ORIGINAL ARTICLE

N. Hitschfeld-Kahler

Generation of 3D mixed element meshes using a flexible refinement approach

Abstract This paper describes and discusses the main characteristics and implementation issues of a 3D mixed element mesh generator based on a generalization of the modified octree approach. This mesh generator uses primitive elements of different type as internal nodes, a flexible refinement approach as refinement strategy (primitive elements are not always bisected), and bricks, pyramids, prisms and tetrahedra as final elements. The mesh generation process is divided in several steps: the generation of the initial mesh composed of primitive elements, the refinement of primitive elements until the point density requirements are fulfilled, the generation of a graded mesh between dense and coarse regions, and finally, the recognition of the final elements. The main algorithms and data structures are described in detail for each step of the mesh generation process. As result, examples of meshes that satisfy the Delaunay condition and that can be used with the control volume method are shown.

Keywords Modified octree approach · Delaunay meshes · Mixed-element meshes

1 Introduction

Since the last 20 years, modified octrees have been successfully used in geometric modeling applications and, in particular, for mesh generation [1–3]. The modified octree approach works as follows: the 3D domain is enclosed in a cube, whose octants are repeatedly refined at their edge midpoints until the boundary and internal quantities are sufficiently approximated. In order to generate a final mesh, elements with and without edge midpoints are partitioned into tetrahedra by using templates or ad-hoc algorithms. In case of mesh generation, the final elements have to fulfill the requirements imposed by the underlying numerical method.

The use of a flexible refinement approach allows us to select the best point at each refinement step. This kind of refinement was originally called *intersection* based approach, because it was introduced to refine intersected elements at the intersection points [4, 5].

This paper presents and discusses the main characteristics and implementation issues of a mixed element 3D mesh generator based on an extension of the modified octree approach. The modified octree approach is extended as follows: (1) The domain is enclosed using a brick. A brick has rectangular faces. (2) The internal elements (nodes) belong to a set of *well-shaped elements*, such as pyramids, prisms and tetrahedra of rectangular basis, and bricks. The set of elements that is called *well-shaped* depends on the application. This set has to be closed under the *refinement operator*, i.e., each element can be refined in such a way that all newly generated elements belong to this set. (3) The refinement is either bisection or what we have called a *flexible refinement* approach. Using the bisection approach, the refinement is always made at the edge midpoints. Using *flexible refinement* approach, the refinement is made at the most convenient edge point. The best point, the one whose associated refinement generates sons with the smallest aspect ratio, is chosen from either the Steiner points (points generated by the refinement of the edge neighbors) or the intersection points (points generated by the intersection between the object geometry and the target element). (4) The number of the newly generated elements under the refinement operator depends on the shape of the element and on the refinement direction. For example, if a refinement is required along one, two, or three coordinate axes, cubes, are subdivided into two halves, four quadrants, and eight octants, respectively. (5) The set of final elements is defined by the application.

N. Hitschfeld-Kahler
Departamento Ciencias de la Computación, FCFM,
Universidad de Chile, Blanco Encalada 2120, Santiago, Chile
E-mail: nancy@dcc.uchile.cl
Fax: +56-2-6895531

The final elements can be of the same type of the ones used as internal elements, or of other type. What we kept from the modified octree approach is that the refinement is parallel to the axes of the coordinate system. The previous ideas are applied first to the generation of 3D Delaunay mixed element meshes and then, to the generation of a subset of Delaunay meshes, the ones required by control volume discretization method.

Preliminary work on this mesh generator has been already published in [6–8]. In this paper, we discuss new improvements, implementation issues, complexity and the outreach of this mesh generator.

2 Characterization of well-shaped meshes based on modified octrees

This section introduces the concept of well-shaped mixed element meshes independent of the application.

2.1 Basic algorithm

Independent of the application, the generation of a well-shaped mesh is done by following the next consecutive steps:

1. Generate a macro-mesh that fits the geometry of the modeled device exactly.
2. Refine the macro-mesh to fulfill the density requirements specified by the user.
3. Generate a proper mesh for the current application,
   - Make the mesh 1-irregular.
   - Look for proper tessellations.
4. Store the information required by the application.

2.2 Macro-elements

A macro-mesh is composed by macro-elements. Macro-elements are used to fit the device geometry. The following theorem characterizes the set of macro-elements used in the generation of well-shaped mixed element meshes.

Theorem 1 *Let P be a set of polyhedra. P leads to well-shaped meshes if each polyhedron* $p \in P$

1. *fulfills the restrictions imposed by the current application, and*
2. *can be refined in such a way that all newly generated polyhedra also belong to P (P is closed).*

Proof: Condition (1) guarantees that the macro-elements fulfill the restrictions imposed by the current application. Condition (2) guarantees that each element generated through the refinement process fulfills condition (1).

2.3 Element refinement approaches

The most common way of refinement is bisecting an element, i.e., each element edge is bisected. This method is easy to analyze and implement, but it does not allow flexibility in choosing the most appropriate refinement point at each refinement step. That is why, our mesh generator uses a *flexible refinement* approach.

2.4 Elements with Steiner points

Irregular macro-elements are elements with edges split at least once. The point splitting an edge is called a *Steiner point*. Irregular elements appear between coarse and fine regions of the macro-mesh. In order to complete the mesh using as few elements as possible, the tessellation of irregular elements is necessary. A well-shaped irregular element is defined as follows:

Definition 1 Let p be an irregular polyhedron. The tessellation t of p is well-shaped if and only if t satisfies the conditions of the current application.

2.5 Final elements

Final elements are the elements that compose the final mesh. In the current version of the algorithm, we are not considering that final elements require further refinement. That is why, the set of final elements can include more basic elements than the ones included in the set of macro-elements.

3 Generation of mixed-element Delaunay meshes

This section describes how the general concepts described above in the Section Characterization of well-shaped meshes based on modified octrees can be applied in the generation of Delaunay meshes.

The concept of Delaunay triangulation can be extended to the mixed element meshes as follows:

Definition 2 A tessellation T of a set of points S is a Delaunay tessellation if there exists a point-free circumsphere for each tessellation element.

We use the term *Delaunay tessellation* and not *Delaunay triangulation* because our meshes include other element types than tetrahedra. Valid elements are convex polyhedra whose points are co-spherical.

3.1 Macro-elements

The following theorem characterizes the set macro-elements used in the generation of Delaunay mixed element meshes.

Theorem 2 *Let P be a set of convex polyhedra. P leads to Delaunay meshes if each polyhedron* $p \in P$

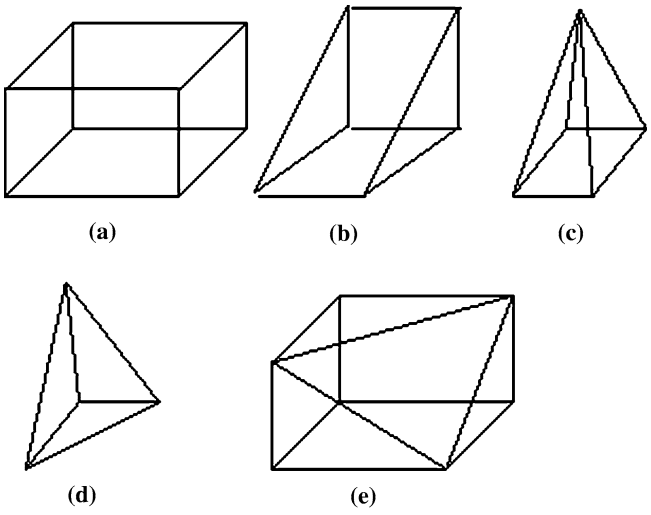


Fig. 1 Set of elements used to fit the device geometry

1. *is defined by co-spherical points (the circumsphere of p is point-free) and*
2. *can be refined in such a way that all newly generated polyhedra also belong to P (P is closed).*

Our set of macro-elements is composed of rectangular pyramids, rectangular prisms, bricks, rectangular tetrahedron and its complement inside a brick (Fig. 1). They are elements that satisfy Theorem 2 and can be properly refined as it will be shown in the next section.

