D I S S E R T A T I O N

Mesh Generation for Technology CAD in Three Dimensions

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik
von

Peter FLEISCHMANN

[redacted]

Wien, im Dezember 1999

Abstract

The partitioning of a spatial model into a very large number of small elements is required for solving partial differential equations with the finite element or finite volume method. The generation of such a tessellation is called mesh generation. Solving the partial differential equations allows the simulation of the underlying physical behavior (as for example electrical current and potential).

During the past decade the demand for a robust and rigorous mesh generator for more complex structures has risen in the semiconductor industry. While a three-dimensional simulation was not possible in the beginnings, today's computers have made such a simulation feasible and desirable to be able to account for structure-dependent physical effects. Investigating the state of the art in other fields where three-dimensional mesh generation was applied longer ago is especially important to avoid limitations of earlier approaches and to meet the demands of the semiconductor industry.

The main contribution of this thesis is the development of a mesh generator based on the Delaunay Triangulation. An algorithm was devised to allow for robust mesh generation under finite precision arithmetics without the use of floating point filters. An existing but fairly seldom used Delaunay Triangulation algorithm was extended to work for typical degenerate point sets such as cospherical and cocircular point sets. A sophisticated mechanism avoids the use of fixed epsilon regions which can always be either too small or too large and which would result in a non-robust implementation. An advancing front passes through the desired regions and generates the tetrahedralization. Undesired regions are never meshed, also not temporarily. It is known that structures exist which cannot be tetrahedralized without inserting further points. The generation of a mesh is guaranteed for cases when the advancing front forms such a structure by allowing tetrahedra with negative volume. They are elegantly removed together with other badly shaped elements in a post-processing step. A bucket octree to store the points has proven especially efficient for tetrahedralization with the chosen Delaunay algorithm.

A major part of the work was devoted to the refinement of a general surface triangulation to be integrated into a conforming Delaunay Triangulation. Several techniques such as edge bisection or the orthogonal projection of a triangle vertex onto the opposite triangle edge were tested. Since none of these techniques proved robust enough for the given purpose, a more elaborate solution was devised. The surface triangulation is at first examined for structural edges. Afterwards, local transformations of triangles are combined with a refinement of structural edges. A specially developed system to derive a suitable refinement point (not necessarily by bisection) guarantees the termination of the refinement loop after a minimal number of point insertions. Finally, a triangle-based refinement follows the edge-based refinement. The Voronoi edge which is the dual of a Delaunay triangle is used to derive a

refinement point. This allows an additional improvement of the quality of the surface triangles with respect to their aspect ratio and angle. The efficiency of the developed mesh generator is documented with a set of examples.

Kurzfassung

Die Zerteilung eines räumlichen Modelles in eine sehr große Zahl kleiner Elemente wird für die numerische Lösung von partiellen Differentialgleichungen mittels der Finiten-Elemente- oder der Finiten-Volumen-Methode benötigt. Die Herstellung solch einer Zerteilung wird Gittergeneration genannt. Das Lösen der partiellen Differentialgleichungen ermöglicht die Simulation des zugrunde liegenden physikalischen Verhaltens (zum Beispiel Stromfluss und elektrisches Potential).

In der Halbleiterindustrie stieg während des letzten Jahrzehntes der Bedarf nach einem robusten und rigorosen Gittergenerator für komplexere räumliche Strukturen. War eine dreidimensionale Simulation in den Anfängen noch undenkbar, so wird sie mit der Rechenleistung heutiger Computer möglich und wünschenswert, um strukturbedingte physikalische Effekte erfassen zu können. Um die Restriktionen anfänglicher Gittergenerations-Methoden zu vermeiden und den Anforderungen in der Halbleiterindustrie gerecht zu werden, ist es besonders wichtig den aktuellen Stand der Forschung anderer Gebiete, in welchen dreidimensionale Gittererzeugung schon länger angewandt wird, miteinzubeziehen.

Der Hauptbeitrag dieser Dissertation ist die Entwicklung eines Gittergenerators basierend auf der Theorie der Delaunay-Triangulation. Ein effizienter Algorithmus wurde entworfen, welcher robuste Gittergeneration unter endlicher Rechengenauigkeit und ohne Verwendung von numerischen Fließkommafiltern ermöglicht. Hierzu wurde ein bestehender aber vergleichsweise selten genutzter Delaunay-Triangulations-Algorithmus um die Behandlung von degenerierten Punktmengen, wie etwa kosphärischer und kozirkularer Punktmengen, erweitert. Ein ausgeklügelter Mechanismus vermeidet die Benutzung von fixen Epsilon-Umgebungen, die immer zu klein oder zu groß sein können und in einer nicht robusten Implementation resultieren würden. Eine fortschreitende Front erfasst die gewünschten Gebiete und erzeugt die Tetrahedrisierung. Unerwünschte Regionen werden auch nicht temporär vergittert. Es ist bekannt, dass es Strukturen gibt die nicht tetrahedrisierbar sind, ohne weitere Punkte einzufügen. Um die Herstellung eines Gitters garantieren zu können, auch wenn die fortschreitende Front so eine Struktur formt, werden Tetrahedra mit negativem Volumen erlaubt. Diese werden elegant und gemeinsam mit anderen nachteiligen Elementen in einem Nachbearbeitungsschritt eliminiert. Ein *Acht-Baum/octree* zur Speicherung der Punkte hat sich in Kombination mit dem gewählten Delaunay-Algorithmus als besonders effizient für die Tetrahedrisierung erwiesen.

