large memory savings. Besides this algebraic mechanism to specify the simplices, the J_1^a triangulation offers a labelling scheme for traversing and accessing, in any dimension, the neighborhood relationship of any simplex contained in the decomposition. As opposed to other approaches that handle multiple dimensions ([3, 18]), the adptive mechanism intrinsic to J_1^a is applicable to any dimension. Therefore, the J_1^a triangulation comprises the good properties of adaptive tetrahedral decompositions with the advantage of being valid in euclidean spaces of any dimension.

3 Basic Concepts

This section introduces the basic concepts and terminology used in the remaining of the text. Definitions and results presented in this and the following sections are restricted to m-dimensional Euclidean spaces.

Let $S = \{v_0, v_1, \ldots, v_n\}$ be a set of points in $\mathbb{R}^m$. The set $c = \{v \in \mathbb{R}^m; v = \sum_{i=0}^{n} \lambda_i v_i, \sum_{i=0}^{n} \lambda_i = 1 \text{ e } \lambda_i \geq 0\}$ is called a **convex cell** generated by S. The **dimension** of c, denoted $dim(c)$, is the number of linearly independent vectors in the set $\{v_0 - v_1, \ldots, v_0 - v_n\}$.

A **cell decomposition** of $K \subset \mathbb{R}^m$ it is a finite collection $\mathcal{C}$ of convex cells satisfying:

1. $K = \cup_{c \in \mathcal{C}} c$;
2. if c_1, $c_2 \in \mathcal{C}$, then either $c_1 \cap c_2 = \varnothing$ or $c_1 \cap c_2 \in \mathcal{C}$.

A cell decomposition $\mathcal{C}$ of $K \subset \mathbb{R}^m$ is a n**-dimensional piecewise linear manifold** (n-dimensional PL-manifold) if each vertex v_i (0-dimensional cell), $\bigcup\limits_{c_i \supset v_i} c_i$ is homeomorphic to an n-dimensional sphere, where c_i are the n-dimensional cells containing v_i. From such a definition we have that each $(n-1)$-dimensional cell in $\mathcal{C}$ is contained in one or two n-dimensional cells of $\mathcal{C}$.

The **boundary** of a n-dimensional PL-manifold $\mathcal{C}$ is the set of $(n-1)$-dimensional cells that are contained in only one n-dimensional cell of $\mathcal{C}$.

A k**-simplex** in $\mathbb{R}^m$ is a convex cell of dimension k generated by $k+1$ points $S = \{v_0, \ldots, v_k\}$ of $\mathbb{R}^m$. Each subset of S generates an l-simplex that is called an l-face (or face for short) of the original simplex. In particular, 0-simplices are called vertices, 1-simplices are edges, 2-simplices triangles, and 3-simplices tetrahedra. A **triangulation** of a set $K \subset \mathbb{R}^m$ is a cell decomposition T where each cell of T is a simplex. If T is a PL-manifold then it is called a **triangulated PL-manifold**.

Each point v contained in a m-simplex $\sigma = [v_0, \ldots, v_m] \in T$ can be uniquely written by $v = \sum\limits_{i=0}^{m} \lambda_i v_i$, where $\sum\limits_{i=0}^{m} \lambda_i = 1$ and $\lambda_i \geq 0$, $i = 0, \ldots, m$. That way, given a map $F : K \subset \mathbb{R}^m \to \mathbb{R}^n$ and a triangulation T of K, we can define the affine map $F_\sigma : \sigma \to I\!R^n$ as follows:

$$F_\sigma(v) = F_\sigma(\sum_{i=0}^{m} \lambda_i v_i) = \sum_{i=0}^{m} \lambda_i F(v_i)$$

where $\sigma = [v_0, \ldots, v_m] \in T$. It is easy to see that F_σ is a linear interpolation of the values of F in the vertices of σ.

¿From the above, we can define a *piecewise linear approximation* (PL-approximation)

$F_T : K \subset \mathbb{R}^m \rightarrow \mathbb{R}^n$ for F as follows:

$$F_T(v) = F_\sigma(v); \quad v \in \sigma \in T$$

That way, F_T is a linear interpolation of the values of F in the vertices in T.

4 J_1^a Triangulation

In this section we present the J_1^a triangulation and the algebraic mechanism that allows the implicit representation and traversing of its simplices. In order to improve understanding, we try to outline the main ideas through examples in two or three dimensions.

4.1 Description

Let $\mathbb{I} = \{I_1, ..., I_m\}$ be a set of m closed intervals in $\mathbb{R}$, i.e., $I_i = [a_i, b_i]$. A **hypercube** $D^m \subset \mathbb{R}^m$ is defined as the product of the intervals in $\mathbb{I}$, i.e., $D^m = I_1 \times \ldots \times I_m$.

A **k-dimensional face** f of D^m, $k < m$ is the product of a subset of intervals $S \subset \mathbb{I}$ with the ends of the interval in $\mathbb{I} - S$, i.e., $f = I_{i_1} \times \cdots \times I_{i_k} \times \alpha_{j_1} \times \cdots \times \alpha_{j_{m-k}}$, where $\{i_1, \ldots, i_k\} \cap \{j_1, \ldots, j_{m-k}\} = \varnothing$, $\{i_1, \ldots, i_k\} \cup \{j_1, \ldots, j_{m-k}\} = \{1, 2, \ldots, m\}$, and either $\alpha_{j_s} = a_{j_s}$ or $\alpha_{j_s} = b_{j_s}$.

For instance, in $\mathbb{R}^3$ a hypercube is a cube, such as the one in Figure 1, where $I_1 = [a_1, b_1], I_2 = [a_2, b_2]$ and $I_3 = [a_3, b_3]$. Figure 1 also presents examples of $2-$, $1-$, and $0-$dimensional faces. For the $2-$dimensional face f_1 in Figure 1, $f_1 = [a_1, b_1] \times [a_3, b_3] \times b_2$. In this case $i_1 = 1, i_2 = 3$ and $j_1 = 2$. Here we have $\{i_1, i_2\} \cap \{j_1\} = \varnothing$, $\{i_1, i_2\} \cup \{j_1\} = \{1, 2, 3\}$ For the $1-$dimensional face f_2 in Figure 1, $f_2 = [a_2, b_2] \times b_1 \times b_3$ and $i_1 = 2, j_1 = 1$ and $j_2 = 3$. Here we have $\{i_1\} \cap \{j_1, j_2\} = \varnothing$, $\{i_1\} \cup \{j_1, j_2\} = \{1, 2, 3\}$ For the $0-$dimensional face f_3 in Figure 1, $f_3 = a_1 \times a_2 \times b_3$ and $j_1 = 1, j_2 = 2$ and $j_3 = 3$. Here we have, $\{j_1, j_2, j_3\} = \{1, 2, 3\}$

A **refinement** of D^m is a decomposition $D^m = D_1^m \cup D_2^m \cup \ldots \cup D_{2^m}^m$ in 2^m identical hypercubes such that $D_i^m \cap D_j^m = f$, where f is either empty or a common face of D_i^m and D_j^m, $j \neq i$. Note that the refinement can be seen as a recursive process, i.e., each D_i^m can also be refined. The initial hypercube is called a **0-block** and a hypercube generated by the refinement of a **i-block** is called **$(i+1)$-block**. A set of blocks generated from this recursive refinement scheme is called a **grid** in $\mathbb{R}^m$, denoted $\mathcal{G}$.

In order to define the adaptive scheme of the J_1^a triangulation correctly, it is necessary to restrict the blocks of a grid $\mathcal{G}$ in such way that every i-block, $i \geq 0$, of $\mathcal{G}$ is adjacent to only $(i-1), i$ and $(i+1)$-blocks (and 0-blocks are only adjacent to 0 and 1-blocks of $\mathcal{G}$). Therefore, the refinement of a i-block may imply in the refinement of neighbor blocks in

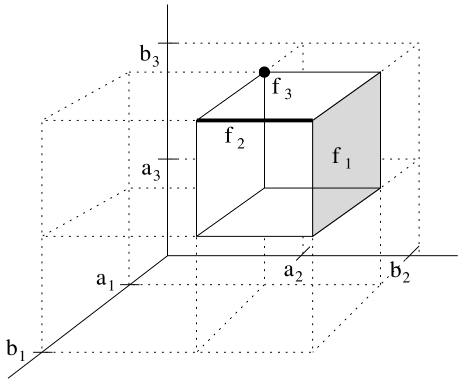


Figure 1: Examples of $k-$dimensional faces in $\mathbb{R}^3$

order to satisfy this restriction. In the remaining of the text we assume that all grids in satisfy the restriction.

If an i-block is not adjacent to $(i+1)$-blocks then it is called **basic block**. Otherwise, it is called **transition block**. Transition blocks are responsible for the connection among basic blocks in different levels of refinement. Figure 2 shows a two-dimensional grid containing blocks in three levels of refinement.