3.2 Flexible element refinement

Since the bisection-based approach is a particular case of the flexible refinement approach, the current section only shows the refinement for each macro-element under flexible refinement. The refinement position of a target element is determined from the location of its Steiner points. As we have said before, we kept from the modified octree approach the refinement parallel to the main axes.

3.2.1 Refinement of bricks

Bricks can be split into two halves, four quarters or eight octants as before but edges are not necessary bisected. Figure 2 shows the different ways to split a brick using arbitrary refinement points. The only restriction is that parallel edges have to be split at the same relative position from their endpoints in order to generate smaller bricks and not hexahedra.

3.2.2 Refinement of prisms

Rectangular prisms can be partitioned in one, two, or three directions to generate two prisms, one brick plus two prisms, and two bricks plus four prisms, respectively, as shown in Fig. 3.

3.2.3 Refinement of the pyramid, the tetrahedron and its complement

The refinement of a rectangular pyramid, a tetrahedron and its complement must always be simultaneously in the three axes in order that the newly generated elements belong to the set of well-shaped macro-elements. As shown in Fig. 4, the refinement of the pyramid generates one brick, two prisms and two pyramids (left), the rectangular tetrahedron is refined into three similar elements and a rectangular tetrahedron complement (middle), and the rectangular tetrahedron complement is refined into three similar elements, four bricks and one rectangular tetrahedron (right).

3.3 1-irregular elements

1-irregular macro-elements are the elements with at most one Steiner point on each edge. These elements are

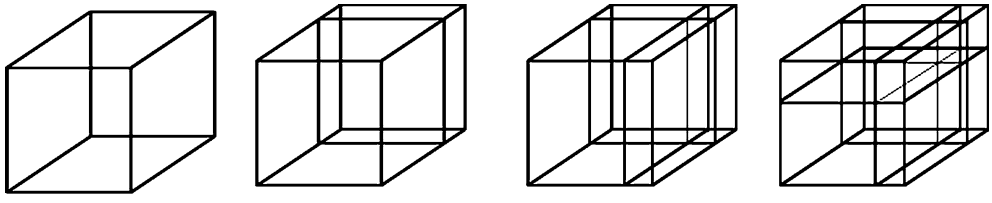


Fig. 2 Brick refinement

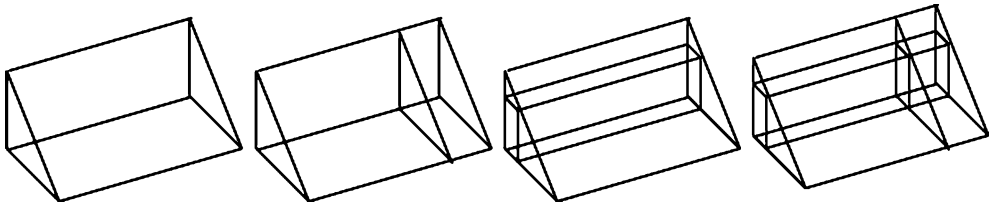


Fig. 3 Prism refinement

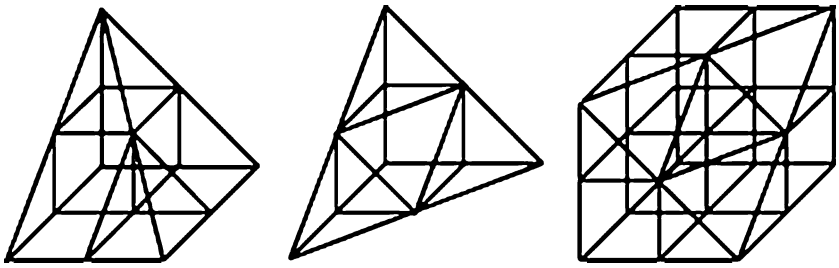


Fig. 4 Refinement of the pyramid, tetrahedron and its complement

generated in order to get a smooth transition between coarse and fine regions of the modeled object.

The next definition characterizes a well-shaped 1-irregular element for the generation of a mixed-element Delaunay mesh.

Definition 3 Let l be a 1-irregular macro-element. l is well-shaped if no Voronoi point of l lies outside its convex hull (outside the macro-element itself).

The mesh generation algorithm proposed below in Section The algorithm inserts points into appropriate edges in case an 1-irregular element is not well-shaped. In order to know if an 1-irregular element is well-shaped or not, the algorithm checks the condition formulated in the next theorem.

Theorem 3 *Let $S \subset R^3$ be a set of points, C the convex hull of S, and T a Delaunay tessellation of S. Then no Voronoi point of S lies outside C if and only if for each face f of T on the surface of C, the circumsphere of f with the center in the middle of f is point-free.*

The detailed proof of this theorem can be found in [9]. Theorem 3 guarantees that the whole mesh satisfies the Delaunay condition because (1) the local tessellation of each 1-irregular element must be Delaunay and (2) the smallest circumsphere of each face on the surface of a 1-irregular element is point-free. Condition 2 implies that elements sharing internal faces are also Delaunay. Figure 5a shows the Delaunay tessellation a well-shaped 1-irregular element and Fig. 5b shows its Voronoi diagram.

Note that the elements without Steiner points also fulfill the Delaunay condition, because (1) they belong to the set of well-shaped macro-elements and (2) the regular point distribution generated by our refinement strategy.

3.4 Final elements

During the tessellation of well-shaped 1-irregular elements, several co-spherical point configurations whose circumsphere is point free are candidates to define a valid final element. In order to avoid the handling of a high number of final elements, our application recognizes as final element the point configurations that fulfill the following restrictions: (1) their convex hull is defined by triangular and/or quadrilateral faces and (2) it is not possible to tessellate the interior of their convex hull into simpler final elements without the insertion of diagonals on the convex hull surface. The current set of final elements is shown in Fig. 6. This set of elements solves the most used 1-irregular configurations of bricks [10]. In case a well-shaped 1-irregular element cannot be tessel-

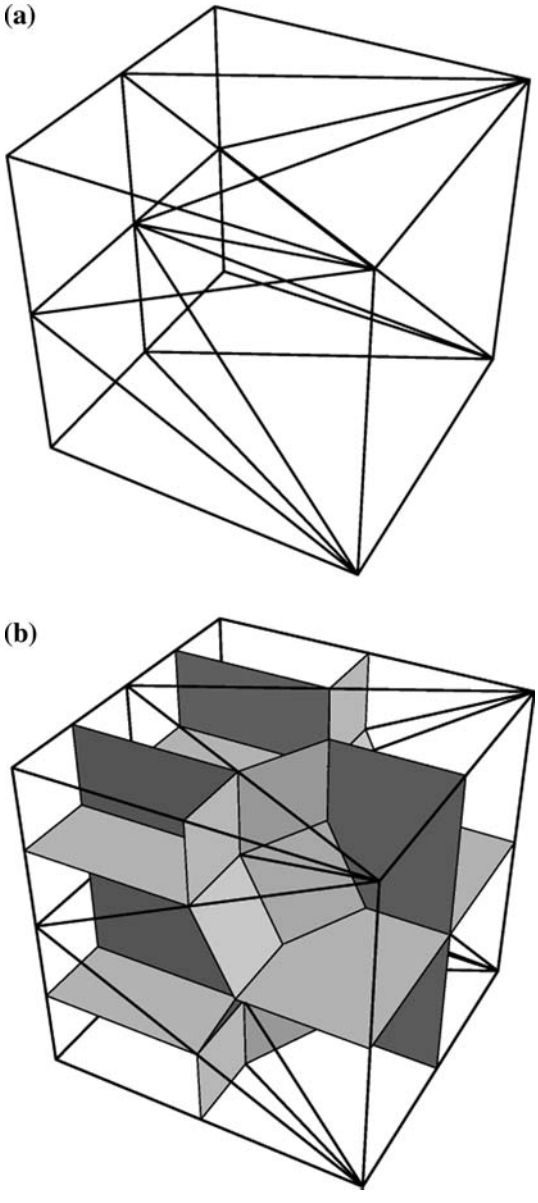


Fig. 5 Well-shaped 1-irregular brick (**a**) Delaunay tessellation (**b**) Voronoi diagram

lated into this set of elements it is handled as a non well-shaped 1-irregular element. This means, points are inserted into appropriate edges until this element can be tessellated.