Ein bedeutender Teil der Arbeit ist der Verfeinerung einer allgemeinen Oberflächen-Triangulation gewidmet, um sie in eine konforme Delaunay-Triangulation zu integrieren. Hierzu wurde mit diversen Methoden experimentiert, wie zum Beispiel der Kanten-Halbierung oder der orthogonalen Projektion eines Dreiecks-Punktes auf die gegenüberliegende Dreiecks-Kante. Nachdem sich keine dieser Techniken als wirklich robust erwiesen hat, wurde eine

umfassendere Lösung entwickelt. Die Oberflächen-Triangulation wird zunächst auf strukturelle Kanten untersucht. Im weiteren wird die lokale Transformation von Dreiecken mit der Verfeinerung von strukturellen Kanten kombiniert. Ein speziell entwickeltes System zur Ableitung eines günstigen Verfeinerungs-Punktes (nicht unbedingt ein Halbierungs-Punkt) garantiert die Terminierung der Iterationsschleife nach einer minimalen Anzahl von Verfeinerungen. Auf diese Kanten-basierte Zerteilung folgt letztendlich eine Dreiecks-basierte Zerteilung. Dabei wird die zu einem Delaunay-Dreieck duale Voronoi-Kante zur Ableitung eines Verfeinerungs-Punktes herangezogen. Dies erlaubt eine zusätzliche Verbesserung der Qualität der Oberflächen-Dreiecke im Bezug auf Seitenverhältnis und Winkel. Die Effizienz des entwickelten Gittergenerators wird anhand einer Sammlung von Beispielen dokumentiert.

Acknowledgement

Foremost I would like to express my gratitude to Prof. Siegfried Selberherr who is responsible for the wonderful working environment and the perfect infrastructure at the Institute for Microelectronics. He always visualized important longterm aims and never got lost or distracted by bureaucratic details. In such a way he was always open to suggestions and very flexible in dealing with the specific needs of each student. This for each student different balance between freedom and pressure helped greatly to keep the motivation high and to guarantee the best possible outcome for each person and each project. I am also grateful to Prof. Dietmar Dietrich who was willing to serve on my examining committee on a very short notice.

Furthermore, I would like to thank all members of the institute for contributing to such an inspiring and pleasant atmosphere. I will never forget the good times I spent with my colleagues Rainer Sabelka, Heinrich Kirchauer, Rudi Strasser, Christian Harlander, Andi Hössinger, and Klaus Dragosits during the many interesting lunch hours. Rainer was always a great help and our time as system administrators was a lot of fun due to his many creative puns and funny jokes. Heinrich taught me how to really play badminton and Klaus always found intriguing stories on the verge of truthfulness. Rudi was always ready to entertain us all with a joke, a game, or a song. I also thank my first roommate Ernst Leitner who taught me most details about HP-UX and other things. With my other roommate Tibor Grasser I enjoyed many fruitful discussions during our special lunches at the "Wok" about all things in life. With Goran Kaiblinger I had a great colleague during night times. It was always interesting to see who would stay awake longer through the nights at the institute. Gerhard Schrom was the source of many original ideas and he served as an excellent partner for great discussions. Christian Troger was a colleague who was never tired of helping others. Regarding my work on mesh generation I would like to particularly mention Ernst Leitner, Wolfgang Pyka, Bernhard Haindl, and Robert Kosik who contributed a great deal to this thesis.

∞

Contents

Chapter 1

Introduction

MESH generation is the partitioning of a domain into a large number of simple pieces called elements. The domain is usually defined through the boundary of a geometrical structure or given by a Computer Aided Design (CAD) model. The elements must be well connected, neither gaps nor overlapping elements are permitted. A numerical analysis or computer simulation involves the formulation of i.e. non-linear coupled partial differential equations which describe the physical problem on a physical domain, in terms of algebraic equations on the discrete domain. The large system of linear algebraic equations which is assembled after discretization can be solved using direct or iterative solvers.

Among the various approaches [154, 156, 193] the *finite element method* and the *finite volume method* or *control volume method* are the most established discretization schemes. All methods imply specific qualities of the generated mesh to ensure convergence of the iterative solution process and to yield correct simulation results. With the growing importance of three-dimensional simulation mesh generation has become a critical factor. Surprisingly the generation of a suitable mesh often becomes the bottle neck and poses more difficulties than the subsequent simulation. The amount of data in three dimensions requires efficient and more sophisticated algorithms and data structures. Meshing algorithms which have worked well for two-dimensional problems are often not feasible for higher dimensions. Manual partitioning techniques are not desirable and cannot be efficiently applied to model descriptions which contain several thousand or more vertices. At the same time automatic meshing schemes are challenged by the increasing topographical complexity of the CAD model.

This situation is also experienced in semiconductor device and process simulation. The complexity of a modern semiconductor device often makes a three-dimensional analysis necessary to capture important physical effects which would not be exhibited by idealization to fewer dimensions. With the upcoming of three-dimensional device and process simulation the importance of three-dimensional mesh generation for Technology CAD (TCAD) has significantly increased. The stiff and highly non-linear equations governing the behavior of a semiconductor device [156] and the moving boundary situation during oxidation in process simulation require a powerful and efficient meshing tool. The need for a better fitted mesh with regard to certain quality or error measures necessitates fast global and local mesh adaptation techniques. Local remeshing also becomes important to repair mesh deformations resulting from moving boundaries and interfaces of the semiconductor device.