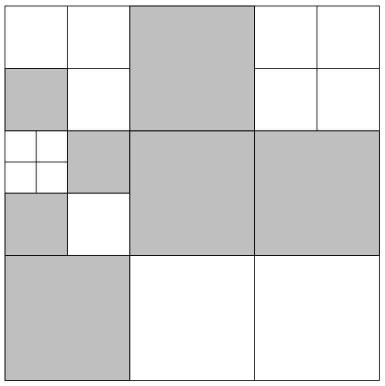


Figure 2: Basic (white) and trasition (gray) blocks in a Level 2 refinement of grid.

A particularly important hypercube in $\mathbb{R}^m$, called **standard hypercube**, is generated by the product of intervals centered in the origin with length 2, i.e., $D^m = [-1,1] \times [-1,1] \times \cdots \times [-1,1]$. This kind of hypercube is important because all other hypercubes in $\mathbb{R}^m$ can be obtained from it by translation and scaling and its faces can be labeled in a straight way: if the coordinate axis of $\mathbb{R}^m$ are denoted by $\{1, 2, \ldots, m\}$, then an $m-1$-face of D^m is marked with the label i, $i = 1, \ldots, m$ if it intersects the positive part of the i^{th} axis. Otherwise, if it intersects the negative part of the i^{th} axis, it is labeled $-i$. The label of the

$(m-k)$-faces, $1 < k \leq m$, are derived from the labels of the $(m-1)$-faces in accordance with their intersections. For example, in Figure 3, the highlighted vertex and edge are labeled $\{-2, 1, 3\}$ and $\{1, 2\}$ respectively as they are generated by the intersection of the 2-faces $\{-2\}, \{1\}, \{3\}$ and $\{1\}, \{2\}$ respectively. Note that the label of a i-face is a set of numbers without ordering, i.e., the label $\{1, 2\}$ can also be represented by $\{2, 1\}$.

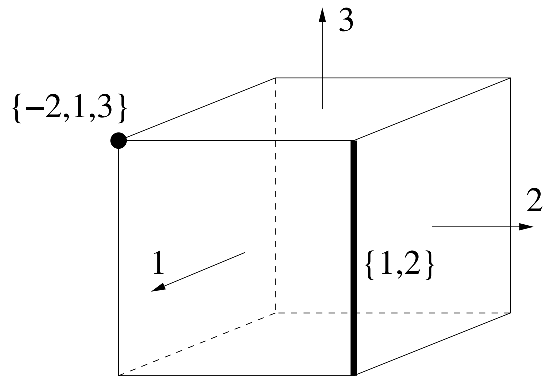


Figure 3: Labeling the faces of a standard hypercube.

A standard hypercube D^m in $\mathbb{R}^m$ can be decomposed in a set of simplices through an algebraic process that encapsulates their geometrical information. This decomposition is produced by combining, in an appropriate way, the vectors of the canonical basis. In order to describe this combination, suppose that $S^m = \{s_1, \ldots, s_{2^m}\}$ is the set of all m-dimensional vectors in $\mathbb{R}^m$ whose components are 1 or -1 and $B^m = \{e_1, \ldots, e_m\}$ is the canonical basis of $\mathbb{R}^m$. Also suppose that $\Pi^m = \{\pi_1, \ldots, \pi_{m!}\}$ is the set of all m-dimensional vectors generated by permuting $(1, 2, \ldots, m)$. For example, in $\mathbb{R}^2$, $S^2 = \{(1,1), (-1,1), (1,-1), (-1,-1)\}$, $B^2 = \{(1,0), (0,1)\}$ and $\Pi^2 = \{(1,2), (2,1)\}$. Each $s_i \in S^m$ and $\pi_j \in \Pi^m$ gives rise to a m-simplex contained in D^m whose vertices $[v_0, \ldots, v_m]$ are obtained as follows:

$$\begin{aligned} v_0 &= (0, \ldots, 0) \\ v_1 &= v_0 + s_i^{\pi_j^1} e_{\pi_j^1} \\ \ldots & \\ v_m &= v_{m-1} + s_i^{\pi_j^m} e_{\pi_j^m} \end{aligned} \tag{1}$$

where $s_i^{\pi_j^k}$ is the $\pi_j^{k\,th}$ component of s_i and π_j^k is the k^{th} component of $\pi_j \in \Pi^m$. For example, if $\pi_j = (\pi_j^1, \pi_j^2) = (1, 2)$ and $s_i = (-1, 1)$ then $s_i^{\pi_j^1} = s_i^1 = -1$ and $s_i^{\pi_j^2} = s_i^2 = 1$

In order to illustrate the construction above, suppose $D^2 = [-1,1] \times [-1,1]$, $S^2 = \{(1,1),(-1,1),(1,-1),(-1,-1)\}$, $\pi_1 = (1,2)$, and $B^2 = \{(1,0),(0,1)\}$. If $s_1 = (1,1)$ then a 2-simplex in D^2 can be defined by the vertices $v_0 = (0,0)$, $v_1 = v_0 + 1.(1,0) = (1,0)$, and $v_2 = v1 + 1.(0,1) = (1,1)$, as shows Figure 4a). A permutation of the vectors in B^2 generates the 2-simplex given by the vertices $v_0 = (0,0)$, $v_1 = v_0 + 1.(0,1) = (0,1)$, and $v_2 = v_1 + 1.(1,0) = (1,1)$ as in Figure 4b). It is not difficult to see that each $s_i \in S^2$ jointly with all permutations of B^2 (given by Π^2) generates the simplices that triangulate a quadrant of D^2.

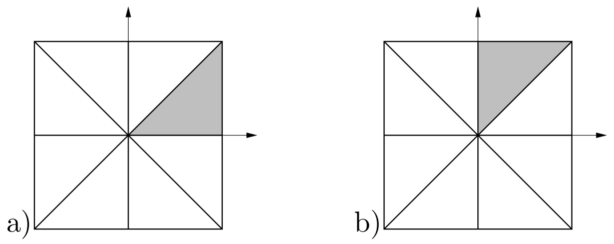


Figure 4: Simplices generated from a) $s_1 = (1,1)$ and $\pi_1 = (1,2)$; b) $s_1 = (1,1)$ and $\pi_2 = (2,1)$.

The same construction can be used to generate a triangulation of $D^m \subset \mathbb{R}^m$. Particularly in $\mathbb{R}^3$, if $D^3 = [-1,1]\times[-1,1]\times[-1,1]$, $S^3 = \{(1,1,1),(-1,1,1),(1,-1,1),\ldots,(-1,-1,-1)\}$, $\Pi^3 = \{(1,2,3),(1,3,2),\ldots,(3,2,1)\}$ and $B^3 = \{(1,0,0),(0,1,0),(0,0,1)\}$, each $s_i \in S^3$ jointly with all elements $\pi_j \in \Pi^3$ generate a triangulation of an octant of D^3.

It is worth noting that the algebraic definition of the simplices presented above is capable of encoding all geometrical information and it allowing traversal of the simplices without their explicit representation. In the J_1^a triangulation, all the basic blocks of a grid $\mathcal{G}$ are triangulated in the same way as the scheme exemplified above.

In order to complete the definition of the J_1^a triangulation from a grid $\mathcal{G}$ it is necessary to show how to triangulate the transition blocks of $\mathcal{G}$. Suppose that D^m is a transition i-block in a grid $\mathcal{G}$, i.e., some faces of D^m, called **refined faces**, are generated by the union of faces contained in adjacent $(i+1)$-blocks. Figure 5a) shows an example of a transition block in $\mathbb{R}^3$ with a two- and a one-dimensional refined face.

Let $\Phi = \{\phi_1,\ldots,\phi_k\}$ be the set of labels of the refined faces of D^m. In Figure 5a), $\Phi = \{\{3\},\{1,2\}\}$. The main problem with the triangulation of a transition block is that the simplicial decomposition of its refined faces demand a special attention. Before introducing the algebraic mechanism that produces the vertices of the simplices in a transition block, we present an important construction that will be essential in the definition of such vertices.

Consider $s_i \in S^m$ and $\pi_j \in \Pi^m$ a permutation of $(1,2,\ldots,m)$. Let h be the smallest integer such that there is a $\phi_k \in \Phi$ where $\phi_k \subset \{s_i^1\pi_j^1,\ldots,s_i^h\pi_j^h\}$ ($h = m$ if there is no $\phi_k \subset \{s_i^1\pi_j^1,\ldots,s_i^m\pi_j^m\}$). For example, if $s_i = (1,1,1)$ and $\pi_j = (3,2,1)$ in the transition block of Figure 5a) then $h = 1$ as $s_i^1 \pi_j^1 = \{3\} \in \Phi$. If $\pi_j = (2,1,3)$ then $h = 2$ because $\{s_i^1 \pi_j^1, s_i^2 \pi_j^2\} = \{2,1\}$ and there is the refined face $\{1,2\}$ in Φ. For $\pi_j = (1,3,2)$, h also equals 2 since $\{3\} \in \Phi$ and $\{3\} \subset \{1,3\}$. Consider $T^{m,h}$ the set of all m-dimensional vectors in $\mathbb{R}^m$ whose i^{th} component is 1 or -1 if $i \geq h$ an 0 otherwise. For example, if $h = 2$ in then $T^{3,2} = \{(0,1,1), (0,-1,1), (0,1,-1), (0,-1,-1)\}$.