It is worth to point out that the final elements shown in Fig. 6 labeled from (a) to (e) are already used in the simulation of semiconductor devices with the control volume method. The two last elements have not been incoporated yet, because they do not appear very frequently.

3.5 The algorithm

The Delaunay mesh generator begins the fitting of the geometry by enclosing the domain with its smallest brick. Brick edges are refined along one, two or three coordinate axes at the most convenient edge point selected from the intersection points between the input geometry and a target element. The intersection points can be located in the interior of the element or on the element edges or faces. In our implementation, each brick contains information about the polygons by which it is intersected. The intersection points are the points that define each one of these polygons. When an intersected brick is refined, the coordinates of the intersection points are stored into three ordered lists: one for the x-coordinates, one for the y-coordinates and one for the z-coordinates. For each nonempty list, the coordinate closest to the respective coordinate of the brick center is selected as candidate to define the refinement point. The candidate that generates the brick sons with better aspect ratio defines the refinement direction and the refinement point. For example, if the candidate is d in the direction x, all brick edges parallel x are divided into two edges by inserting a point whose x-coordinate is d. Once the two brick sons are generated, the polygons that intersect the target brick are cut in order to assign to each brick son only the part of the geometry by which it is intersected. Figure 7 shows a 2D example illustrating the process: the geometry is shown in Fig. 7a, the initial smallest rectangle surrounding the geometry and the first refinement at point p_7 is shown in Fig. 7b. Note that p_7 was selected from all the geometry points that are in the interior of the initial rectangle or are located on one of its edges. The geometry is cut through the line l so that each new rectangle only knows the part of the geometry that intersects it. Figure 7c shows a probably next refinement of the right son, where p_5 was selected to locate the refinement line. This process finishes when all the intersected rectangles can be divided into two triangles or a fractal configuration is recognized. Figure 7d shows a macro-mesh composed of triangles and rectangles, where the triangles/rectangles outside the geometry have been eliminated. Note that the initial macro-mesh is usually not conforming.

Figure 9a shows a 3D macro-mesh composed of bricks, prisms and rectangular pyramids. In a similar way to the 2D case, a brick is refined at the most convenient point selected from the intersection points between the object geometry and the brick itself. The process finishes when all the intersected bricks have the intersection points at the brick corners or a fractal configuration is recognized. If all the intersection points are at brick corners, the brick is exactly fitted using prisms and pyramids of rectangular basis, and rectangular tetrahedra (well-shaped macro-elements). The tessellation of these intersected bricks is done template-based. The simplest intersected bricks are shown in Fig. 8. The detection of fractal configurations is discussed in Section Recognition of macro-elements and fractal configurations. If a fractal configuration is found, the brick is not further refined and in the next steps, it is handled as a cut brick.

After the fitting of the device geometry, each element is refined along the required axis until the point density specified by the user is obtained. Element edges are normally bisected except when their edges already contain Steiner points. From the available Steiner points, the one whose associated refinement generates sons with smallest aspect ratio is chosen. Figure 9b shows the bipolar transistor with the required point density specified by the user.

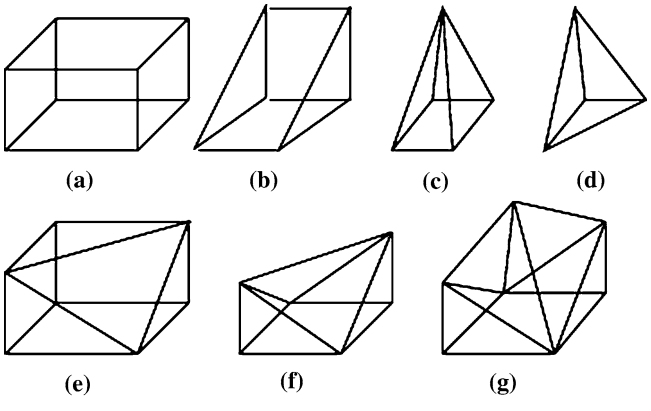


Fig. 6 Set of final elements

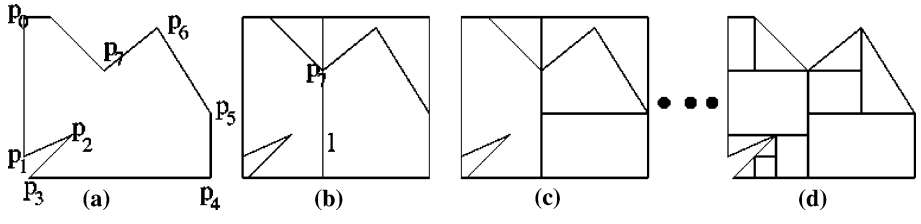


Fig. 7 Fitting a 2D device geometry using the flexible refinement approach. (**a**) The geometry, (**b**), and (**c**) the first steps to fit the device geometry (**d**) initial macro-mesh

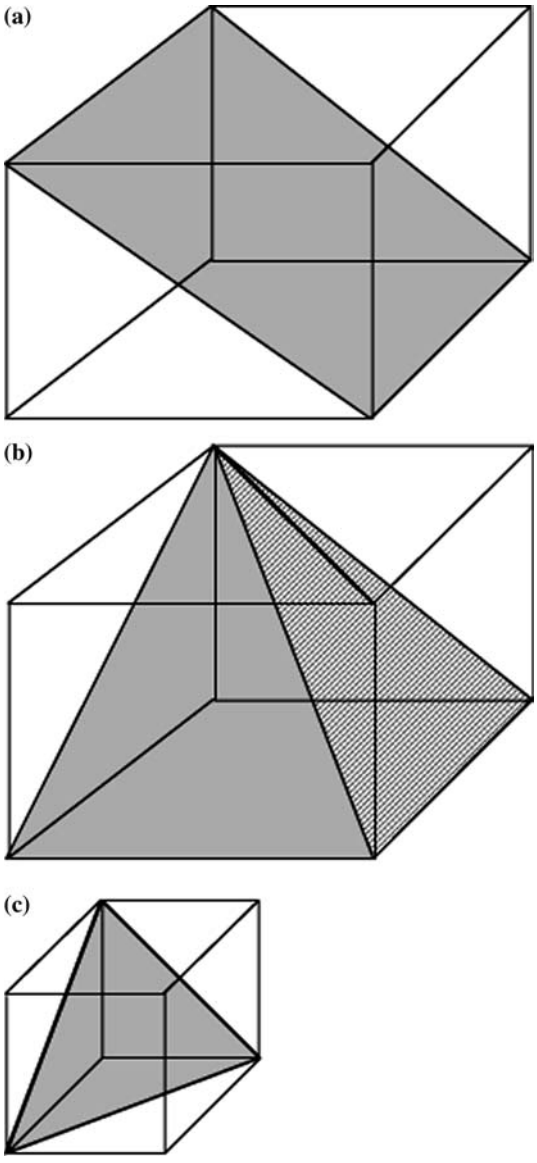


Fig. 8 Intersected bricks

Subsequently, the mesh is done 1-irregular in order to obtain a graded mesh between coarse and fine regions and to generate well-shaped 1-irregular elements (1-irregular elements that satisfy Definition 3). Bad 1-irregular elements are improved by either inserting proper points on specific edges or refining the element in an adequate direction so that the depth of the tree is never incremented. Figure 10a shows a 1-irregular mesh for the bipolar transistor.

Once all 1-irregular elements are well-shaped, each local tessellation is computed either template-based or using an ad-hoc algorithm. A co-spherical set of points, whose circumsphere is point-free, is accepted as final element if its tessellation into (simpler) basic elements requires the addition of new edges (diagonals) or vertices on its surface. A final mesh is shown in Fig. 10b.