1.1 Outline of the Thesis

After a discussion of some of the problem issues and the state of the art in three-dimensional mesh generation and the demands in the fields of CAD and TCAD in Chapter 2, a brief overview of the concept and the means of mesh generation is given in Chapter 3. Such a "theory" of mesh generation includes topics as mesh quality, the discretization scheme, the generation of a surface mesh, and mesh adaptation. The notion of control space which defines a control function to guide a mesh generator in making more problem-dependent than geometry-dependent meshes is introduced.

Chapter 4 provides a concise summary of common meshing techniques. Most existing meshing methods can be distinguished in such a manner. Hybrid methods combine some of the given techniques. The text further concentrates on Delaunay methods. These are among the unstructured approaches the one with the strongest mathematical background. All aspects related to the theory of Delaunay are discussed in Chapter 5. This includes the definition of the Delaunay properties, some combinatorial facts, an overview of Delaunay algorithms, and the concept of Steiner Triangulations. The basic characteristics of a Delaunay Triangulation which is defined for a given point set are the duality to the Voronoi graph and the empty circumsphere property. If the reader is not at all familiar with that, he is advised to browse through Chapter 5 beforehand. Previous chapters which include the review of research and methodologies refer to Delaunay techniques frequently.

The design and architecture of the developed mesh generator are presented in Chapter 6. Finally, several examples of CAD and TCAD structures are given in Chapter 7 and an Outlook in Chapter 8.

1.2 Terminology

Most technical names are self-explanatory and care has been taken to use well defined and precise expressions. In order to avoid any confusion a few terms require further comments and are given in Table 1.1.

location	a space defined by a set of coordinates
vertex	a zero-dimensional part of a higher dimensional entity, e.g. part of a geometry, a polygon, or polyhedron
n-simplex	(n+1) affinely independent vertices
point	a zero-simplex
edge	a one-simplex
triangle	a two-simplex
tetrahedron	a three-simplex
node	not a point, but part of a tree data structure
facet	a two-dimensional entity, e.g. a polygon
face	the verb as in "facing a direction"
non-planar	curved, but mostly used as short form for piecewise-planar
topology	the connectivity of a set of points where adjacency is given by the edges regardless of the location of the points
boundary	(n-1)-dimensional topology in R^n
grid	n-dimensional structured topology in R^n, numerical grid
mesh	n-dimensional unstructured topology in R^n
surface mesh	(n-1)-dimensional unstructured topology in R^n
triangulation	a simplicial complex [118] where the highest order element is a 2-simplex in R^n
tetrahedralization	a simplicial complex where the highest order element is a 3-simplex in R^n
Triangulation	special triangulation or tetrahedralization as for example in "Delaunay Triangulation" or "Steiner Triangulation", independent of the order of the simplices
boundary consistent	incorporating the boundary, as opposed to intersecting or overlapping
boundary-fitted	nicely aligned with the boundary and not just boundary consistent
boundary conforming	This term is avoided because it is sometimes used in different ways throughout literature, e.g. as boundary consistent or as boundary-fitted.

Table 1.1: A short glossary of important terms.

Chapter 2

Challenge and Demands

ACTIVITIES in three-dimensional mesh generation are examined with special respect to semiconductor process and device simulation. Some of the challenges encountered when embedding a mesh generator into a CAD-Analysis system are explored. The demands on meshing differ between TCAD and computational mechanics and fluid dynamics applications.

2.1 CAD

The conversion of the input structure, e.g. a CAD model, to obtain a meshable structure description can be a non-trivial process. It calls for a transformation of the input data to provide the mesher with more and/or different geometrical information. It will be easier to apply a solid model oriented mesher to a solid input model rather than to a boundary representation model (*BREP* model). On the other hand a mesh generator may require a consistent triangular boundary description which can be extracted from a *BREP* model in a straightforward manner. A solid model may not contain the necessary information on the boundary without calculating the intersection of the solids. The type and availability of geometrical information on the input model becomes a key issue to automate the meshing process. In the early beginnings of finite element analysis engineers had to create meshes manually exploiting their knowledge and understanding of the model's geometry [194]. Since then, more general algorithms have been introduced and activities have been geared more and more towards automation [19, 25, 56, 105, 151, 169, 187, 191]. Often commercial CAD systems offer a large number of different algorithms which the user must utilize by finding a good selection, a good sequence, and good parameters to create and adjust a mesh interactively. Today most mesh generators claim to work automatically and to "understand" the geometry without human interaction. To some extent human interaction means additional geometrical information. A fully automatic mesh generator which requires very detailed input data may be less sophisticated than one that requires only a minimum of information. Two extremes can be distinguished, which both have their application:

- Exploiting as much geometrical information as possible (parts, layers ...). Possibly utilizing the same data structures as for example the CAD editor with which the model was created.

- Accepting a minimum of information like e.g. a flat data hierarchy consisting of a single list of polygons only.

Impressive looking meshes of, for example, an engine with many complicated mechanical parts, cylinders, and valves, are often easily generated because the information on each cylinder and part itself is provided. Structured grids are built for all simple parts and merged to form the entire mesh. If a single, combined surface of all parts is the only input data, the meshing process becomes unproportionally more complicated. This case has to be dealt with when the CAD model is not manually constructed using an editor, but rather derived from a simulation where the topography itself has been subject to the analysis. Such topography simulation is a common task in semiconductor process simulation.