Let D^m be a transition block whose labels of its refined faces are in Φ. For each permutation $\pi_j \in \Pi^m$ and $s_k \in S^m$ we can compute an h such that the vertices of the simplices in D^m can be specified as follows:

$$
\begin{aligned}
v_0 &= \quad (0,\ldots,0) \\
v_i &= \quad v_{i-1} + s_k^{\pi_j^i} e_{\pi_j^i} \qquad 0 < i \leq h-1 && (2) \\
v_h &= \quad v_{h-1} + \tfrac{1}{2} \sum_{s=h+1}^{m} s_k^{\pi_j^s} e_{\pi_j^s} && (3) \\
v_i &= \quad v_h + \tfrac{1}{2} \sum_{s=h+1}^{i} t_k^{\pi_j^s} e_{\pi_j^s} \qquad h < i \leq m && (4)
\end{aligned}
$$

where $t_k^{\pi_j^i}$ is the $\pi_j^{i\,th}$ component of a $t_k \in T^{m,h}$.

It is important to note that the vertices $[v_0, \ldots, v_h, \ldots, v_m]$ of each simplex σ generated from the scheme above are sorted in a such way that $\{v_h, \ldots, v_m\}$ generates the face of σ that is contained in a refined face of D^m.

Another important fact to be noted is that if $h \neq m$ then the rightmost term in expression (3) above computes the vertices of the simplices in D^m that are the center of the faces contained in the $(i+1)$-blocks adjacent to D^m and right most term in (4) computes the others vertices of the simplicial decomposition of these faces.

For example, the vertices of the simplex σ_1 in Figure 5b), that appear in the simplicial decomposition of the transition block in Figure 5a), are defined from $s_i = (1,1,1)$ and $\pi_j = (3,2,1)$ (then $h = 1$), has its vertices defined as $v_0 = (0,0,0)$, $v_1 = (0,0,1) + (1/2,1/2,0) = (1/2,1/2,1)$ (expression (3)), $v_2 = (1/2,1/2,1) + (0,1/2,0) = (1/2,1,1)$ and $v_3 = (1/2,1/2,1) + (0,1/2,0) + (1/2,0,0) = (1,1,1)$, as $t_k = (0,1,1)$. If we take $s_i = (1,1,1)$ and $\pi_j = (1,3,2)$ (thus $h = 2$), we can generate the simplex σ_2 of Figure 5b) whose vertices are $v_0 = (0,0,0)$, $v_1 = (1,0,0)$, $v_2 = (1,0,1) + (0,1/2,0) = (1,1/2,1)$ and $v_3 = (1,1/2,1) + (0,1/2,0) = (1,1,1)$, as $t_k = (0,0,1)$. It is not difficult to note that a complete decomposition of D^m can be obtained changing s_i, π_j, and t_k. Figure 5c) shows the boundary of the complete decomposition of the transition block in Figure 5a).

¿From both algebraic arrangements above (for basic and transition blocks) it is possible to generate a complete simplicial decomposition for a grid $\mathcal{G}$ and this decomposition is called the J_1^a **triangulation** of $\mathcal{G}$. A property of the J_1^a triangulation that will be important

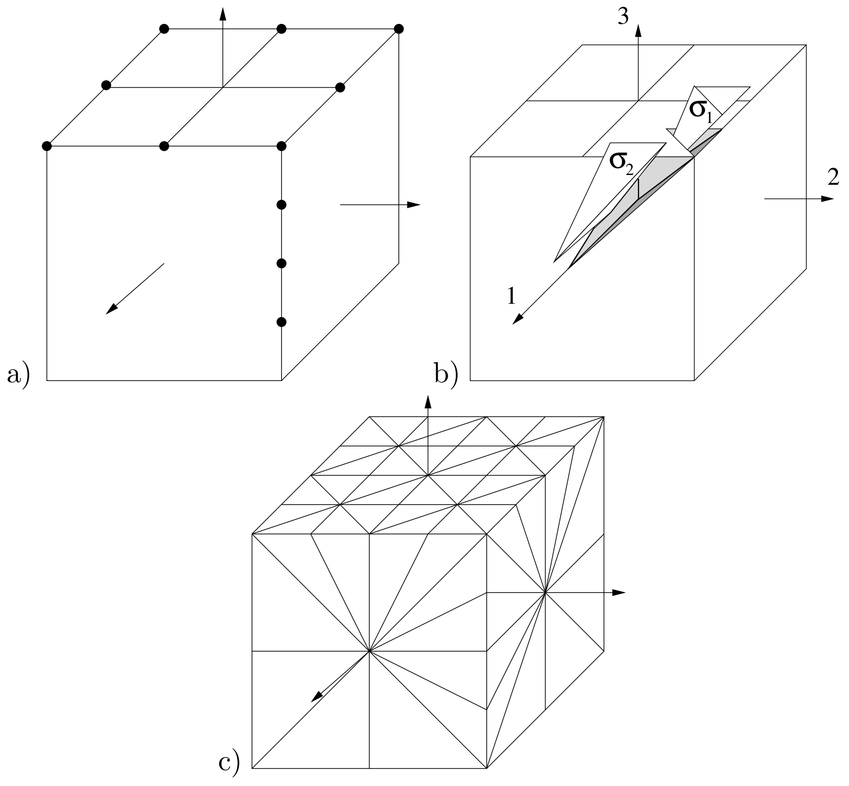


Figure 5: Refinement of D^m. a) Transition block with a two-dimensional and a one-dimensional refined face. b) Two tetrahedrons in a transition block. c) The boundary of the complete refinement of the transition block in a).

to the development of the pivoting rules describe bellow is that all m-simplices of a D^m block share the central vertex of the block. In the following sections we shall assume the convention that v_0 in a m-dimensional simplex $\sigma = [v_0, \ldots, v_m]$ is the central vertex of D^m.

In the next section we present an efficient mechanism that allows to explore the adjacency relationship of the simplices in a J_1^a triangulation.

4.2 Labeling and Pivoting Rules

In Section 4.1 above we show how the J_1^a triangulation decomposes standard hypercubes in a set of simplices. Since a generic hypercube in $\mathbb{R}^m$ can be seen as a standard hypercube, the same decomposition can be used in each block in a grid $\mathcal{G}$. The main problem with applying this decomposition scheme to the blocks of a grid $\mathcal{G}$ is that the location of each simplex and its neighbors would demand the storage of a large set of additional information, compromising the applicability of the J_1^a triangulation.

In order to solve this problem we propose a labeling scheme that allows to locate any simplex of the J_1^a triangulation and traverse its neighborhood very efficiently also reducing storage requirements. This labeling scheme consists in associating a set of integer numbers to each simplex σ and retrieve, from these integers, the spatial localization and the neighborhood relationships of σ.

4.2.1 Labeling

As shown here, only four integers are sufficient to specify any simplex in a basic block. In a transition block six integers are needed. The labeling of a simplex σ is derived from the algebraic mechanism that defines the vertices of σ jointly with a numbering of the blocks in $\mathcal{G}$.

The numbering of the i-blocks in $\mathcal{G}$ is generated as follows: let $n_1, n_2, \ldots, n_m$ be the number of initial 0-blocks of $\mathcal{G}$ in each coordinate axis of $\mathbb{R}^m$. If we sort the 0-blocks from negative to positive direction in each coordinate axis, we can associate to each 0-block D an m-dimensional vector g_D^0 where the k^{th} component of g_D^0 corresponds to the positioning of the 0-block in the k^{th} axis. For example, in Figure 6a) the 0-block D is associated with the vector $g_D^0 = (1, 2)$. The same idea can be employed to a general i-block of $\mathcal{G}$. If we suppose that all blocks in $\mathcal{G}$ are in the same level i, we can associate for each i-block D a vector g_D^i as above. An example of this association is shown in Figure 6b) where two blocks D_1 and D_2 in levels of refinement 1 and 2 respectively are associated with the vectors $g_{D_1}^1 = (2, 4)$ and $g_{D_2}^2 = (5, 11)$. From the level of refinement and the vectors g_D^i we can generate a unique label to each i-block $D \in \mathcal{G}$ by making use of the following function:

$$l_g(g_D^i) = \sum_{j=1}^{m} b_j g_D^{i\,j}$$

where $g_D^{i\,j}$ is the j^{th} component of g_D^i and b_j is given by $b_1 = 1,\ b_j = b_{j-1} 2^i n_{j-1};\ j = 2, \ldots, m$.