3.6 Restrictions for the control volume method

The control volume method is widely used in semiconductor device simulations [11, 12]. It has been proven that a subset of the Delaunay meshes provides a good discretization for this numerical method. Both the Delaunay mesh and its dual the Voronoi diagram are required in the numerical computations. In particular, the Voronoi regions are used as the control volumes to approximate the numerical integration associated to a mesh point. In order to get good simulation results, it is required that internal mesh points are surrounded by a region of only one material [12–14]. This restriction imposes the following condition to boundary/interface elements:

Definition 4 A Delaunay tessellation of a set of points S is adequate for the control volume discretization method if each Voronoi point (circumcenter) of a boundary element is inside of it or is inside a neighboring element through internal faces.

The mesh generator described in the previous section generates Delaunay tessellations. In order to fulfill Definition 4, both macro-elements used to fit the object geometry and boundary/interface elements generated in the tessellation of 1-irregular elements must have their Voronoi point in their interior or inside a neighboring element through internal faces. The elements of the 1-irregular tessellations satisfy Definition 4 because they fulfill Theorem 2. However, not all the macro-elements satisfy it. The macro-elements that satisfy the previous definition without the insertion of new points are the rectangular prism, rectangular pyramid and brick. The brick contains the center of its circumsphere in its interior, and the rectangular prism and pyramid on their surface. (Fig. 11 shows the Voronoi diagram inside each one of these macro-elements.) Then we use only these macro-elements to fit the object geometry. Since the whole mesh fulfills the Delaunay condition, the existence of the Voronoi diagram is guaranteed.

3.7 Comments on the complexity of the algorithm

In the following, we discuss how good and efficient is the previous algorithm in generating a mesh in the context of algorithms based on octrees.

The algorithms to fit the device geometry and to fulfill the density requirements are very efficient. The strategy to fit the object geometry allows that at each refinement step, each new element contains a simpler geometry than its father except when the intersection is a fractal configuration. Then, this process converges and generates a number of elements that depends on the number of points of the input geometry and on the number of geometry edges (faces) that generate new intersection points on edges or faces of the new bricks. Note that the first element refinements produce more new intersection points between the input geometry and the brick edges and faces than the later ones, because the first bricks contain a larger part of the geometry. Each new gener-

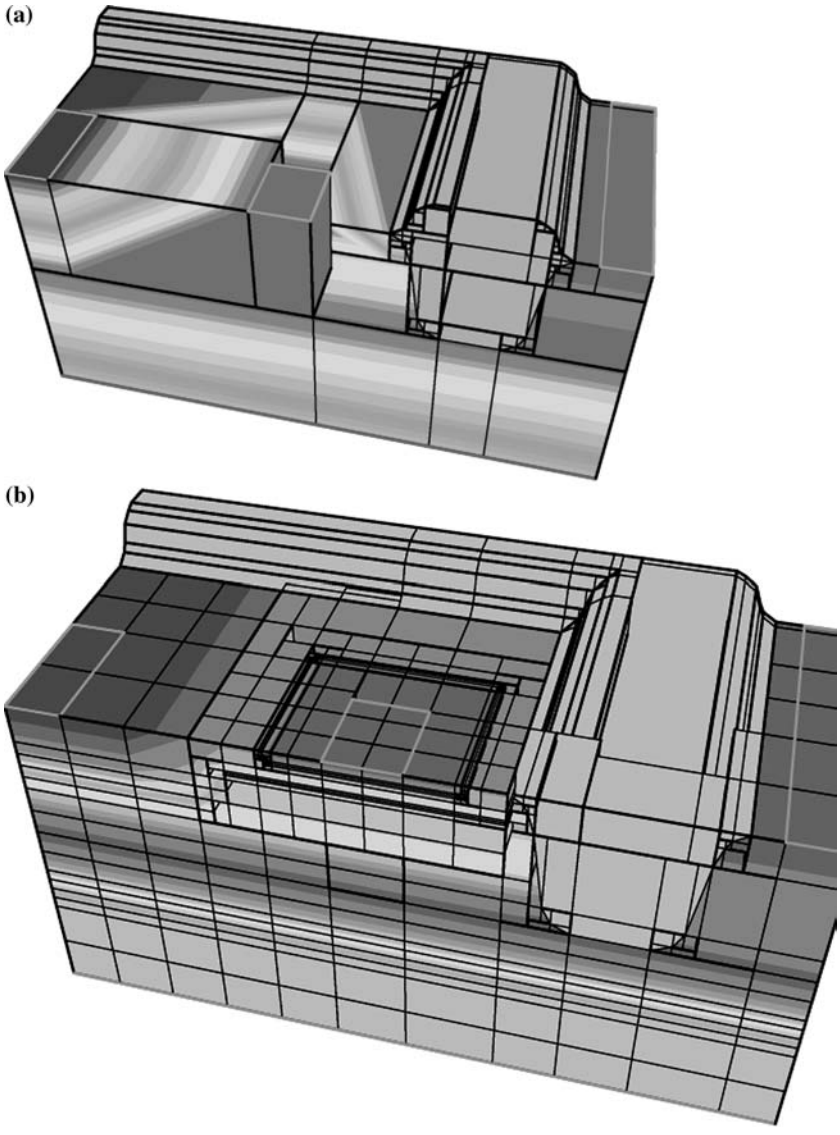


Fig. 9 (**a**) Fitting the device geometry for the bipolar transistor: 554 points and (**b**) getting the desired mesh density: 3,030 points

ated element is visited once. The same occurs in the next step where the density requirements are fulfilled dividing each element into smaller ones. Each element is visited once for each density parameter.

The number of elements generated in the process to get a 1-irregular mesh is also efficient, because each element is refined in a certain direction only if in this direction one of its edges has more than one Steiner point. However, the process to generate a well-shaped 1-irregular mesh still introduces too many points and elements because of: (1) our improvement strategy uses only local information to decide the refinement of a bad 1-irregular element. Thus, the new elements are closer to be well-shaped elements, but the ones of the neighborhood might be deteriorated, (2) Definition 3 is more restrictive than it is necessary in order to decide locally if an 1-irregular element is well-shaped.

Once all the 1-irregular elements are well-shaped, the number of generated elements is optimal because the tessellation algorithm usually does not divide co-spherical point configuration into smaller elements. If meshes for the control volume method are required, the computation of the Voronoi diagram is linear on the number of edges. This is done by traversing once the leaves of the mixed element tree after the final mesh was generated.

4 Implementation issues

This section describes key aspects in the design and implementation of the algorithm described in the previous section.

4.1 Recognition of macro-elements and fractal configurations

This section describes the conditions used during the generation of the initial mesh to detect fractal configurations and to detect intersected bricks that can be tes-

