3D NOffset mixed-element mesh generator approach

Claudio Lobos and Nancy Hitschfeld-Kahler
Depto Ciencias de la Computación,
FCFM, Universidad de Chile,
Blanco Encalada 2120,
Santiago - Chile, Zip code: 837-0459
clobos@dcc.uchile.cl, nancy@dcc.uchile.cl

ABSTRACT

In this paper we present a new approach to generate a mixed mesh with elements aligned to boundary/interfaces wherever is required. A valid element is: (a) any convex co-spherical element that fulfills the requirements of the underlying numerical method and (b) any element that satisfies domain specific geometric features of the model. The algorithm is based on the normal offsetting approach to generate coarse elements aligned to the boundary/interfaces. Those elements are later refined to accomplish layer density requirements. The main steps of the algorithm are described in detail and examples are given to illustrate the already implemented parts. As far as possible, we contrast this algorithm with previous approaches.

Keywords: mixed-element meshes, normal offsetting approach, advancing front.

1 INTRODUCTION

CVM-conforming Delaunay meshes are Delaunay meshes where the circumcenter (center of the circumcircle of an element) of each boundary/interface element is inside the element itself or inside a neighboring element through internal edges/faces. The previous restriction guarantees that when Voronoi regions are used as control volumes in the control volume method (CVM), a mesh satisfies the conditions for the numerical integration around each boundary or internal point.

CVM-conforming Delaunay meshes are used in several applications, but in particular in the simulation of semiconductor devices. In this application, the meshes must also fulfill two additional requirements: due to the geometry of the devices, they should properly model very thin layers, and due to physical properties, edges parallel to the current flow are desirable.

Octree-based and mixed-element tree-based(MET) mesh generators have been developed for the generation of 3D mixed element CVM-conforming Delaunay meshes [2, 4]. The MET approach generalizes the modified octree approach [8, 6] in several aspects: (1) it considers the whole device no longer encapsulated in a single octree, but partitioned in a set of basic elements: cuboids, rectangular prisms and pyramids; each basic element becomes the root of a tree, (2) elements are either bisected or refined by introducing appropriate edge points in order to get the density requirements and (3) a Delaunay tessellation that includes tetrahedra, prisms, pyramids and other basic elements [3] is generated. The main advantage of this approach is the use of several element types that naturally fit the requirements of the CVM. The main drawback is that it can not efficiently generate element faces aligned to the boundary/interfaces and it generates too many points if the device geometry is not well-oriented.

The normal-offsetting approach has been used to generate 3D tetrahedral meshes for the CVM with element edges aligned to the boundary [5]. In this approach, the surface and interfaces of the device are triangulated first and then, the front is moved, step by step, to generate almost prismatic elements. These elements are later divided into tetrahedra. In order to generate a CVM-conforming Delaunay tetrahedral mesh, bad-shaped tetrahedra are divided by orthogonal refinement or by applying a more complex strategy in case the orthogonal refinement does not work [7]. The main advantages of this approach is that it generates edges aligned to the boundary, wherever required. The main drawback is that the improvement of bad-shaped tetrahedra along the boundary/interfaces might require the insertion of a high number of points [7, 4]. In addition, the generation of each prismatic element is slow because it requires a front expansion computation.

In this paper we present an algorithm that takes the advantages of the mixed element tree and of the normal offsetting approaches and improves their drawbacks. First, it includes as valid mesh elements all the convex co-spherical elements that fulfill the requirements of the CVM. Cuboids, in particular, are very useful to repre-

Short Communications proceedings ISBN 80-86943-05-4
WSCG'2006, January 30 – February 3, 2006
Plzen, Czech Republic.

sent very thin layers. Tetrahedra are only used where other element types cannot be used. Second, it uses the normal offsetting approach to generate coarse elements edges aligned to the boundary. Coarse elements are later refined to generate the required layer density specified together with the front information. We conjecture that the refinement of an element involves less, more robust and simpler operations than the ones involved in the computation of a front expansion. The main steps of the algorithm are described in detail and several examples are shown.

2 BASIC CONCEPTS

2.1 NOffset

Normal Offsetting (NOffset) is a particular case of the advancing front technique [1]. NOffset adds the constraint that the expansions must be parallel to the fronts.

In order to compute the expansion of a front face in a distance h, the new position of each face point is calculated. The new position of a point P depends on: (a) the normal vector n of the face, (b) a distance h and (c) the fact that other geometry faces that share P can also be front faces or not.