On the other hand human interaction may provide engineering judgement. Not all geometrical tasks interfacing the design and the analysis phase are readily to be automated. A better understanding of the important balance between automation and control may save considerable computation time. Is it really necessary to mesh an entire model with all features? Or could some of the data be omitted to simplify the meshing process while still providing the desired simulation results? The input model can contain a wide variety of inconsistencies harmless for visualization purposes but crucial for any meshing algorithm. The order of the vertices in a polygon definition might not be correct or gaps in the boundary and overlapping elements might exist. In such cases one depends on powerful interactive tools to repair the problem areas [23]. A technique which is application dependent and requires the judgement of the engineer is the dimensional reduction [38] of the input model by means of the medial axis [3, 169]. The medial axis or medial object is a method to detect and "understand" features of the geometry (Fig. 2.1). Utilizing the knowledge of these features one can perform various preprocessing steps like decomposing a complex structure into several simple ones or removing too much detail in a model (de-featuring). Such a qualitative data reduction results in a simplified model which may suffice for e.g. a stress analysis in computational mechanics. Unfortunately, the computation of the medials is very costly, because it requires the Delaunay Triangulation of a highly refined boundary.

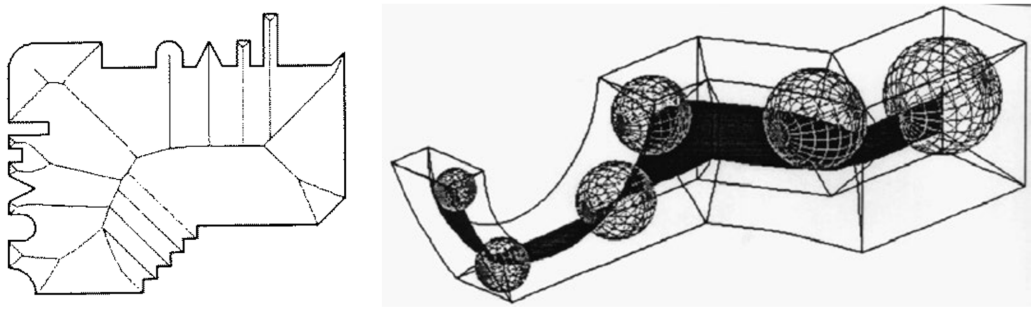


Figure 2.1: Medial axis and medial object, M. Price et al. [124].

A definition of the common term *local feature size* will prove useful to theoretically formulate and grasp complex geometrical constellations.

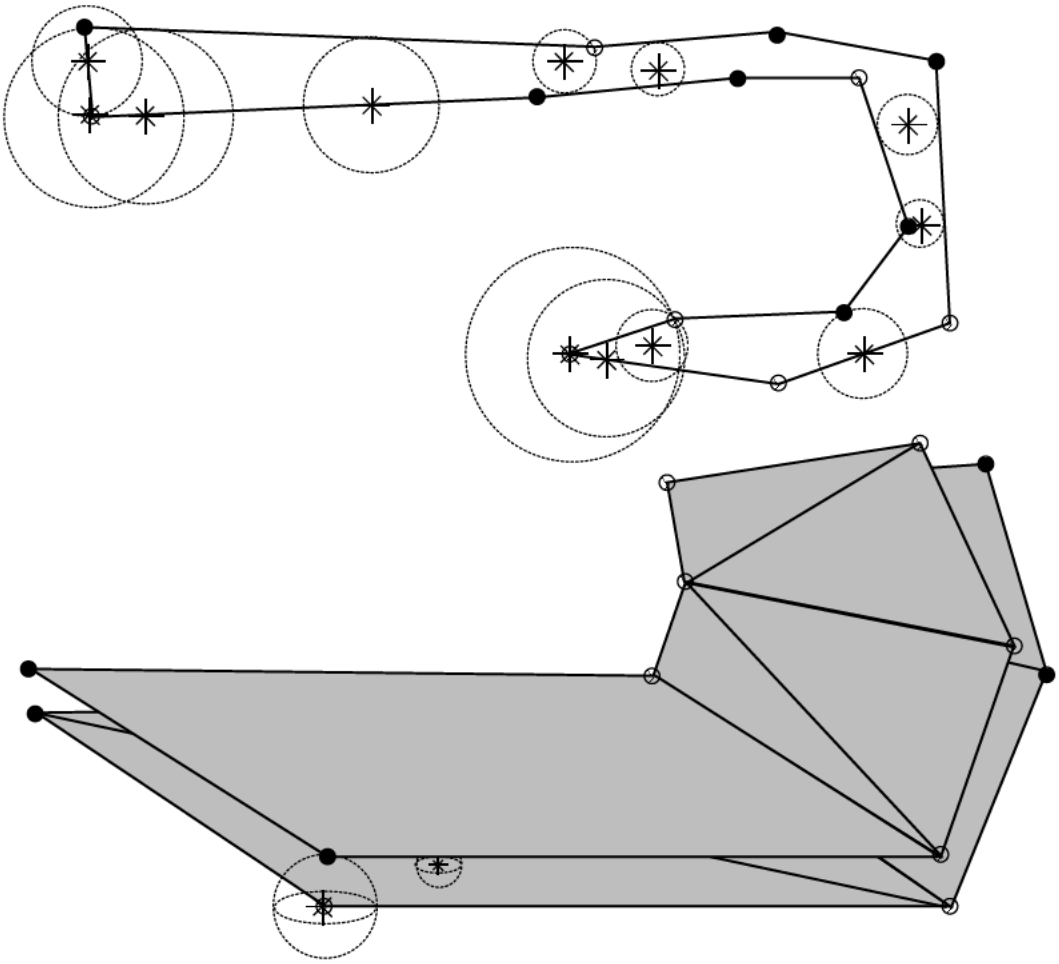


Figure 2.2: Thin layers in two and three dimensions with the local feature size (radius of the circles/spheres) at example locations (stars).