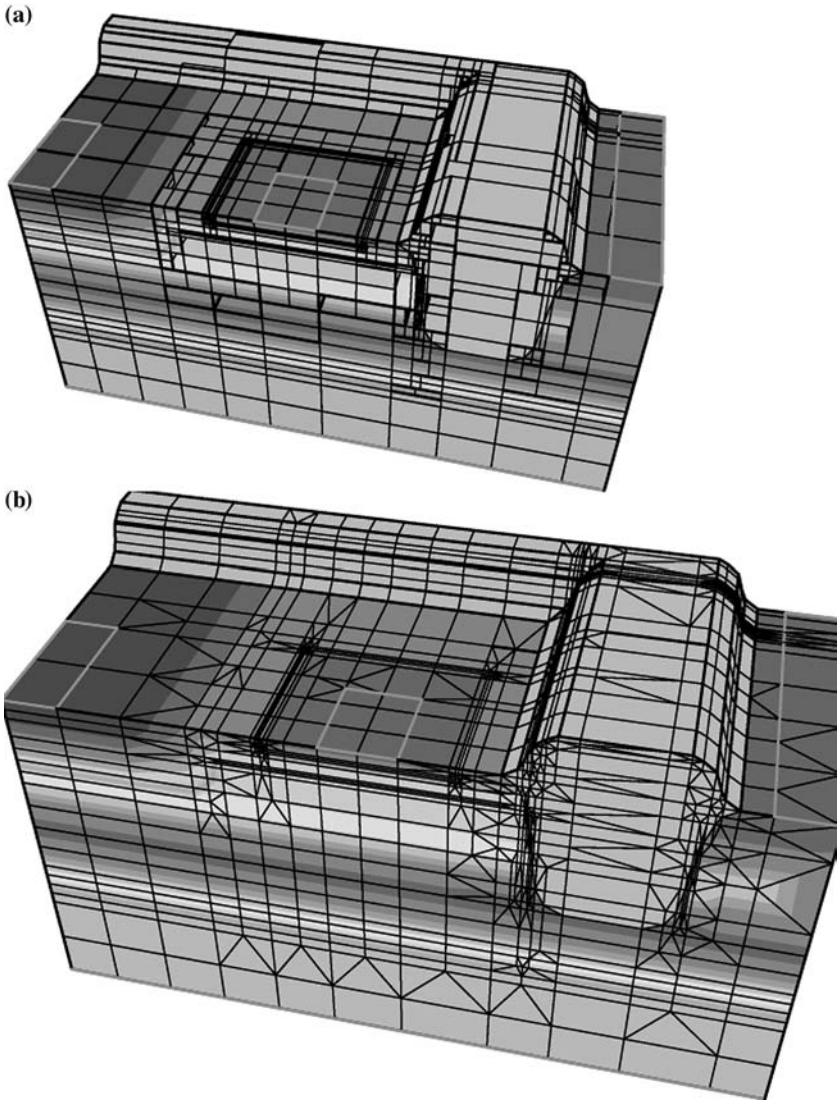


Fig. 10 (**a**) Making the mesh density 1-irregular: 6,230 points and (**b**) final mesh: 11,403 points

sellated into macro-elements. We first describe the conditions used for the generation of 2D initial meshes, because it simplifies the description of the conditions used in 3D.

A fractal configuration is defined as follows:

Definition 5 Let C_0 be a rectangle in 2D or a brick in 3D and G_0 the object geometry intersecting C_0. The pair (C_0, G_0) is a fractal configuration if after i refinements of C_0, a new pair (C_i, G_i) can be transformed to (C_0, G_0) using scale or mirroring transformations.

4.1.1 2D fractal configurations

Let $C_0 = (x_c, y_c, d_x, d_y)$ be a rectangle defined by the bottom left corner (x_c, y_c) and the top right corner $(x_c + d_x, y_c + d_y)$. Let $S = \{e_0, \ldots, e_{n-1}\}$, $n > 0$ be a set of segments of the geometry G_0 that intersects C_0.

Definition 6 The rectangle C_0 is tessellated into macro-elements if and only if the endpoints of each segment e_i is located at a corner of the rectangle C_0.

Figure 12 shows the two kinds of intersected rectangles: the rectangle at the left of the figure is completely inside or outside the object geometry and the rectangle at the right is cut by a segment. The right rectangle is divided into two triangles. The segments that coincide with some rectangle edges are used to find the region to which a macro-element belongs.

It is known that the pair (C_0, G_0) that has a fractal behavior in 2D is the one shown in Fig. 13a. The fractal behavior can be observed after the orthogonal refinement at the intersection point P_0. This refinement introduces a new intersection point P_1 as shown in Fig. 13b. Thus, the lowest rectangle C_1 in Fig. 13c is intersected by a mirroring configuration of the geometry shown in Fig. 13a.

Theorem 4 *A 2D geometric configuration is a 2D fractal configuration if and only if:*

1. *All the segments of G_0 that intersect the interior of C_0 share an endpoint.*

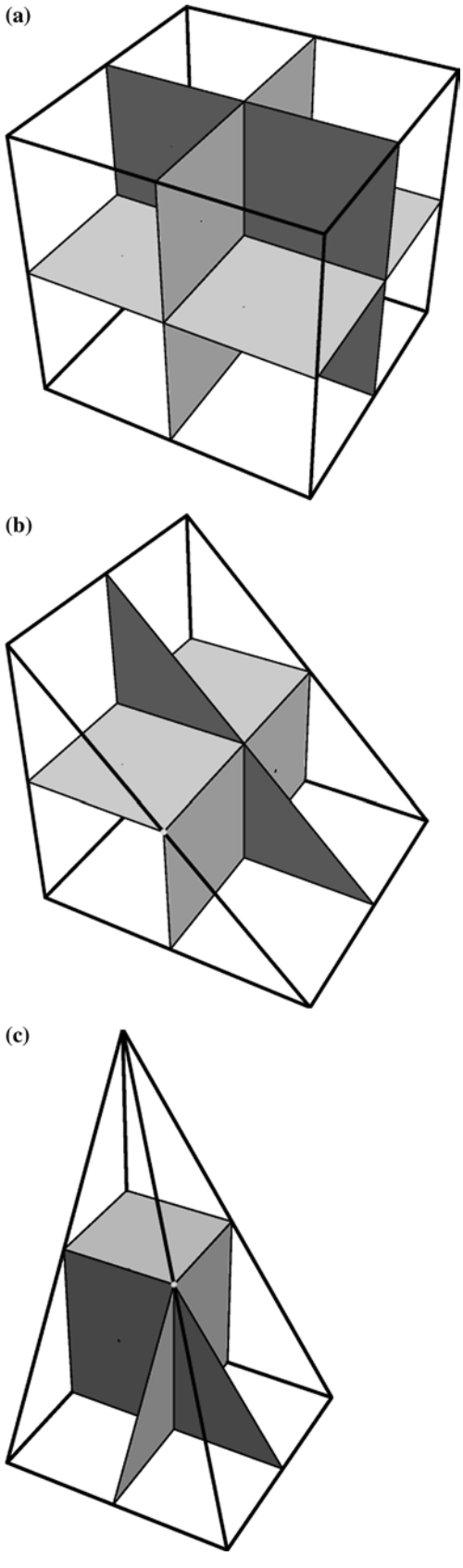


Fig. 11 Macro-elements and their Voronoi regions

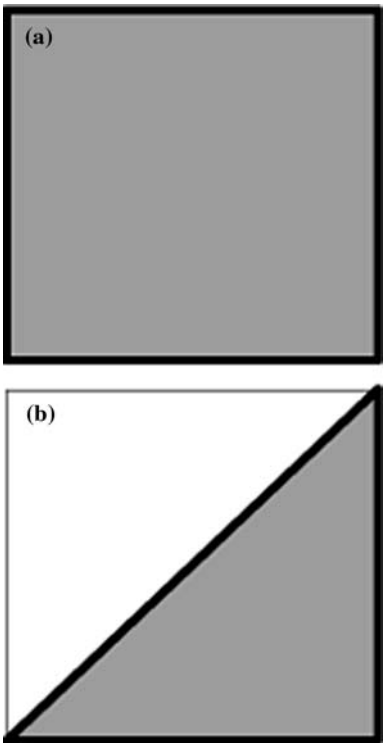


Fig. 12 Tessellations of a cut rectangle into 2D macro-elements

2. *Let* p_0 be the shared endpoint and let us assume that p_0 *coincides with the bottom left corner* (x_c, y_c) *of* C_0. *The endpoints* p_1 *and* p_2 *of the segments* $e_1 = (p_0, p_1)$ *and* $e_2 = (p_0, p_2)$ *must be located on the edges of* C_0 *whose endpoints do not include* (x_c, y_c). *This condition can be formulated as follows:* $(p_1 - p_0)^t (p_1 - p_0) = d_x^2 + d_1^2$ *where* $0 < d_1 \leq d_y$, *and* $(p_2 - p_0)^t (p_2 - p_0) = d_2^2 + d_y^2$ *and* $0 < d_2 \leq d_x$. (p_1 *can replace* p_2 *and vice versa*).