Let k be the number of front faces that share the point P, n_i be the normal vector of each front face and h_i be the distance of expansion associated to each front face. The new point P' is calculated following one of the next rules [5]:

1. Only one front face contains P (k=1). Then $P' = P + n_1 h_1$
2. Two front faces share P (k=2). Then $(P'-P)\cdot n_i = h_i \mid i = \{1,2\}$ and $(n_1 \times n_2)\cdot(P'-P) = 0$.
3. Three front faces share P (k=3). Then $(P'-P)\cdot n_i = h_i \mid i = \{1,2,3\}$
4. Four or more faces share P ($k > 3$). Then, the nearest point to all intersection planes is computed. Our work does not consider this case yet.

Figure 1 shows an example where the new positions of P and Q (P' and Q', respectively) are obtained by applying rule 2.

3 ALGORITHM DESCRIPTION

In order to generate an adequate mixed element mesh, we have divided this process in the following steps:

- Generation of a coarse anisotropic mixed element discretization
- Generation of a primitive mesh.
- Fulfilling the required layer density
- Fulfilling the required maximum edge length

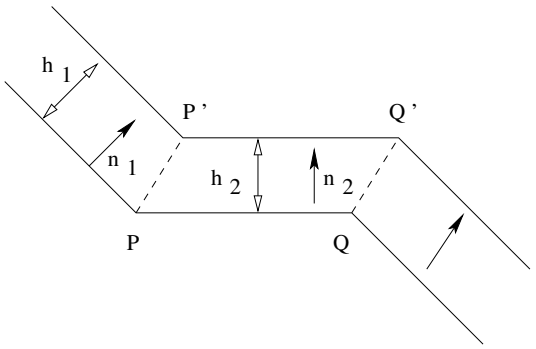

Figure 1: The initial edge PQ is expanded in a distance h_2.

- Generation of a CVM-conforming mixed element mesh

The next subsections describe each of these steps.

3.1 Generating a coarse mixed element discretization

This first step generates coarse anisotropic elements aligned to the boundary or interfaces according to the front information specified by the user. Two inputs are required: the initial geometry to be meshed and a file where the user specifies the fronts with the following information [5]:

- A list of original geometry faces that conform the front.
- Thickness of first layer (*hloc*).
- Coarsening factor of the next layers (*factor*).
- Number of layers (*endline*).
- Maximum edge length of the generated elements (*mel*).

For each front face only one coarse element is generated. The thickness (h_{final}) of each one of these elements is obtained from the number of layers and coarsening factor as follows:

$$h_{final} = hloc \cdot \sum_{i=0}^{t-1} factor^i \quad \mid \quad t = endline$$

The geometry is specified by a set of polyhedral elements and a subset of the faces that define those elements conform the fronts. Each face is part of just one front.

Let us use Figure 2 to illustrate the algorithm to generate one coarse anisotropic element. Figure 2(a) shows a truncated prism with two front faces: the top one and the right one. Let F_n be the new face obtained by the expansion of the top front face. Figure 2(b) shows the

points of F_n. It can be observed that the position of the points to the right was obtained applying the rule 2 and the position of the points to the left was obtained applying rule 1. If each front face point is shared by only front faces, F_n is already in the right position. But if this fact does not occur, some points of F_n may need to be recalculated. Then, the next step is to find the intersection of F_n with adjacent faces that are not front faces. This is the case of the left face of the truncated prism. The new positions of the left points of F_n are shown in Figure 2(c). Once F_n is well defined (Figure 2(d)), the truncated prism is divided into two elements: the first one formed by the top front face, the face F_n, and the lateral faces that join both faces, and the second one formed by the rest of the original truncated prism. This can be observed in Figures 2(e) and 2(f). The process illustrated in Figure 2 is repeated for each polyhedral element whose boundary faces must be expanded.