Definition 2.1 (local feature size) *For any given location x the radius of the smallest ball (volume defined by a sphere) with center x that intersects two non-incident points, edges, or facets is called the local feature size $\mathcal{L}(x)$. In two dimensions the smallest ball is a smallest disk (area defined by a circle). Note that two different points can never be incident.*

The continous function $\mathcal{L}(x)$ which is defined on the entire domain satisfies the inequality $|\mathcal{L}(x_1) - \mathcal{L}(x_2)| \leq \|x_1\vec{x}_2\|$ as proven in [135]. Intuitively $\mathcal{L}(x)$ should reflect small features of the input geometry but it should not reach arbitrarily small values for sensible inputs. In three dimensions it can therefore be advisable to slightly adjust the definition by combining incidence with visibility as described in detail in [162].

2.2 Semiconductor Process and Device Simulation

The demands can be roughly categorized.

- automatic mesh generation of complex semiconductor devices within an integrated framework of simulators
- anisotropic geometrical requirements (structure dependent)
- mesh density grading, anisotropic density requirements (application dependent)
- finite element mesh quality and closed control volumes for finite volume methods

Combined full scale three-dimensional process and device simulation has suffered from the lack of available TCAD tools for the various established data formats in the field. A framework of such a TCAD environment providing the necessary tools is itself still a subject to research and development [79, 172, 182]. Aside from preliminary means to couple and control some of the simulators [121, 173, 185], powerful geometry processors are missing. Considerable person power is consumed to detect inconsistencies in a tedious manner, and to debug structures that the mesher should, but does not mesh. Such structures typical for semiconductor simulation purposes often exhibit extreme ratios between the size of the smallest and the largest features, where no structural simplifications can be afforded. Only quantitative data reduction may be performed to decrease the amount of redundant data.

Once the mesher is provided with a valid and clean model it has to deal with the topographical as well as topological complexity of the underlying geometry which contains many internal surfaces, edges sharing more than two facets, and thin layers.

Thin Layer Two polygonal surfaces with a large area compared to the maximal local feature size $\mathcal{L}_{\max}$ measured inbetween. Less general subtypes are formed when the surfaces are planes. Furthermore, the planes might be parallel or have normals in the direction of a coordinate axis. For the simplest case of two parallel planes their normal distance $d = \alpha\mathcal{L}, \alpha \in [1..2]$ is small compared to their lateral expansion (Fig. 2.2).

It is relevant especially in three dimensions to detect such precisely defined thin layers in order to avoid an unnecessary high number of mesh elements. While standard isotropic elements are still manageable in two dimensions, the number of isotropic elements in three dimensions roughly increases quadratically with the ratio of lateral edge length to $\mathcal{L}_{\max}$. It will be necessary independent of the physical application or solution quantity and due to a pure geometrical feasibility that the mesh generator enforces anisotropic elements at automatically detected regions which convey anisotropic geometrical information. The most general and sophisticated algorithms have to be applied in order for the mesh generator to deal with automatically generated structures. Computer generated topographies not only evolve from etching and depositon modules, but also through three-dimensional extraction of geometrical data from two-dimensional simulations and through solid modelers deriving their information from layout data. An automatic geometry preprocessor and mesh generator can complete the integrated framework system and enable the fast and efficient optimization of various design and manufacturing parameters.

The range of the simulated magnitude (e.g. the concentration during diffusion simulation or in device simulation) covers several exponents and leads to difficulties when discretized with conventional meshes. The variation of the local mesh density over space and over direction (anisotropic mesh density) is the key factor to keep the size of the mesh manageable and at the same time the discrete distribution of the magnitude accurate. A balance between the number, the size, and the quality of the elements must be achieved for such a good mesh grading. The number should be minimized while the size should satisfy local density criteria. Often, the mesh is not optimally fitted and too coarse or too fine elements exist in various areas of the simulation domain. Alternatively, the desired accuracy of the analysis cannot be achieved in all areas, or the subsequent tools and simulators are pushed to limits beyond the scope of an average computer by a generally too fine mesh. An increased flexibility in refinement allowing for rapid changes of the mesh density while keeping the overall element count low would be ideal. This flexibility cannot be increased infinitely. The rapid change of element size is limited under the premise of maintaining a certain element quality.

The finite element method requires that the elements possess a certain geometrical quality. It can have a very bad influence on the convergence of the solution if the elements have extremely obtuse angles. The angle spanned by two planes (dihedral angle) becomes an important measure in three dimensions (Chapter 3). Achieving good bounds on the dihedral angle of the elements becomes a major demand in three-dimensional finite element mesh generation.