This condition guarantees that the endpoints p_1 and p_2 are located on different edges of the rectangle. The endpoints can only be on the same edge if one of them is located at the corner $(x_c + d_x, y_c + d_y)$.

4.1.2 3D fractal configurations

The conditions formulated to detect if an intersected rectangle can be tessellated into macro-elements and if an intersected rectangle contains a fractal configuration can be extended to 3D.

Let $C_0 = (x_c, y_c, z_c, d_x, d_y, d_z)$ be a brick with bottom left front corner (x_c, y_c, z_c) and top right back corner $(x_c + d_x, y_c + d_y, z_c + d_z)$. Let $PL = P_0,\ldots,P_{n\text{-}1}$ be the list of polygons that intersects C_0 . Each P_i is defined by an ordered list of segments $e_{io},\ldots,e_{ik}$.

Definition 8 An intersected brick will be tessellated into macro-elements if each endpoint of the polygon segments is located at one of the brick corners.

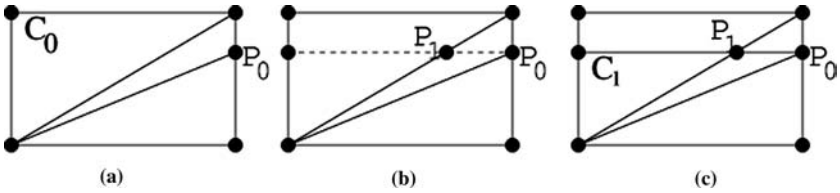


Fig. 13 (**a**) 2D fractal configuration, (**b**) (C_0,G_0) and (**c**) (C_1,G_1)

The Fig. 8 of the Section The algorithm shows some examples of intersected bricks and their tessellations. The recognition of which macro-element fits a specific intersected brick is done template-based.

Before we specify the geometric conditions that appear in a 3D fractal configuration, let us use Fig. 14 to illustrate some examples of 3D fractal configurations. Figure 14a shows an intersected brick where two of its faces contain a 2D fractal configuration. Figure 14b shows a brick intersected by a geometry whose orthogonal projection on four of the six brick faces is a 2D fractal configuration.

The next theorem describes the condition to recognize a fractal configuration in 3D.

Theorem 5 *A 3D geometric configuration is a 3D fractal configuration if and only if:*

1. *At least two segments e_1 and e_2 of G_0 that intersect the interior of C_0 share an endpoint.*
2. *Let p_0 be the shared endpoint and let us assume that p_0 coincides with the bottom left front corner (x_c, y_c, z_c) of C_0. The endpoints p_1 and p_2 of the segments $e_1 = (p_0, p_1)$ and $e_2 = (p_0, p_2)$ must be located on the edges of C_0 that do not contain (x_c, y_c, z_c) as endpoint. This condition can be formulated as follows: $(p_i - p_0) = (x_i, y_i, z_i)$, $i = 1,2$ where at least two of the coordinates of (x_i, y_i, z_i) must be greater than 0. Furthermore, p_1 and p_2 must be located on different edges of the brick. This implies that $(x_1, y_1, z_1) - (x_2, y_2\ z_2)$ must have two or more coordinates not equal to 0 except when p_1 or p_2 is located at a brick corner.*

The condition formulated in this Theorem is only checked after all segments endpoints that intersect a brick are located on boundary edges or corners of the brick. In case there exist endpoints in the interior of a brick or in the interior of a brick face, the brick is refined as it was explained in the Section Algorithm.

4.2 Basic algorithms and data structures

This section describes the data structures and algorithms designed to implement the mixed element trees and to handle the flexible refinement approach in a efficient way.

4.2.1 Mixed element trees

The mesh elements are stored in a mixed element tree. A mixed element tree is defined as follows:

Definition 6 Let T be a tree. T is a mixed element tree if

1. *each node (internal or leaf) is a macro-element*
2. *each internal node is labeled with the axes across which the node is refined.*

The root of the tree is the smallest brick that surrounds the object geometry. The number of sons of each internal node (macro-element) depends on the axes the node was refined and on the macro-element type. For example, a brick refined through the x-axis has two sons; a brick refined through the x- and y-axes has four sons, and a refined pyramid has always five sons.

Theorem 6 *Let T be a mixed element tree. T leads to well-shaped meshes by construction if and only if*

1. *each internal node is one of the macro-elements described in the Section Macro-elements*
2. *each leaf is an 1-irregular macro-element satisfying Definition 3 or a macro-element without Steiner points and*

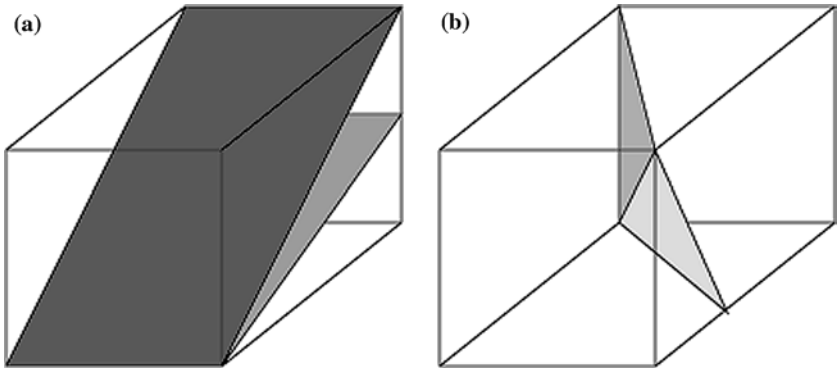


Fig. 14 (**a**) The 2D fractal configuration observed in two brick faces and (**b**) a 3D fractal configuration

3. *each 1-irregular leaf can be tessellated into the final elements.*

This representation allows the generation of well-shaped meshes by construction because it permits the implementation of the concepts introduced in the previous sections in a natural way.

4.2.2 Refinement algorithm

Our mesh refinement strategy has the following characteristics: (1) It is depth first, i.e., the last generated element is refined first. Then, after some refinements elements that were originally face neighbors can lie at very different tree depth. (2) Neighboring elements might not be refined at the same location. An element can have several Steiner points on its edges because of the previous refinement of its neighbors. The decision of which plane will be used to cut an element depends on (1) the part of the geometry that intersects the element, (2) the Steiner points already available, and (3) the aspect ratio of the possible sons. Let us use a 2D example to illustrate the previous ideas. The case (1) is illustrated in Fig. 15. The rectangle **A** is refined into two smaller rectangles **B** and **C**. The rectangle **B** is refined again into **D** and **E**, and the rectangle **E** is refined in **F** and **G**. The rectangles **D**, **F** and **G** are elements in the neighborhood of the rectangle **C** and **F** and **G** are located two levels deeper than **C**.

Figure 16 illustrates the case (2). First, the rectangle **A** is refined into two smaller rectangles **B** and **C**. **B** and **C** share edge ($\mathbf{p_4}$, $\mathbf{p_5}$). Then, the refinement of the rectangle **B** generates the point $\mathbf{p_6}$ on edge ($\mathbf{p_4}$, $\mathbf{p_5}$) and the refinement of rectangle **C** generates the point $\mathbf{p_7}$ on the same edge. In order to be able to refine these new elements further, edge ($\mathbf{p_4}$, $\mathbf{p_7}$) must have access to $\mathbf{p_6}$ and edge ($\mathbf{p_6}$, $\mathbf{p_5}$) must have access to $\mathbf{p_7}$.

3D elements are refined by inserting simultaneously three or four points on their edges. These points define a plane whose normal is always one of the main axes.