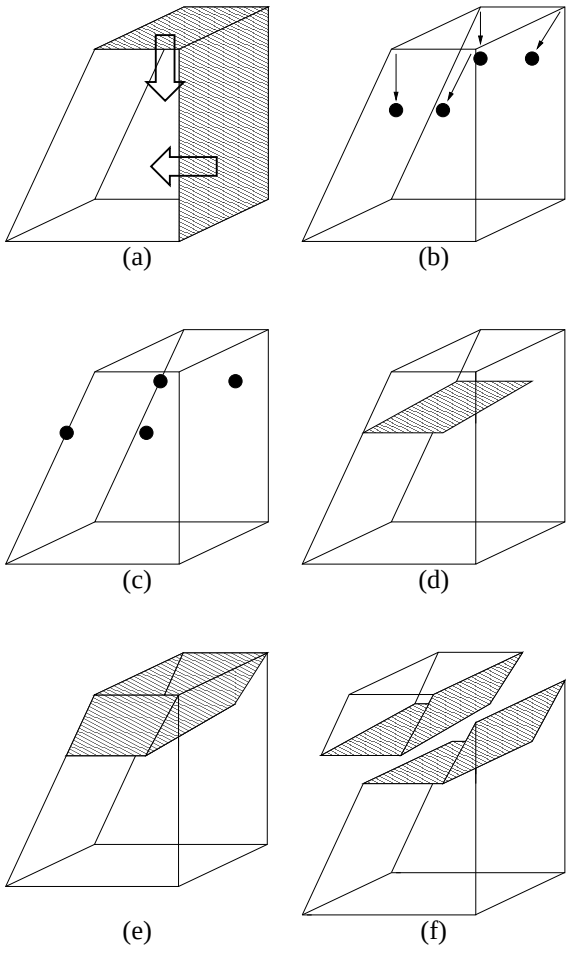

Figure 2: Building a macro element.

The algorithm 1 implements the designed strategy for this step. We assume that each front can be expanded to the maximum distance specified by the user.

Note that when adding a new face it may overlap a previous one. In that case the overlapped face is split into two new ones. If other element shares the same overlapped face, this element is also updated with the two new faces.

Input: front specification file and the geometry
for each face F of each front **do**
  let E be the element that contains F
  for each face point P (not expanded) **do**
    find which rule applies to P
    P' = NOffset of P using the previous rule
  afne = adjacent faces to F that do not expand
  move each point P' to the intersection with afne
  join the points P' to form the new face F_n
  find the type of F_n
  F_j = lateral faces between F and F_n
  E_n = new element defined by F, F_n and F_j
  find the type of E_n
  update element E by E - E_n
Output: coarse discretization of the original geometry

Algorithm 1: Generating a coarse mesh

The previous algorithm shows that the calculation of the NOffset for each point P is a very expensive computational task. That is why we do it only once at the generation process of the aligned elements. This strategy avoids applying the previous algorithm each time the user wants to generate a new parallel layer.

We have left out of the scope the detection and handling of collisions among the fronts that require a global approach. Figure 3 shows three cases of problematic collisions and three possible solutions. Figure 3(a) shows the case where an expanded edge becomes an inverted edge. This problem can be solved by computing this edge expansion until the edge length is zero, and then, as shown in the right picture, the expansion of the neighboring edges to the target one should continue. Figure 3(b) shows an edge expansion that goes out of the original geometry because the current implementation of the algorithm only cut it by the neighboring face. Figure 3(c) shows a case where two different fronts collide. This problem can be solved by testing if the current expansion crashes a previous one. Although these problems has been studied, they are not implemented yet.

3.2 Generating a primitive mesh

At this point we have an initial mesh, that might contain general polyhedra as coarse elements. Since it is difficult to develop a method that is capable of refining any type of element, we would like to have a subset of different kind of elements known as primitives elements: tetrahedra, prism, pyramid, bricks and some truncated variants of them. For this type of elements we are capable of defining a refinement strategy.

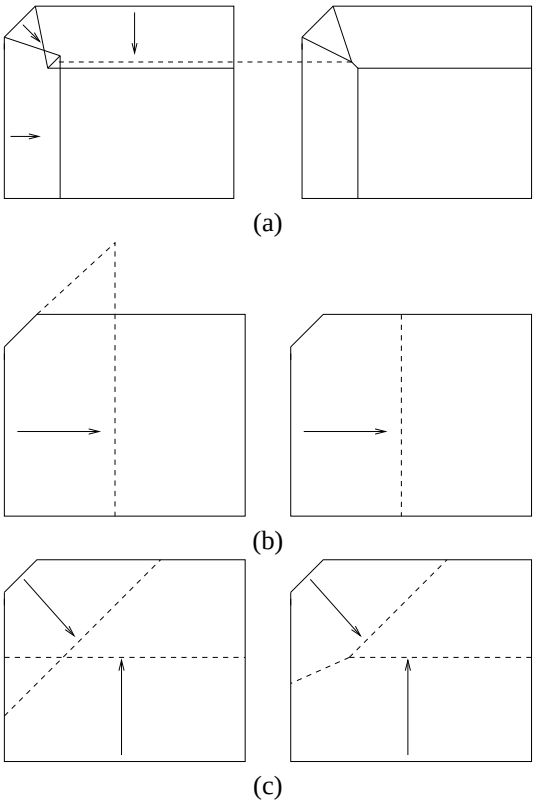

Figure 3: Collisions between expansions and possible solutions

The main idea is to split each coarse element in several primitive elements and refine those elements whenever the density constraints (layer density and maximum edge length) are not accomplish.