The function above defines the blocks in $\mathcal{G}$ uniquely, i.e., given $l_g(g_D^i)$ we can recover g_D^i through the following recurrency relation:

$$\begin{cases} g_D^{i\,j} = \lfloor \frac{l_g(g_D^i)}{b_j} \rfloor \\ l_g(g_D^i) \leftarrow l_g(g_D^i) \texttt{ mod } b_j, \quad j = m, \ldots, 2, 1 \end{cases}$$

where $\lfloor a \rfloor$ is the biggest integer smaller or equal than a.

Note that the labeling scheme above makes it possible to restore any block $D \in \mathcal{G}$ from only two integers, namely, the level of refinement of D and $l_g(g_D^i)$ (since we have $n_i,\ i = 1, \ldots, m$ and the configuration of the original domain from which the 0-blocks are derived).

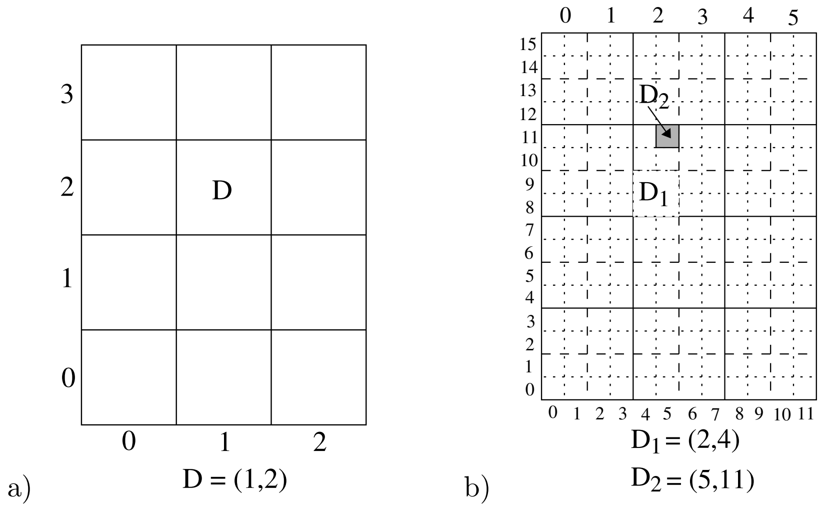


Figure 6: Labeling of i-blocks. a)0-refinement in $\mathbb{R}^2$. b)1-refinement in $\mathbb{R}^2$

As the simplices in a basic i-block are generated from $s_k \in S^m$ and $\pi_j \in \Pi^m$, if we define two bijective maps $l_s : S^m \to \mathbb{N}$ and $l_\pi : \Pi \to \mathbb{N}$ we can recover any simplex contained in a basic i-block from the integers given by l_g, l_s, l_π and i. In a transition block, in addition to these four integers already, we also need to make use of h and $l_t : T^{m,h} \to \mathbb{N}$, where l_t is also a bijective map. Examples of how such bijective maps can be constructed are given in the book by Page and Wilson [19]. As a consequence of the above construction it is also possible to define relationships, such as neighborhood between simplices, algebraically. Adjacency, for instance, can be given by the pivotin rules presented in the next section.

4.2.2 Pivoting Rules

This section presents the pivoting rules of the J1a triangulation. They are responsible to define adjacency relationships algebraically. The advantage of pivoting using algebraic relations is to avoid explicit storage of neighborhood, as opposed to other triangulation methods that have no means of finding a neigboring element unless it is explicitly represented in the data structure. Additionally, by pivoting to find a neighbor, it becomes feasible to avoid storing of the simplicial elements altogether.

The pivoting rules therefore are responsible for the localization of neighbor simplex in the J_1^a triangulation. This neighborhood relationship is derived from the definition of pivoted of a simplex.

Let T be the J_1^a triangulation obtained from a grid $\mathcal{G}$, σ_1 an m-simplex in T generated by the vertices $[v_0, \ldots, v_m]$, and τ an $(m-1)$-face of σ_1 given by $[v_0, \ldots, v_{k-1}, v_{k+1}, \ldots, v_m]$. If τ is not in the boundary of T, then there is exactly one m-simplex σ_2 in T generated by

$[v_0, \ldots, v_{k-1}, w, v_{k+1}, \ldots, v_m]$, called **pivoted** of σ_1 by the k^{th} vertex. Note that σ_1 and σ_2 have τ as a common face. Figure 7 illustrates a pivoted simples. As a general rule, the pivoted of a simplex σ by a vertex v, is the simplex in T (if any) that shares with σ the face that does not contain v.

Let D^m be a basic block in $\mathcal{G}$ and $\sigma = [v_0, \ldots, v_m]$ a simplex of the J_1^a decomposition of D^m. It is not difficult to see that the pivoted of σ by v_0 is always in an adjacent block to D^m and that the pivoted of σ by the vertices $v_1, \ldots, v_m$ are all contained in D^m. Based on that fact, we can generate a set of rules that, from the labels of a simplex, identify its pivoted. For example, consider the simplex σ in Figure 7 generated from $\pi_j = (2, 1)$ and $s_k = (1, -1)$. The pivoted of σ by the 2nd vertex (v_1) is obtained by maintaining the label of the block (s_k) and the level of refinement unchanged, only changing π_j to (1,2) (see Figure 7).

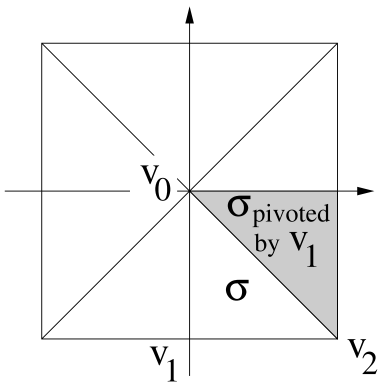


Figure 7: Pivoting by a vertex.

If the pivoting rule is defined as a map $\texttt{piv} : \mathbb{N}^4 \to \mathbb{N}^4$, that maps labels to labels then we can write:

$$\texttt{piv}(l_g(D^2), i, l_\pi((2,1)), l_s(s_k)) = (l_g(D^2), i, l_\pi((1,2)), l_s(s_k))$$

There are two different sets of pivoting rules, one for basic blocks and another for transition blocks. These sets of pivoting rules are listed in Appendix A, and are capable of handling all possible cases. The proof that this set of pivoting rules are really capable of computing the pivoted of any simplex correctly is very exhaustive and is document in a Technical Report [6].

The J_1^a triangulation lends itlef naturally to adptive refinement. One form of adaptive refinement implemented for the method is presented in the next section.

5 Adaptive Mechanism Based on Submanifolds

In this section we describe the adaptive mechanism implemented in code. As already mentioned, any adaptive strategy could be adopted, as for example curvature [22]. Here we choose to present a scheme for refinement based on submanifolds.

Let $M = F^{-1}(0)$ be an implicitly defined manifold where $F : \Omega \subset I\!R^m \rightarrow I\!R^n$. An implicity submanifold N of M can be defined by $N = H^{-1}(0)$, where $H = (F, G)^t$, with $G : \Omega \subset I\!R^m \rightarrow I\!R^k$. Notice that N is given by the intersection between M and the implicity manifold which is the zero set of G, as shown in Figure 8a).

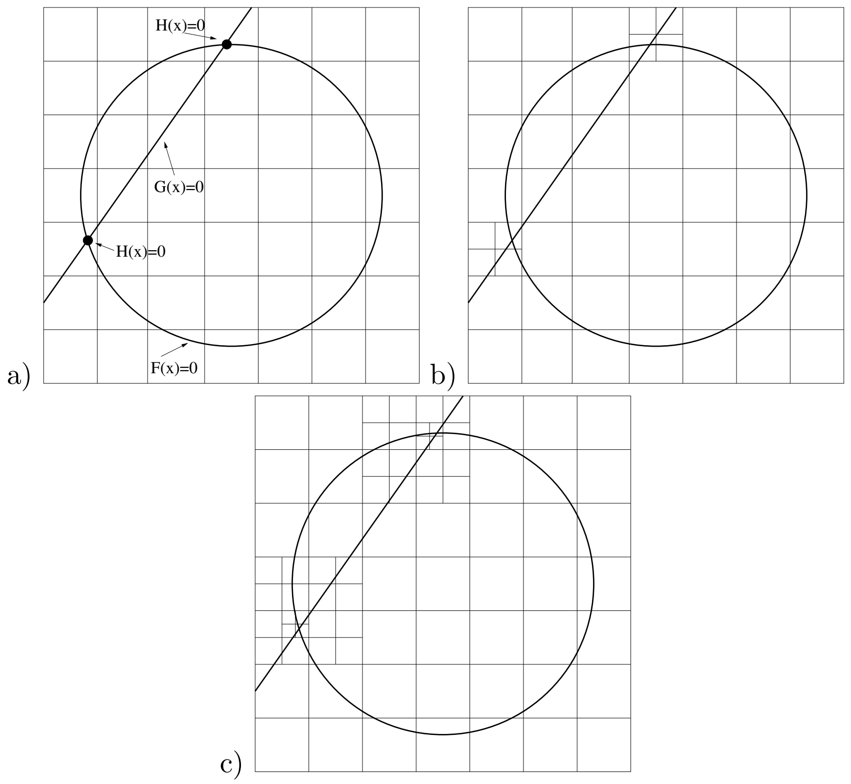


Figure 8: Refinement on submanifold.
a) one level of refinement. b) two levels of refinement.