The finite volume or control volume method which is often also called the box integration method is crucial for semiconductor device simulation, because it can be combined with the Scharfetter-Gummel scheme [58][119][146] which takes the exponential carrier concentration into account. As will be discussed in Chapter 3 "closed" control volumes are required which is usually accomplished by constructing a *Voronoi* [117] type mesh. The *Voronoi box* associated with each point from the mesh satisfies the requirements and possesses some advantages as opposed to e.g. boxes which are defined by centroids (gravity boxes). The common approach to obtain such a Voronoi tessellation is to construct its dual Delaunay Triangulation. (See Chapter 5 for a description of the Delaunay theory and its definitions.) Stable Delaunay meshing of a complex semiconductor structure is a demanding effort in three dimensions. The generalization of a two-dimensional to a three-dimensional Delaunay Triangulation poses not a mere quantitative but rather a qualitativ challenge (Chapter 5). Among other reasons is that a bounded dihedral angle between facets of the input structure becomes a crucial factor for a provably terminating Delaunay algorithm. With an increasing number n of points in three dimensions, keeping the computational complexity below $O(n^2)$ is another crucial requirement for success.

The tasks can be summarized in relation to their application.

- Meshing of comparatively simple (near 90° dihedral angles between input facets) but huge structures composed of a large number of vias and lines. *Simulation of 3D Interconnects.*

- Handling the minimum and possibly faulty information on arbitrary complex structures provided by topography simulation with eventually moving structure boundaries [45]. *Semiconductor process simulation.*

- Resolving highly non-linear quantities with a directional mesh density which is not only limited to the three directions of the cartesian axises (physical anisotropy). *Semiconductor device simulation.*

- Dealing with strongly acute dihedral angles between input facets and extreme ratios between local edge lengths and local feature size (geometrical anisotropy). *Semiconductor process and device simulation.*

2.3 State of the Art

Most of the research in mesh generation is done in the fields of computational fluid dynamics and stress mechanics. Three-dimensional mesh generation for applications in TCAD and for semiconductor process and device simulation is a recent topic and leaves room for much innovation. The following is a summary of modern approaches respectively in the order of cartesian-based, octree-based, unstructured tetrahedral, advancing front, hybrid, and Delaunay methods.

Aftosmis et al. [2] developed cartesian meshes for applications in computational fluid dynamics. They incorporate complex boundaries with a rigorous method to intersect the cartesian cells with the boundary representation. A limited flexibility in refinement is experienced, due to the cartesian nature of the mesh and the isotropic refinement technique. Each further refinement level yields about 2–4 times more elements. With intial 1.6 million elements a further increase in accuracy by adding one more refinement level would result in 3.3–6.6 million elements according to one of the examples in [2]. Still, in some areas the achieved accuracy is not optimal and refinement would be appropriate. Such a high point propagation due to refinement and the rigorous intersection method which greatly increases the number of surface elements is less likely to be afforded on a typical computer available to a user for everyday simulation purposes.

The cartesian-based mesh generator *OMEGA* well known for semiconductor simulation applications is described in [53]. This intersection-based method originates from previous works using bisection-based and octree decomposition techniques [35, 65, 66]. Improvements include the ability to tessellate the cartesian cells into Delaunay tetrahedra and a more anisotropic refinement approach. Subdivisions for higher refinement levels are performed by splitting the nodes at intersections or geometry-specific vertices instead of splitting the nodes always at bisections. However, if vertices of the geometry lie closely together without conveying important geometrical information as for instance in the case of staircase-like approximations of slopes, too thin elements are created and unnecessarily propagated throughout the mesh. The key issue with cartesian methods is to merge the coordinate system aligned cells with the unstructured surface of the geometry. It seems that without rigorously generating a cell-consistent surface triangulation as in [2], difficulties are experienced dealing with the boundaries and interfaces of a semiconductor device. To attain optimal flexibility regarding both the mesh density distribution as well as the fitting of the boundary it will be desireable to achieve anisotropy in arbitrary directions (not just along the three coordinate axises).

Johnson et al. [76] use an unstructured tetrahedral method for three-dimensional flow problems. The mesh generator is capable to efficiently deal with moving objects in fluids.

It is linked to an input modeler for non-uniform rational B-spline (*NURBS*) surface patches which however makes it less suitable for TCAD applications.

Advancing front methods [19, 94] are mostly used because of their good point placement properties which result in better boundary-fitted meshes as opposed to mere boundary consistent meshes. Elements are not intersected with the boundary, instead they are grown from the boundary and are therefore well aligned with it. An interesting technique to utilize the efficient and well defined Delaunay Triangulation as a background mesh[1] for advancing front style mesh generation has been developed by P.L. George et al. at *INRIA*, France [50]. Generally, it can be of interest to construct *protection layers* around physically crucial boundaries or interfaces to provide the required orthogonal mesh resolution for a correct representation of a physical magnitude of the solution. It remains an open problem how to effectively apply such methods to the complexity of a three-dimensional semiconductor device. In two dimensions approaches to enforce a certain anisotropy along inversion layers or otherwise highly non-linear interfaces have been investigated by [4, 82, 112, 133, 177]. Most methods have come to a stage where they more or less rely on a triangulation engine, e.g. *TRIANGLE* [160], to generate a valid tessellation of a set of supporting grid lines and/or mesh points supplied together with the boundary. Conformal mapping techniques to ensure orthogonal boundary-fitted meshes by means of mathematical transformations of parametrized non-planar boundaries pose computation difficulties in three dimensions. The related partial differential equation (PDE) gridding method[2] seems better extendable for three-dimensional TCAD applications. So far the two-dimensional implementation *CGG* [26] of the PDE method has proven powerful for semiconductor device simulation.