Although this step is not very hard to implement it is not done yet, because we have several examples where this is not required. This occurs, for example, when the coarse elements generated after the first step are directly primitive elements. The priority was given to other steps of the algorithm and this was left as part of the ongoing work.

3.3 Fulfilling the required layer density

The input of this step is the initial discretization composed of coarse anisotropic mixed elements aligned to boundary/interfaces and of polyhedrical elements that model the part of the device geometry (cavities) where elements aligned to boundary/interfaces are not required. This step can be done independently of the previous one, i.e., the input could be a primitive coarse mesh or just a coarse mesh as coming from the first step.

Each coarse element contains the information of the front by which was generated. Our algorithm splits each coarse element by planes parallel to the front face at a distance defined by the expression $hloc * factor^i \mid i = 0, ..., endline - 2$ from the front face. The first layer is then located at a distance of *hloc* from the front face, the second layer is located at a distance of *hloc* + *hloc*factor* from the front face, and so on. Note that the layer obtained with $i = endline - 1$ was already calculated in subsection 3.1 to build the coarse element.

Input: Coarse discretization of the geometry
for each element E **do**
  get front data
  for (i = 0; i < endline-1; i++) **do**
    for each F_j **do**
      compute the intersection points produced by $expansion_i$
    generate $face_i$ defined by the points of $expansion_i$
    generate $element_i$
Output:Discretization with the required layer density

Algorithm 2: Fulfilling layer density requirements

Figure 4 illustrates one refinement of a coarse element. Figure 4(a) shows the discretization of the truncated prism shown in Figure 2. The top coarse element is refined once by intersecting its edges with a plane located at the distance *hloc* from the front face as shown in Figure 4(b). The coarse element is divided into two new elements as shown in Figure 4(c) and (d).

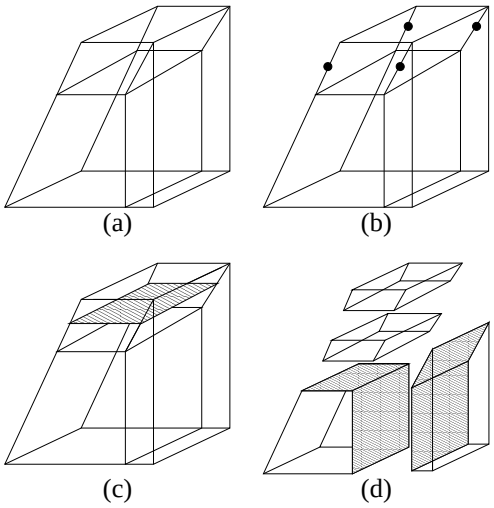

Figure 4: Refining a front element.

Let E be a coarse element, F be the front face of E, F_n be the face of E obtained by the maximum expansion of F, and F_j, the faces of E generated by joining F and F_n. The algorithm splits first all the F_j of the element E at the locations defined by $hloc * factor^i \mid i = 0, ..., endline - 2$, and then builds the new elements. The coarse element is divided in a number of new elements equal to the value of *endline* hence each F_i is also split in a number of faces equal to this quantity. The pseudo-code of the refinement process is shown in algorithm 2.

3.4 Fulfilling the required maximum edge length

As we mentioned in subsection 3.1 a front specification has several fields. Two of them are the maximum edge length (mel) constraint and the list of faces to expand. The *mel* field affects all the elements generated due to an expansion of the faces defined on this front.

We re-use the previous work (subsection 3.3) to accomplish the *mel* constraint on brick type of elements. The layering process of a brick generates only new bricks hence the same algorithm can be applied in other directions until the *mel* field is satisfied. Figure 5 shows how this process is done.