Let $\sigma = [v_0, \ldots, v_m]$ be a simplex and $N = H^{-1}(0)$ be an implicit submanifold of M. In a similiar way as described in Section 2 we can define a linear aproximation for N in σ as

$N_\sigma = \{v \in \sigma | v = \lambda_0 v_0 + \cdots + \lambda_m v_m\}$, where:

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ F(v_0) & F(v_1) & \cdots & F(v_m) \\ G(v_0) & G(v_1) & \cdots & G(v_m) \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Note that $N_\sigma \subset M_\sigma$ since, for every $v \in N_\sigma$, $\lambda_0 F(v_0) + \cdots + \lambda_m F(v_m) = 0$ is satisfied.

Therefore, to construct an adaptive approximation of M close to N, all that is necessary is to mark as a block to be refined the blocks whose simplices have $N_\sigma \neq \emptyset$. The process is repeated for every refined block until the desired level of refinement is reached. Is worth reminding that every refinement imply in the calculation and re-triangulation of the transition blocks. Figure 8 illustrates two levels of the refinement process when N is a submanifold given by two points of the circle $F(x) = 0$.

It is well worth noting that the process of detecting the submanifolds in a model gives as a result all the indexes that actually represent the submamifold, due to the characteristics intrinsic of the labeling process. Therefore, if a separate representation of the submanifold is needed, it can be completely obtained in a single pass, while the reconstruction is carried out. An example of this type of adaptive mechanism is given in the next section.

The next section also presents the results of applying J_1^a to a number of cases and compares its efficiency with using Marching Cubes decomposition.

6 Results

In this section we present two sample applications where the J_1^a triangulation can efficiently be employed, namely: iso-surface extraction from a sequence of images and implicit manifold reconstruction. In both cases we emphasize how the intrinsically adaptive behavior of the J_1^a triangulation can be useful both in the improvement of the visualization and in the accuracy of the model.

6.1 Iso-Surface Extraction

To test the behavior of the J_1^a triangulation in iso-surface extraction from images, we make use of a synthetic data set and another generated from real images of a heart.

First we generated a set of twenty images of disks by slicing a cylinder. Those were input into the algorithm. Figure 9a) shows the resulting mesh where a $5 \times 5 \times 5$ initial grid is employed without any refinement. No interpolation scheme was used, in order to illustrate the effect of the adaptive refinement. This effect can be seen in Figure 9b) where two levels of refinement were employed in a predefined region. Figure 9c) shows the result of reconstructing the cylinder through the marching cubes (MC) algorithm with a $40 \times 40 \times 40$. This is equivalent in terms of resolution to two levels of refinement applying J_1^a over a $5 \times 5 \times 5$ grid. As the MC algorithm does not adaptively refine, in order to improve the resolution in a specific region it is necessary to refine the whole domain. Table 1 shows the number of faces (triangles) generated by the MC against the J_1^a triangulation. It can be seen from table 1 that although the J_1^a triangulation generates more faces than the marching cubes algorithm in an uniformly refined domain, this number is considerable smaller when adaptive procedures are employed in 25% of the domain. Most applications in fact need progressive refinement depending on object feature, which renders J_1^a a good alternative. Also, the labeling and pivoting strategies given offer the possibility of larce savings in storage.

Table 1: Number of faces in J_1^a vs. MC

J_1^a	$5 \times 5 \times 5$ without refinement 3,088	$5 \times 5 \times 5$ with two levels of refinement 8,232
MC	$10 \times 10 \times 10$ 784	$40 \times 40 \times 40$ 13,804

Different criteria could be used to define where the adaptive refinement must be done, such as curvature for example. In our implementation of iso-surface extraction, the refinement is performed in portions of space, previously specified by the user. Although very simple, this approach to specify the refinement is appropriate to some applications we are interested in, such as fluid flow simulation in complex domains. An example of such domain is shown in Figure 10a) where the surface of the heart bounds the simulation domain. As can be seen from Figure 10b), refinement was realized around the main artery, which is the target region in our simulations, i.e., the region where it is necessary to obtain more accurate results. It is worth noticing that the J_1^a triangulation can be just as efficiently utilized to decompose the interior of the domain, not just the surface. This interior decomposiction produces simplices that are robust and thus appropriate to numerical simulations [5].

6.2 Implicit Manifold Reconstruction

In the case of implicit manifolds, the J_1^a triangulation has also performed well the reconstruction process. In order to demonstrate how general the refinement criteria can be we have employed a adaptive mechanism based on sub-manifolds.

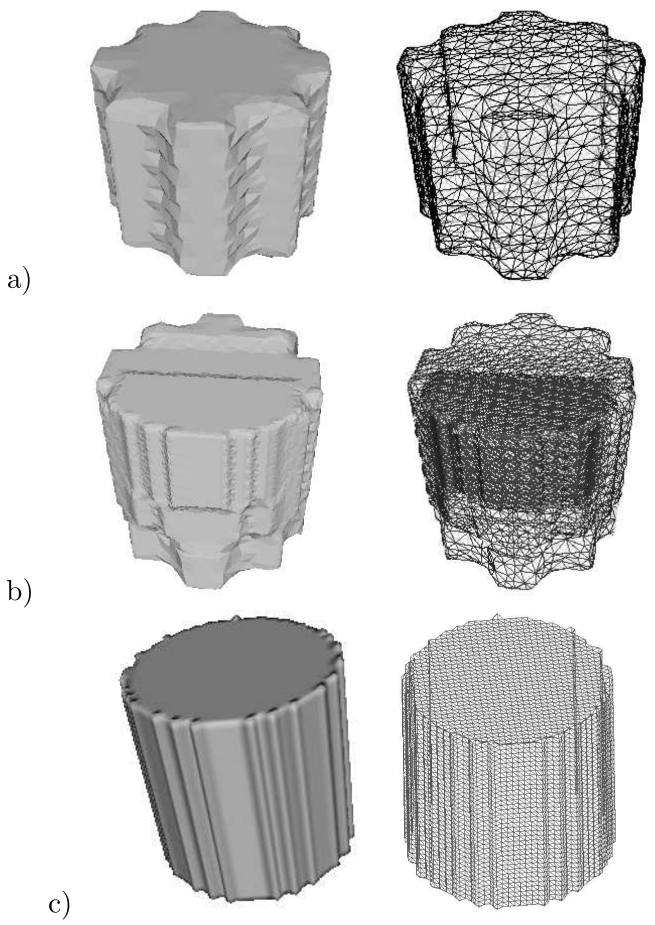


Figure 9: Cylinder iso-surface extraction: a) initial grid $5 \times 5 \times 5$ without refinement; b) two levels of a local refinement; c) marching cubes in a $40 \times 40 \times 40$ grid.

Figure 11 shows a set of implicitly defined curves on a sphere describing the word `VISUAL`. We implement the refinement strategy based on submanifolds presented in Section 5 by searching for submanifolds (curves) contained on the surface and executing the refinement around those. Figures 12 a), b) and c) present the approximations of the curves in three consecutive levels of refinement. The initial grid is $20 \times 20 \times 20$. The respective meshes can be seen in Figures 12 d), e) and f).

In Figure 13 we illustrate the employment of the J_1^a triangulation in higher dimensions. It presents a projection of $S^1 \times S^1$ (a surface in $\mathbb{R}^4$).