Pure octree-based solutions combined with tetrahedral templates are hardly used for todays TCAD applications. Considering the variety of existing approaches in three dimensions one can observe that recently most methods have evolved to become hybrid methods and require some sort of general tetrahedralization [77, 81, 90, 178]. The software package *LAGRIT* formerly known as *X3D* provides a universal mesh generation toolkit [102]. Another hybrid mesh generator is *MESHise* [54] which is a further development of *OMEGA*.

For moving boundary situations level set methods have gained great importance [1, 27, 90, 127]. They require a mesh to define the magnitude which describes the moving boundary. At some stage a boundary consistent mesh must be derived from the level set representation by means of intersection and eventually tetrahedralization.

The history of methodologies in two-dimensional process and device simulation leads to the observation that an unrestricted and stable triangulation engine is one of the most important tools for mesh generation purposes. The theory of Delaunay provides a mathematical foundation for provably terminating triangulation algorithms [15]. While the two-dimensional Delaunay Triangulation poses less difficulties and a vast amount of literature exists [41, 70, 86, 145, 175], the integration of boundaries into a three-dimensional Delaunay Triangulation remains a very active research area. An extremely thin oxide layer on top of a comparatively big silicon block calls for very sophisticated algorithms to automatically generate an unstructured Delaunay mesh consistent with the boundaries and interfaces. This is one of the reasons why most of the software for a three-dimensional Delaunay Triangula-

[1] The term "background mesh" will be explained in Section 3.3.
[2] See Section 4.1.2.

tion available to the public domain is less suitable for semiconductor simulation purposes. A Delaunay method using local transformations is included in *GEOMPACK* by B. Joe [72, 74]. *QHULL* is a code which constructs a Delaunay Triangulation by computing the convex hull in higher dimensions [12]. Excellent work related to Delaunay refinement mesh generation can be found in [164]. The *QMG* software package for quality mesh generation based on octrees is described in [109]. A detailed and much more complete survey of the worldwide research activities in mesh generation is maintained by R. Schneiders [147]. Regarding future aspects of TCAD and automation issues one may refer to [34, 83].

Chapter 3

Mesh Generation

3.1 Geometrical Mesh Quality

Well shaped elements are especially important for finite element methods. Certain geometrical quality measures and ratios can be defined to evaluate the shape of an element. Such purely geometrical criteria lead by nature to an isotropic mesh density distribution. They are implemented without the knowledge of the physical problem at hand. Various types of not so well shaped elements are depicted in Fig. 3.1.

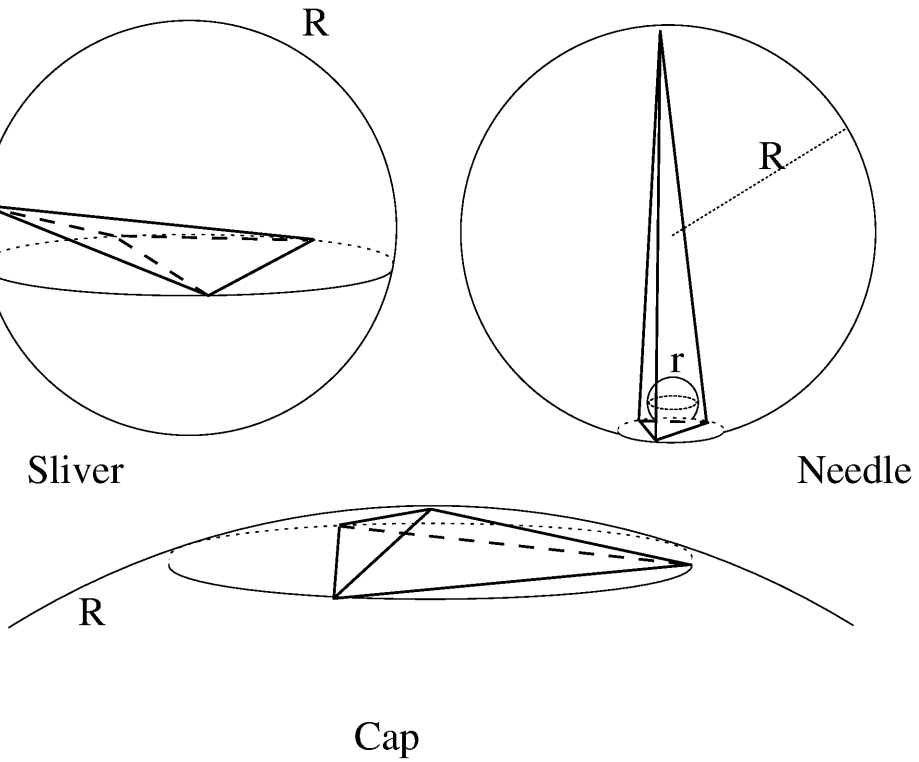


Figure 3.1: Various types of not so well shaped elements and some parameters.