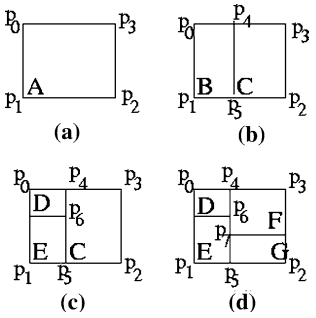
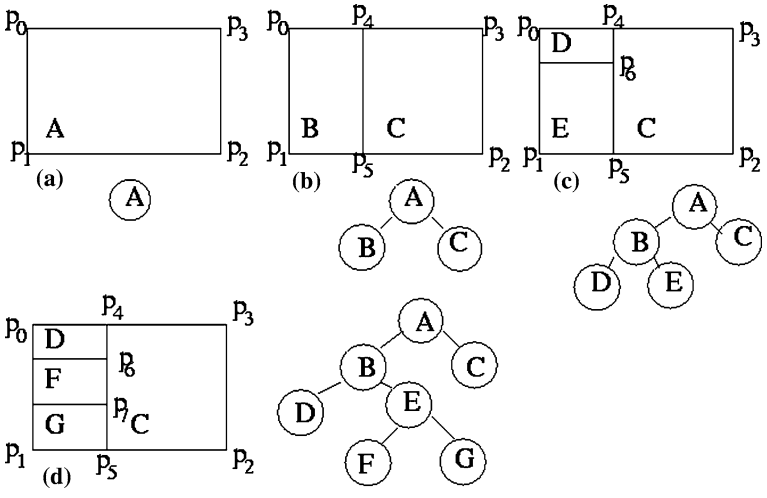


Fig. 16 Refinement of a rectangle

Faces are shared at most by two elements but edges can be shared by several ones. The refinement of an element is done by the refinement of its edges and faces. Each edge is refined by inserting a point between its endpoints and faces by inserting an edge parallel to one or two face edges.

In order to implement our mesh generation strategy, the algorithm needs to know all the points previously inserted on its edges. The edge data structure designed to efficiently and consistently handle this information is the following: for each edge ($\mathbf{v_1}$, $\mathbf{v_2}$) only one inserted point **p** is stored explicitly between its endpoints. The rest of the points must be visible from ($\mathbf{v_1}$, **p**) and (**p**,$\mathbf{v_2}$). The next inserted points are stored in one of the descendants of ($\mathbf{v_1}$, $\mathbf{v_2}$). The algorithm is illustrated in Fig. 17. Figure 17a shows the case when a point *p* is inserted on an edge ($\mathbf{v_1}$, $\mathbf{v_2}$) without Steiner points. Two new edges are generated ($\mathbf{v_1}$, **p**) and (**p**, $\mathbf{v_2}$) and p is stored as the first point of edge ($\mathbf{v_1}$, $\mathbf{v_2}$). Figure 17b shows the case where edge ($\mathbf{v_1}$, $\mathbf{v_2}$) has already stored a Steiner point **gp** and the point **p** to be inserted lies between **gp** and $\mathbf{v_2}$. The point **p** is inserted recursively between **gp** and $\mathbf{v_2}$. Note that if edge (**gp**, $\mathbf{v_2}$) has no Steiner point, **p** is inserted here. Otherwise, the point **p** is passed through its edge descendants until one of them has no Steiner point on it. When the recursive call finishes, a new edge is created with endpoints ($\mathbf{v_1}$, **p**) and first Steiner point **gp**. The edge

Fig. 15 Neighboring elements. Refinement of a rectangle and the associated tree structure



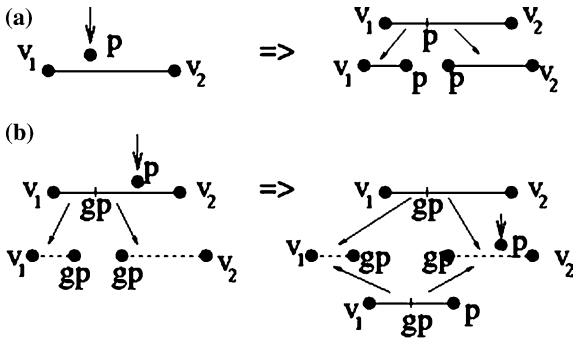


Fig. 17 Algorithm to insert a point on an edge

($\mathbf{v_1}$, **p**) must be created because it belongs to one of the sons of the refined element. The insertion of **gp** on ($\mathbf{v_1}$, **p**) guarantees that the sons of the refined element that share ($\mathbf{v_1}$, **p**) would have access to all the points already inserted between their endpoints. Figure 16b has a symmetrical case when the point **p** is inserted between edge ($\mathbf{v_1}$, **gp**).

Note that behind the edge data structure, there is an implicitly binary search tree. In order to look if a Steiner point **p** is already inserted between $\mathbf{v_1}$ and $\mathbf{v_2}$, **p** is compared with **gp**. If **p** is equal to **gp**, the search finishes. If **p** is between ($\mathbf{v_1}$, **gp**), the search is continued inside ($\mathbf{v_1}$, **gp**). Otherwise, the search is done inside (**gp**, $\mathbf{v_2}$). Then, the search or insertion of a point *p* is done O(ln(n)), in average, where n is the number of Steiner points of the edge.

A similar data structure and algorithms were designed and implemented for inserting an edge on a element face [6].

4.3 Recognizing final primitive elements

Final elements are recognized using the two common strategies adapted to the recognition of several element types: (1) template-based, i.e, a tessellation is obtained from a table of pre-computed tessellations of the most generated 1-irregular elements. This technique is applied to 1-irregular configurations, whose Steiner points are edge midpoints, and (2) the use of an ad-hoc algorithm to compute the local tessellations. This algorithm is used for the well-shaped 1-irregular macro elements that are not solved using a template-based approach.

The template-based strategy uses the fact that several 1-irregular configurations (patterns) are permutations of the same topology. The patterns have been then classified into equivalence classes or pattern types. Patterns and pattern types are stored in tables. Each pattern is associated with a pattern type and a permutation of its corners in order to be mapped to the pattern type. A pattern type is associated with its Delaunay tessellation and the required eccentricity condition where this tessellation is valid. The number of possible patterns generated by using only edge midpoints is 2 number-of-edges. For example, in case of a brick, there are 2^{12} 1-irregular configurations. An integer code obtained from the split edges of a 1-irregular element is used to index the pattern table and so to obtain the pattern type and the corner permutation. Then, the pattern type is used to index the pattern type table and to obtain by using the corner permutation information the corresponding tessellation. Currently, the pattern type table contains 25 equivalence classes. The total number of equivalence classes for 1-irregular configurations defined by edge midpoints is 144 [8]. Then, it is possible to handle all of them in a table. A lower bound for the number of equivalence classes for 1-irregular configurations with Steiner points at any position is 34,058. The total number of 1-irregular configurations is 187^3 [8]. Note that it is not possible to have a template based approach in this case. The advantage of a template-based approach over an ad-hoc algorithm is that it is more efficient (tessellation in constant time) and robust.

The ad-hoc algorithm computes the Delaunay tessellation with the lowest number of elements under the following conditions: (1) it never divides quadrilateral faces into triangles, (2) it obtains all the co-spherical points (whose circumsphere is point-free) before an element is built.

The algorithm can be summarized in the following steps: (1) each 1-irregular face on the surface of a 1-irregular element is divided into triangles and/or rectangles. (2) Final elements with a face on the surface of the 1-irregular element are recognized first. (3) A final element is built from a target face whose vertices and some other vertices of the 1-irregular element define a point-free circumsphere (Delaunay condition). (4) Depending on the type of the target face (triangle or quadrilateral) and the selected number of vertices, one of the final elements is recognized. For example: a triangular target face and one vertex form a tetrahedron, a rectangular target face and one vertex form a pyramid, a rectangular target face and two vertices can form a rectangular prism or a deformed prism.

5 Examples

In order to illustrate which kind of meshes can be generated using the approach described in this paper, Fig. 18 shows a mixed element mesh for two known semiconductor devices: the ecl bipolar transistor and the locos. In addition, Table 1 shows the number of mesh elements of each type shown in Fig. 6.

6 Conclusions

The mixed element mesh generator described in this paper can be used in several geometric modeling applications, in particular in applications that require a Delaunay mesh. Mixed element meshes reduce strongly the number of element and edges in comparison with meshes that only use tetrahedra.