Note that in both image and implicit applications only the cubic grid needs to be stored to construct the J_1^a triangulation, as the simplices can be generated during the track of

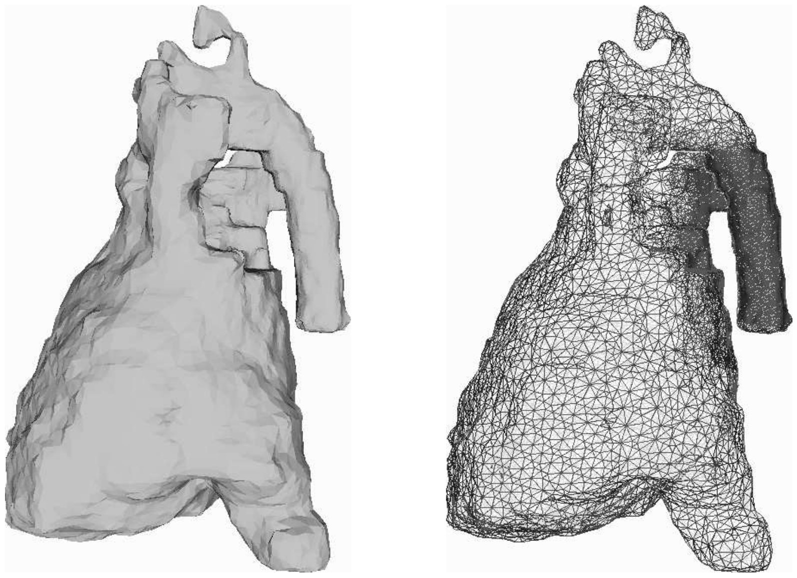


Figure 10: Decomposition of a surface of a heart

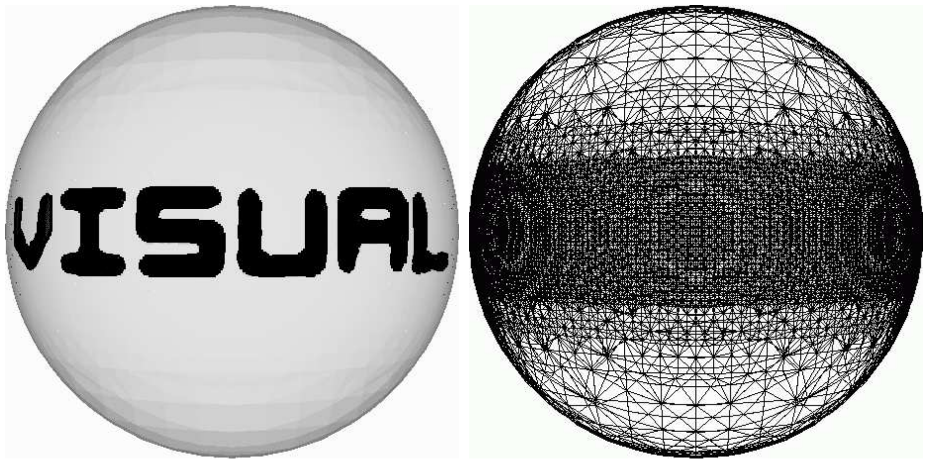
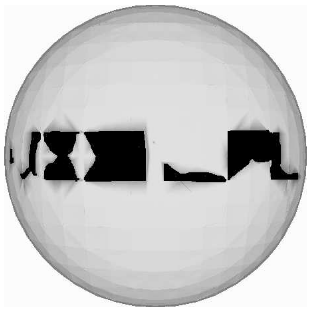
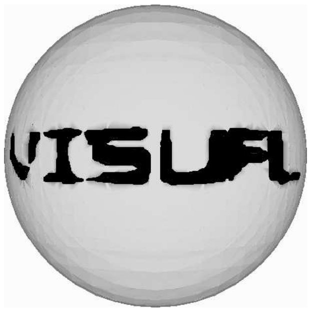


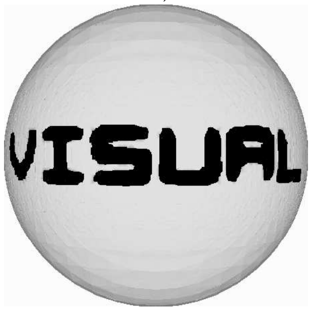
Figure 11: Four levels of refinement around the sub-manifolds. a) Shading b) Mesh

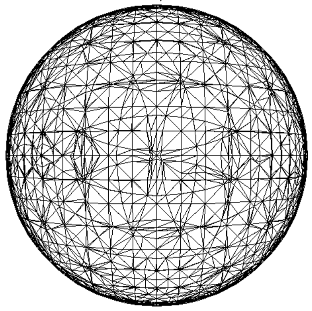
the faces through the algebraic mechanisms presented in the previous sections. There is, therefore, no need to store the triangulation itselv or to build any specific data structure to represent neighborhood relationships. This is a considerable advantage over other known approaches, which rely data strutures that must store adjacency relationships or otherwise pay expensive computational cost to recover this information.

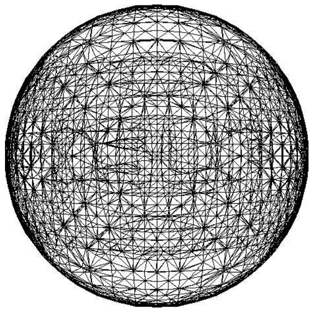
Table 6.2 shows the computational times, in seconds, for processing the models in Figures

a)


b)


c)


d)


e)


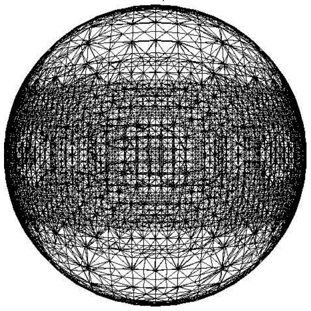
f)


Figure 12: a),b) and c) approximations with 1,2 and 3 levels of refinement; d),e) and f) meshes of a),b) and c) respectively.

10, 12, and 13. It is worth mentioning that the values in table 6.2 take into account the time spent to write the models on disk.

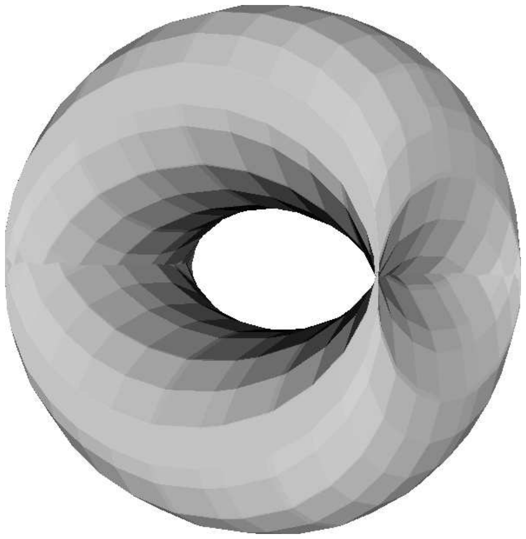


Figure 13: A projection of $S^1 \times S^1$

Heart	8.51
Spheres	
1 - Refinement	0.39
2 - Refinement	1.20
3 - Refinement	3.22
$S^1 \times S^1$	8.55

Table 1: Computational times (seconds) to process the models in Figures 10, 12, and 13.

As a final example to illustrate the employment of J_1^a with a refinement process based on curvature, we have reconstructed a implicitly defined model of two spheres connected by a thin cylinder. Figure 14 show two reconstructions. The original reconstruction (Figure 14 a)), without refinement, is unable to detect the correct topology of the object, whilst a refinement drawn by an estimate of the curvature radius will correctly generate the object. The magnitude of the curvature radius dictates the level of refinement necessary. In the example of figure 14 the decision criteria indicated two further levels of refinement in the region with lowest curvature radius, generating the correct topology for the object.

7 Conclusions

This paper described a novel spatial decomposition technique that handles three problems adequatelly: decomposition in any dimension, control of storage requirement and intrinsic adaptive behaviour (also in any dimension). These features of the J_1^a triangulation

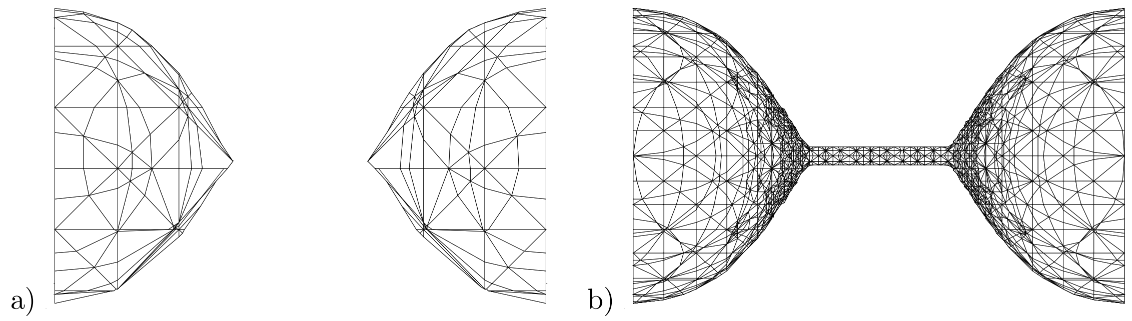


Figure 14: Curvature based refinement of an implicit model, supporting correct detection of object topology. a) No refinement b)Two levels of refinement based on curvature radius

are mainly due to the provision of algebraic mechanisms that are capable of generating the simplices that form the simplicial mesh as well as the adjacency relationship between simplices. These mechanisms allow the generation of geometrical and topological data about the simplicial mesh without explicitly storing either the neighborhood relationships or even the mesh itself. Results have shown the efficiency of the process for meshes generated in applications such as iso-surface construction, implicit manifold reconstruction with submanifold refinement, and surface tracking in dimension higher than three. No known spatial decomposition strategy is capable of offering all these characteristics together.