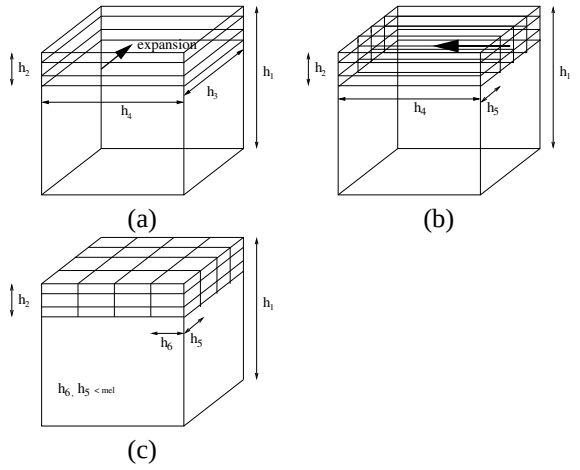

Figure 5: (a) Layer density constraint satisfied (b) mel satisfied in one direction (b) mel satisfied in the other direction

The same strategy cannot be applied over other types of element, at least not in every direction. It is necessary to specify the way to split the edges and the way to form the new elements for each primitive type. This task is part of the ongoing work.

3.5 Generating a cavity mesh

When all the expansions are done, there might remain a portion of the initial body unmeshed like the example shown in figure 6(b). When this happens it means that the user do not need a specific element density in that zone. Then the final step is to build a conforming mesh.

We have a library capable of building a tetrahedralization of a body. When all the steps are done, the remaining parts of the original body are tetrahedralized and the overall process is finished. Figure 6(f) shows an example of it.

A better strategy is to implement a mix element cavity generator, to produce a real mixed mesh. We have already a mesh generator to accomplish this task [4], however it must be adapted to generate a cavity mesh.

Example 1 (Figure 6) consists of: (a) the initial body, (b) the generation of the coarse mesh in relation to the given front input, (c) satisfying layer density, (d) satisfying the *mel* attribute in one direction (only for brick type of elements), (e) satisfying the *mel* attribute in the other direction and (f) the final mesh including the tetrahedralization of the cavity.

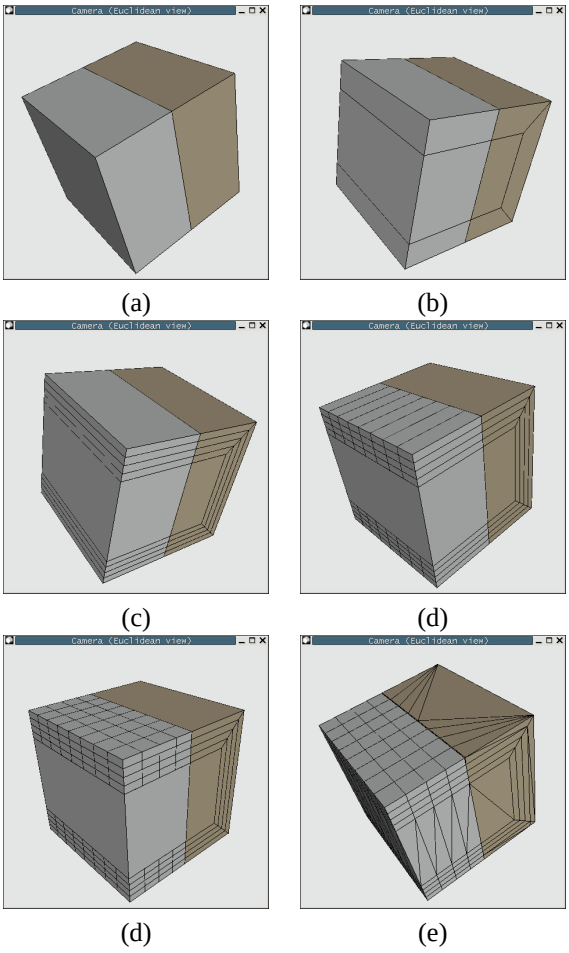

Figure 6: Example 1: mesh generation process for a simple geometry

4 COMPARISON BETWEEN NOFFSET STRATEGIES

There is an implementation of a NOffset strategy with only the tetrahedral type of elements specified in [5]. The differences between that work and ours are: (a) we use mixed elements, (b) we apply NOffset only one time and not every time a new layer is needed and (c) we refine the coarse elements in order to fulfill the layer density.

The more important difference between both implementations is produced in the layering step. The next table shows the number of operations to calculate each new point by each strategy in relation to the number of layers *n* required by the user.

current implementation	$15+6*n$
old implementation	$15*n$

The most important result is that we obtain a 60% reduction in the number of operations in relation to the old strategy. This is because the refinement process is much easier than to apply NOffset at each time a new layer is needed.

Another result that we should obtain in the future is that the final mesh should need less elements to accomplish the same required density. This is because we use mixed elements. The worst case is to mesh a body with just tetrahedra so the final number of elements would be the same in both strategies.