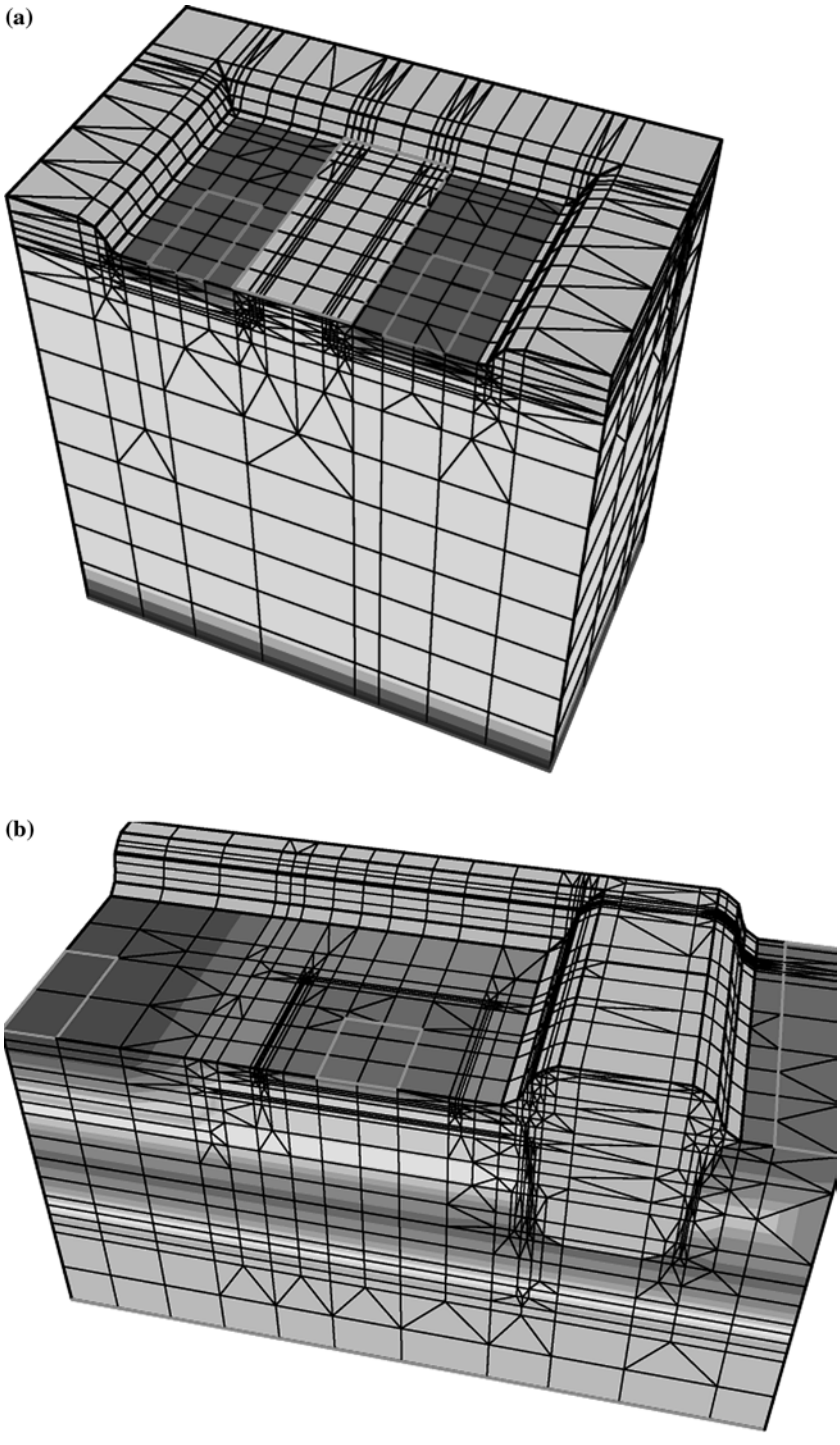


Fig. 18 (**a**) Locos: 4,939 points (**b**) ecl bipolar transistor: 11,403 points

The current implementation can generate Delaunay meshes for any geometry that can be represented using bricks, pyramids, prisms, tetrahedra and complement of tetrahedra. In addition, it can generate control volumes meshes for any geometry that can be represented using bricks, prisms and pyramids.

Table 1 Number of elements of each type in the final meshes

	Brick	Prism	Pyramid	Tetrahedron	Tetrahedron complement	Total
ecl	1,418	2,029	7,880	4,612	570	16,509
Locos	701	1,089	3,471	1,912	187	7,630

The flexible refinement approach allows to represent the geometry exactly and with fewer macro-elements than a bisecting approach, but it requires more sophisticated data structures and algorithms to obtain an efficient and robust implementation.

Currently, we are working on (1) the design of strategies to generate local Delaunay meshes inside

fractal configurations that satisfies the control volume method restrictions, (2) a generalization of the algorithm that generates well-shaped 1-irregular elements. The tessellation of internal 1-irregular elements must only fulfill the Delaunay condition and does not require the additional restriction that the Voronoi points must be inside 1-irregular element itself. This algorithm should strongly reduce the number of points generated in the step 1-irregular elements are done well-shaped.

Acknowledgments This work was partially supported by project N° 1030672. The visualization tool Picasso belongs to ISE-AG Switzerland.

References

1. Yerry M, Shephard M (1984) Automatic three-dimensional mesh generation by the modified-octree technique. Int J Num Methods Eng 20:1965–1990
2. Shephard M, Georges M (1991) Automatic three dimensional generation by the finite octree technique. Int J Num Methods Eng 32:709–749
3. Schroeder W, Shephard M (1990) A combined octree/Delaunay method for fully automatic 3-D mesh generation. Int J Num Methods Eng 29:37–55
4. Hitschfeld N (1993) Grid generation for three-dimensional non-rectangular semiconductor devices. PhD Thesis, ETH Zürich. Series in Microelectronics, 21, Hartung-Gorre Verlag, Konstanz
5. Garretón G (1999) A hybrid approach to 2D and 3D mesh generation for semiconductor device simulation. PhD Thesis, ETH Zürich, Series in Microelectronics, 80. Hartung-Gorre Verlag, Konstanz
6. Hitschfeld N (1995) Algorithms and data structures for handling a very flexible refinement approach. In: Proceedings of the 4th international meshing roundtable, Sandia National laboratories, Albuquerque, pp 265–276
7. Hitschfeld N (1997) Generalization of modified octrees for geometric modeling. In: geometric modeling: theory and practice. The state of the art. Springer, Berlin Heidelberg New York, pp 260–272
8. Hitschfeld N, Navarro G, Farías R (2000) Tessellations of cuboids with Steiner points. In: Proceedings of the 9th international meshing roundtable, New Orleans, pp 275–282
9. Hitschfeld N, Fichtner W (1993) 3-D Grid generator for semiconductor devices using a fully flexible refinement approach. In: Simulation of semiconductor devices and processes,vol 5. Springer-Verlag, pp 413–416
10. Hitschfeld N, Farías R (1996) 1-irregular element tessellation in mixed element meshes for the control volume discretization method. In: Proceedings of the 5th international meshing roundtable, Pittsburgh, pp 195–204
11. Bank R, Rose D, Fichtner W (1983) Numerical methods for semiconductor device simulation. IEEE Trans Electron Dev 30:1031–1041
12. Pinto M (1990) Comprehensive semiconductor device simulation for silicon ULSI. PhD Thesis, Stanford University
13. Hitschfeld N, Conti P, Fichtner W (1993) A mixed element mesh generator for the simulation of complex 3d devices. IEEE Trans on CAD/ICAS, 12(11):1714–1725
14. Miller G, Talmor D, Teng S, Walkington N, Wang H (1996) Control volume meshes using sphere packing: generation, refinement and coarsening. In: Proceedings of the 5th international meshing roundtable, (Pittsburgh, Pennsylvania), pp 47–61