As a future development it is our intention to apply the adaptive mesh generation technique presented here to problems of 3D reconstruction from projections, an important problem areas such as medical images, and to problems in 3D computational fluid dynamics.

Additionally, we are studying the use of J_1^a in compression/decompression of models.

8 Acknowledgements

We wish to acknowledge the Brazilian financial agenncies CNPq (proc. 307268/2003-9) and FAPESP (proc. 03/02815-0) involved in financing the project that generated these results.

Appendix A

This appendix contains the pivoting rules for the J_1^a triangulation. Supose that v_i represents the i^{th} vertice of a simplex σ contained in a r-block D^m, whose associated vector is $g^r_{D^m}$, and that σ is generated from s_k, π_j and t_p. We represent the changes of $g^r_{D^m}$, s_k, π_j and t_p during the pivoting by $\overline{g^r_{D^m}}$, $\overline{s_k}$, $\overline{\pi_j}$ and $\overline{t_p}$ respectively (Note that just some elements need to change during the pivoting).

If D^m is a basic block then:

1. if $0 < i < m$ then
   $\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k)) = (l_g(g^r_{D^m}), r, l_\pi(\overline{\pi_j}), l_s(s_k))$
   $\pi_j = (\pi^1_j, \ldots, \pi^{i-1}_j, \pi^i_j, \pi^{i+1}_j, \ldots, \pi^m_j)$ and $\overline{\pi_j} = (\pi^1_j, \ldots, \pi^{i-1}_j, \pi^{i+1}_j, \pi^i_j, \ldots, \pi^m_j)$
2. if $i = m$ then
   $\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k)) = (l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(\overline{s_k}))$
   $\overline{s_k} = s_k - 2s_k^{\pi^m_j} e_{\pi^m_j}$
3. if $i = 0$ and $\overline{g^r_{D^m}}$ is a basic block then
   $\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k)) = (l_g(\overline{g^r_{D^m}}), r, l_\pi(\pi_j), l_s(\overline{s_k}))$
   $\overline{g^r_{D^m}} = g^r_{D^m} + s_k^{\pi^1_j} e_{\pi^1_j}$
   $\overline{s_k} = s_k - 2s_k^{\pi^m_j} e_{\pi^m_j}$
4. if $i = 0$ and $\overline{g^r_{D^m}}$ is a transition block with $\phi = \{\pi^1_j s_k^{\pi^1_j}\}$ a refined face then
   $\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k)) = (l_g(\overline{g^r_{D^m}}), \overline{r}, l_\pi(\pi_j), l_s(\overline{s_k}), \overline{h}, l_t(\overline{t_p})$
   $\overline{g^r_{D^m}} = \left\lceil \frac{g_{D^m}^{r\pi^1_j}}{2} \right\rceil e_{\pi^1_j} + \cdots + \left\lceil \frac{g_{D^m}^{r\pi^m_j}}{2} \right\rceil e_{\pi^m_j} + s_k^{\pi^1_j} e_{\pi^1_j},$
   $\overline{r} = r - 1$
   $\overline{s_k} = (2(g_{D^m}^{r\pi^2_j} \bmod 2) - 1)e_{\pi^2_j} + \cdots + (2(g_{D^m}^{r\pi^m_j} \bmod 2) - 1)e_{\pi^m_j} - s_k^{\pi^1_j} e_{\pi^1_j}$
   $\overline{h} = 1$
   $\overline{t_p} = s_k^{\pi^2_j} e_{\pi^2_j} + \cdots + s_k^{\pi^m_j} e_{\pi^m_j}$
5. if $i = 0$ and $\overline{g^r_{D^m}}$ is a transition block where $\phi = \{\pi^1_j s_k^{\pi^1_j}\}$ is not a refined face then
   $\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k)) = (l_g(\overline{g^r_{D^m}}), r, l_\pi(\pi_j), l_s(\overline{s_k}), \overline{h}, l_t(\overline{t_p}))$
   $\overline{g^r_{D^m}} = g^r_{D^m} + s_k^{\pi^1_j} e_{\pi^1_j}$
   $\overline{s_k} = s_k - 2s_k^{\pi^1_j} e_{\pi^1_j}$
   $\overline{h} = m$
   $\overline{t_p} = (0, \ldots, 0)$

If D^m is a transition block then:

6. if $i = 0$, $h = 1$ and $\overline{g^r_{D^m}}$ is a basic block then
   $\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(\overline{g^r_{D^m}}), \overline{r}, l_\pi(\pi_j), l_s(\overline{s_k}))$
   $\overline{g^r_{D^m}} = \frac{1}{2}(4g_{D^m}^{r\pi^1_j} + s_k^{\pi^1_j+1})e_{\pi^1_j} + \cdots + \frac{1}{2}(4g_{D^m}^{r\pi^m_j} + s_k^{\pi^m_j+1})e_{\pi^m_j} + s_k^{\pi^1_j} e_{\pi^1_j}$
   $\overline{r} = r + 1$
   $\overline{s_k} = -s_k^{\pi^1_j} e_{\pi^1_j} + t_p^{\pi^2_j} e_{\pi^2_j} + \cdots + t_p^{\pi^m_j} e_{\pi^m_j}$
7. if $i = 0$, $h = 1$ and $\overline{g^r_{D^m}}$ is a transition block then
   $\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(\overline{g^r_{D^m}}), \overline{r}, l_\pi(\pi_j), l_s(\overline{s_k}), \overline{h}, l_t(\overline{t_p}))$

$\overline{g^r_{D^m}} = \frac{1}{2}(4g^{r\pi^1_j}_{D^m} + s_k^{\pi^1_j+1})e_{\pi^1_j} + \cdots + \frac{1}{2}(4g^{r\pi^m_j}_{D^m} + s_k^{\pi^m_j+1})e_{\pi^m_j} + s_k^{\pi^1_j}e_{\pi^1_j}$
$\overline{r} = r + 1$
$\overline{s_k} = -s_k^{\pi^1_j}e_{\pi^1_j} + t_p^{\pi^2_j}e_{\pi^2_j} + \cdots + t_p^{\pi^m_j}e_{\pi^m_j}$
$\overline{h} = m$
$\overline{t_p} = (0, \dots, 0)$

8. if $i = 0$, ϕ is a refined and $s_k^{\pi^1_j}\pi^1_j \in \phi$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(\overline{g^r_{D^m}}), \overline{r}, l_\pi(\pi_j), l_s(\overline{s_k}), \overline{h}, l_t(\overline{t_p}))$
$\overline{g^r_{D^m}} = \left\lfloor \frac{g^{r\pi^1_j}_{D^m}}{2} \right\rfloor e_{\pi^1_j} + \cdots + \left\lfloor \frac{g^{r\pi^m_j}_{D^m}}{2} \right\rfloor e_{\pi^m_j} + s_k^{\pi^1_j}e_{\pi^1_j},$
$\overline{r} = r - 1$
$\overline{s_k} = (2(g^{r\pi^2_j}_{D^m} \bmod 2) - 1)e_{\pi^2_j} + \cdots + (2(g^{r\pi^m_j}_{D^m} \bmod 2) - 1)e_{\pi^m_j} - s_k^{\pi^1_j}e_{\pi^1_j}$
$\overline{h} = 1$
$\overline{t_p} = s_k^{\pi^2_j}e_{\pi^2_j} + \cdots + s_k^{\pi^m_j}e_{\pi^m_j}$

9. if $i = 0$, $h = m$ and $\overline{g^r_{D^m}}$ is a basic block then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(\overline{g^r_{D^m}}), r, l_\pi(\pi_j), l_s(\overline{s_k}))$
$\overline{g^r_{D^m}} = g^r_{D^m} + s_k^{\pi^1_j}e_{\pi^1_j},$
$\overline{s_k} = s_k - 2s_k^{\pi^1_j}e_{\pi^1_j}$

10. if $i = 0$, $h = m$ and $\overline{g^r_{D^m}}$ is a transition block then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(\overline{g^r_{D^m}}), r, l_\pi(\pi_j), l_s(\overline{s_k}), h, l_t(t_p))$
$\overline{g^r_{D^m}} = g^r_{D^m} + s_k^{\pi^1_j}e_{\pi^1_j},$
$\overline{s_k} = s_k - 2s_k^{\pi^1_j}e_{\pi^1_j}$

11. if $i = 0$ and $1 < h \leq m$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(\overline{g^r_{D^m}}), r, l_\pi(\pi_j), l_s(\overline{s_k}), h, l_t(t_p))$
$\overline{g^r_{D^m}} = g^r_{D^m} + s_k^{\pi^1_j}e_{\pi^1_j},$
$\overline{s_k} = s_k - 2s_k^{\pi^1_j}e_{\pi^1_j}$

12. if $i = m$, $h = m$ and for all refined face ϕ, $-s_k^{\pi^m_j}e_{\pi^m_j}$ is not in ϕ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(\overline{s_k}), h, l_t(t_p))$
$\overline{s_k} = s_k - 2s_k^{\pi^m_j}e_{\pi^m_j}$