5 EXAMPLE AND ONGOING WORK

The second example is a body used for real semiconductor devices analysis called a bipolar transistor. Figure 7 shows the entire process: (a) the original body, (b) the coarse mesh, (c) accomplishment of layer density constraint, (d) generation of a cavity tetrahedralization and (e) and (f) the final mesh.

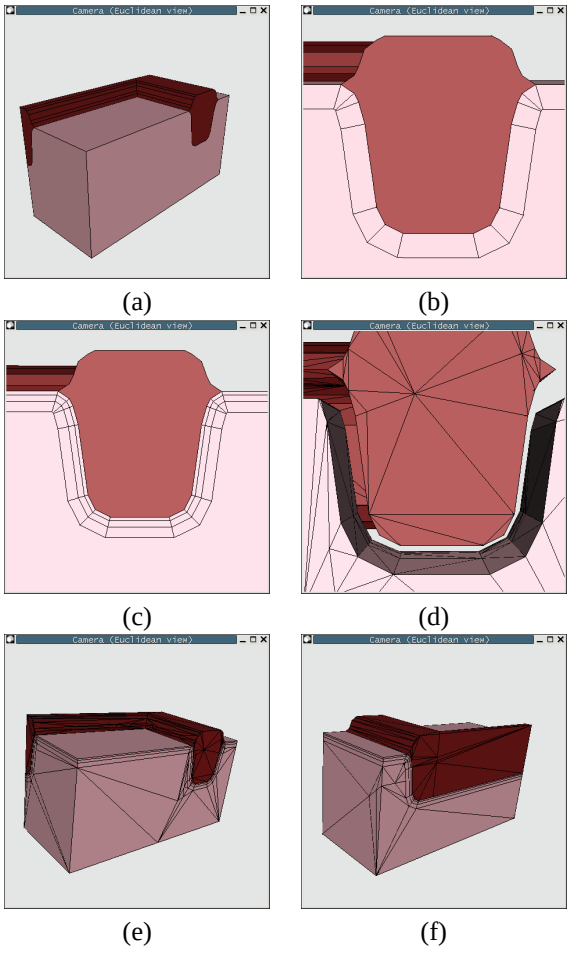

Figure 7: Example 2: mesh generation for a bipolar transistor

Currently, we are working on: (1) the generation of a non-conforming tessellation composed of coarse co-spherical elements such as cuboids, some kinds of prism and pyramid, and tetrahedra inside the coarse anisotropic elements before the layer density is generated, (2) the generation of a final mixed element mesh, (3) making the mesh Delaunay and (4) improving the quality of the elements inside the cavity.

6 ACKNOWLEDGMENTS

This work has been supported by Fondecyt Project N^o 1030672.

REFERENCES

[1] Pascal J. Frey, Houman Borouchaki, and Paul L. George. Delaunay tetrahedralization using an advancing front approach. In *5th International Meshing Roundtable*, pages 31–46, 1996.

[2] N. Hitschfeld, P. Conti, and W. Fichtner. Mixed Elements Trees: A Generalization of Modified Octrees for the Generation of Meshes for the Simulation of Complex 3-D Semiconductor Devices. *IEEE Trans. on CAD/ICAS*, 12:1714–1725, November 1993.

[3] N. Hitschfeld and R. Farías. 1-irregular element tessellation in mixed element meshes for the control volume discretization method. In *Proceedings of the 5th International Meshing Roundtable*, pages 195–204. Pittsburgh, Pennsylvania, U.S.A., 1996.

[4] N. Hitschfeld-Kahler. Generation of 3d mixed element meshes using a flexible refinement approach. *Engineering with Computers*, November 2004. Accepted for publication.

[5] Jens Krause. *On boundary conforming anisotropic Delaunay meshes*. PhD thesis, ETH Zürich. Series in Microelectronics, Vol. 115, 2001.

[6] Mark S. Shephard and Marcel K. Georges. Automatic Three Dimesional Generation by the Finite Octree Technique. In *International Journal for Numerical Methods in Engineering*, volume 32, pages 709–749, 1991.

[7] L. Villablanca. *Mesh Generation Algorithms for Three-Dimensional Semiconductor Process Simulation*. PhD thesis, ETH Zürich. Series in Microelectronics, Vol. 97, 2000. Hartung-Gorre Verlag, Konstanz, Germany.

[8] M.A. Yerry and M.S. Shephard. Automatic Three-dimensional Mesh Generation by the Modified-Octree Technique. *International Journal of Numerical Methods in Engineering*, 20:1965–1990, 1984.