13. if $i = m$ and $0 < h < m$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(\overline{t_p}))$
$\overline{t_p} = t_p - 2t_p^{\pi^m_j}e_{\pi^m_j}$

14. if $0 < i < m$ and $h = m$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\overline{\pi_j}), l_s(s_k), h, l_t(t_p))$
$\overline{\pi_j} = (\pi^1_j, \dots, \pi^{i-1}_j, \pi^{i+1}_j, \pi^i_j, \dots, \pi^m_j)$

15. if $0 < i < h - 1$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\overline{\pi_j}), l_s(s_k), h, l_t(t_p))$
$\overline{\pi_j} = (\pi^1_j, \dots, \pi^{i-1}_j, \pi^{i+1}_j, \pi^i_j, \dots, \pi^m_j)$

16. if $0 < h < i < m$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\overline{\pi_j}), l_s(s_k), h, l_t(t_p))$
$\overline{\pi_j} = (\pi_j^1, \dots, \pi_j^{i-1}, \pi_j^{i+1}, \pi_j^i, \dots, \pi_j^m)$

17. if $0 < h = i < m$ and $s_k^{\pi_j^{h+1}} \neq t_p^{\pi_j^{h+1}}$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(\overline{s_k}), h, l_t(\overline{t_p}))$
$\overline{s_k} = s_k - 2s_k^{\pi_j^{h+1}} e_{\pi_j^{h+1}}$
$\overline{t_p} = t_p - 2t_p^{\pi_j^{h+1}} e_{\pi_j^{h+1}}$

Let ϕ be a refined face of D^m and $q(\pi_j, s_k, \phi)$ difined as:

$$\begin{cases} 0 & \text{if } \phi \not\subset \{s_k^{\pi_j^1}\pi_j^1, \dots, s_k^{\pi_j^m}\pi_j^m\} \\ \min\{i | \phi \subset \{s_k^{\pi_j^1}\pi_j^1, \dots, s_k^{\pi_j^i}\pi_j^i\}\} & \text{otherwise} \end{cases}$$

18. if $0 < i = h < m$, $s_k^{\pi_j^{h+1}} = t_p^{\pi_j^{h+1}}$ and $h + 1 = \min\{q(\overline{\pi_j}, \overline{s_k}, \phi) | \phi \subset \{s_k^{\pi_j^1}\pi_j^1, \dots, s_k^{\pi_j^m}\pi_j^m\}\}$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\overline{\pi_j}), l_s(s_k), \overline{h}, l_t(\overline{t_p}))$
$\overline{\pi_j} = (\pi_j^1, \dots, \pi_j^{h-1}, \pi_j^{h+1}, \pi_j^h, \dots, \pi_j^m)$
$\overline{h} = h + 1$
$\overline{t_p} = t_p - t_p^{\pi_j^{h+1}} e_{\pi_j^{h+1}}$

19. if $0 < i = h < m$, $s_k^{\pi_j^{h+1}} = t_p^{\pi_j^{h+1}}$ and $h = \min\{q(\overline{\pi_j}, \overline{s_k}, \phi) | \phi \subset \{s_k^{\pi_j^1}\pi_j^1, \dots, s_k^{\pi_j^m}\pi_j^m\}\}$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\overline{\pi_j}), l_s(s_k), \overline{h}, l_t(\overline{t_p}))$
$\overline{\pi_j} = (\pi_j^1, \dots, \pi_j^{h-1}, \pi_j^{h+1}, \pi_j^h, \dots, \pi_j^m)$
$\overline{h} = h + 1$
$\overline{t_p} = t_p - t_p^{\pi_j^{h+1}} e_{\pi_j^{h+1}} + s_k^{\pi_j^h} e_{\pi_j^h}$

20. if $i = h - 1$, $1 < h \leq m$ and $h - 1 = \min\{q(\overline{\pi_j}, \overline{s_k}, \phi) | \phi \subset \{s_k^{\pi_j^1}\pi_j^1, \dots, s_k^{\pi_j^m}\pi_j^m\}\}$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\overline{\pi_j}), l_s(s_k), \overline{h}, l_t(\overline{t_p}))$
$\overline{\pi_j} = (\pi_j^1, \dots, \pi_j^{h-2}, \pi_j^h, \pi_j^{h-1}, \dots, \pi_j^m)$
$\overline{h} = h - 1$
$\overline{t_p} = t_p + s_k^{\pi_j^{h-1}} e_{\pi_j^{h-1}}$

21. if $i = h - 1$, $1 < h \leq m$ and $h = \min\{q(\overline{\pi_j}, \overline{s_k}, \phi) | \phi \subset \{s_k^{\pi_j^1}\pi_j^1, \dots, s_k^{\pi_j^m}\pi_j^m\}\}$ then
$\texttt{piv}(l_g(g^r_{D^m}), r, l_\pi(\pi_j), l_s(s_k), h, l_t(t_p)) = (l_g(g^r_{D^m}), r, l_\pi(\overline{\pi_j}), l_s(s_k), h, l_t(t_p))$
$\overline{\pi_j} = (\pi_j^1, \dots, \pi_j^{h-2}, \pi_j^h, \pi_j^{h-1}, \dots, \pi_j^m)$

References

[1] Akkouche S. and Galin E. Adaptive Implicit Surface Polygonization Using Marching Triangles. *Computer Graphics Forum*, 20(2): 67-80, 2001.

[2] Beall, M.W. and Shephard M. A general topology-based mesh data structure. *Int. J. for Numerical Methods in Engineering*, 40: 1573-1596, 1997.

[3] Banhiramka, P., Wenger, R. and Crawfis, R. Isosurface Construction in Any Dimension Using Convex Hulls. *IEEE Trans. on Vis. and Comp. Graph.*, 10(2):130-141.

[4] Bloomenthal J. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4): 341-356, 1988.

[5] Castelo A. Adaptive approximation of implicit manifolds with applications in implicit modeling and algebraic diferential equations. . *Ph.D. Thesis (in Portuguese), Mathematics Department, Pontifical Catholic University*, Rio de Janeiro, Brazil, 1992.

[6] Castelo, A., Siqueira, M.F. Tavares, G. - The J_1^a Triangulation. Technical Report 6/97 (in Portuguese). Pontifical Catholic University, Rio de Janeiro, Brazil, 1997.

[7] Chan S.L. and Purisima E.O. A new tetrahedral tesselation scheme for isosurface generation. *Comput. & Graphics*, 22(1): 83-90, 1998.

[8] Cignoni P., Ganovelli F., Montani C., and Scopigno R. Reconstruction of topologically correct and adaptive trilinear isosurface. *Comput. & Graphics-UK*, 24(3): 399-418, 2000.

[9] Cignoni P., Montani C., and Scopigno R. A comparison of mesh simplification algorithms. *Comput. & Graphics*, 22(1): 37-54, 1998.

[10] Crespin B., Guitton P., and Schlick C. Efficient and accurate tessellation of implicit sweeps. In *Proceedings of Constructive Solid Geometry98*, 1998.

[11] Desbrun M., Tsingos N. and Gascuel M.P. Adaptive sampling of implicit surfaces for interactive modeling and animation. *Computer Graphics Forum*, 15(5):319-325, 1996.

[12] Guéziec A. and Hummel R. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Trans. on Visual. Comp. Graphics*, 1(4): 328-342, 1995.

[13] Hilton A., Stoddart A., Illingworth J. and Windeatt T. Marching Triangles: range image fusion for complex object modeling. *International Conference on Image Processing*, 1996.

[14] Lorenen W.E. and Cline H.E. Marching Cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4): 163-169, 1987.

[15] McKie D. and McKie C. Essentials of Crystallography. *Blackwell Scientific Publications*, Oxford, 1986.

[16] Neves J.M.R and Persiano R.M. Visualizing Scalar Fields Represented by Adaptive Square Triangulation. *IEEE Proceedings SIBGRAPI97*, pp. 95-102, 1997.

[17] Ning P. and Bloomenthal J. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6) : 33-41, 1993.

[18] Min, C. Simplicial isosurfacing in arbitrary dimension and codimension. *Journal of Computational Physics*,190:295-310, 2003.

[19] Page E.S. and Wilson L.B. An Introduction to Computational Combinatorics. *Cambridge Computer Science Text* - 9, 1979.

[20] Ohlberger M. and Rumpf M. Hierarchical and adaptive visualization on nested grids. *Computing*, 59: 269-285, 1997.

[21] Treece G.M., Prager R.W., and Gee A.H. Regularised marching tetrahedra: improved iso-surface extracion. *Comput. & Graphics* 23: 583-598, 1999.

[22] Velho L. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools*, 1(2): 5–24, 1996.

[23] Westermann R., Kobbelt L., and Ertl T. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15: 100-111, 1999.