Some simple parameters of triangles in two dimensions and tetrahedra in three dimensions are the edge lengths l_i, (dihedral) angles α_i, volume V, circumsphere radius R, insphere radius r, and normal distance d_i of a vertex from its opposite edge or triangle. The *height* of an element is defined as $d_{\min}$. It can be desirable to avoid elements with a too small height or too obtuse (dihedral) angles. While such large angles are related to the error of a finite element approximation, small angles can have a negative effect on the condition number of the stiffness matrix [5, 80]. The following are some examples for the definition of an element measure where greater values of Q denote a higher quality shape.

$$Q_1 = \frac{l_{\min}}{R} \tag{3.1}$$

$$Q_2 = \frac{r}{l_{\max}} \tag{3.2}$$

$$Q_3 = \frac{r}{R} \tag{3.3}$$

$$Q_4 = \frac{l_{\min}}{l_{\max}} \tag{3.4}$$

$$Q_5 = \frac{V}{l_{\max}^3} \tag{3.5}$$

To be more precise one has to note that not all of these definitions provide a well behaved quality measure in all dimensions. For example only $Q_{2,3,5}$ satisfy

$$\lim_{area,volume \to 0} Q = 0 \tag{3.6}$$

Q_3 is actually the inverted *aspect ratio* as it is commonly defined for three-dimensional elements. Q_5 has been used by [8, 87]. Q_1 behaves well in two dimensions, but it is inapt to capture the shape of three-dimensional sliver elements as depicted in Fig. 3.1. The volume of such a sliver element can be made arbitrarily small while at the same time Q_1 remains a positive constant. The dihedral angles of the sliver element can be changed to the better or to the worse while the measure Q_1 can be kept constant. One can gradually transform a well shaped element into a sliver element by moving one vertex without changing $l_{\min}, R$ and hence without changing Q_1. This interesting fact follows from the important relation between the angles of an element and its l_i, R parameters. Only in two dimensions it is possible to derive a formula for triangles which describes the relation between the edge length and its opposite angle (Fig. 3.2 and Fig. 3.3).

$$\sin \alpha_i = \frac{l_i}{2R} \tag{3.7}$$

Assuming that R is constant, the smallest angle will correspond to the shortest edge. Hence, a minimum bound for Q_1 also is a bound to the smallest angle in the triangle. Lacking this relationship in three dimensions many quality criteria fail to guarantee bounded dihedral angles. Q_4 has more the nature of a one-dimensional measure. It fails even in two dimensions to avoid badly shaped triangles. With a fixed Q_4 a triangle may have any obtuse angle. It will depend on the application whether or not angles are important and which criteria prove to be useful.

An important conclusion can be drawn. Whatever means are pursued to improve the element quality, the employed technique must fit to the applied measure. Otherwise termination

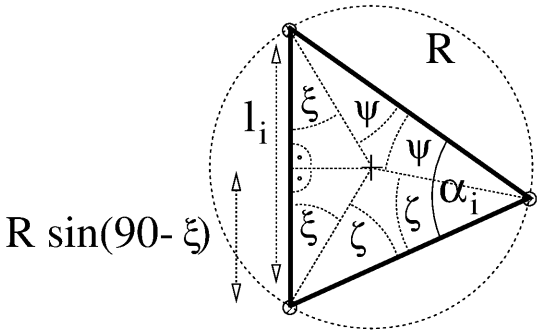


Figure 3.2: The relation between the edge length and its opposite angle in a triangle follows from $2\xi = 180 - 2\psi - 2\zeta = 180 - 2\alpha_i$ and therefore $l_i = 2R\sin(90 - \xi) = 2R\sin(\alpha_i)$.

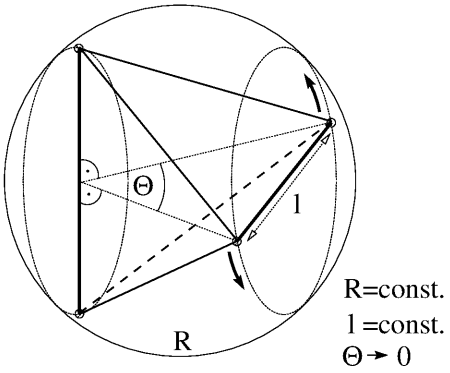


Figure 3.3: With constant edge length and circumsphere radius the opposite dihedral angle in a tetrahedron can have arbitrary values.

is not ensured. If the number of needles and caps should be reduced by means of refinement but the number of slivers by local transformations, the former must be distinguished from the latter. The refinement should then be controlled by e.g. Q_1 which will not detect the slivers. And the local transformations could be applied in a following step for elements which do not comply with e.g. Q_3.

3.2 Finite Volumes and Finite Elements

The two important discretization methods in TCAD are the box integration method (finite volume or control volume method) and the finite element method. Each method imposes certain requirements on the mesh. For the box integration method they result in conditions for the triangles of the surface mesh. For the finite element method geometrical criteria, as discussed in the previous section, suffice for most applications. However, for diffusion problems which are important in semiconductor process simulation further principles need to be investigated. A few simple examples will be presented to show the influence of different meshes on the validity of the discretization.

3.2.1 Requirements for Finite Volume Meshes

The box integration method was early introduced by Macneal [96]. The Scharfetter-Gummel scheme [58, 119, 146] which is crucial for semiconductor device simulation is applied in conjunction with the box integration method. The integration domain is partitioned into well defined boxes (control volumes) with positive cross-sections [52, 64, 158]. At present all major device simulators (*DESSIS* [91], *MEDICI* [181], *MINIMOS* [18, 44]) require that these control volumes coincide with the Voronoi regions [117] of the points[1]. An additional discretization error is avoided due to a specific advantage of the Voronoi box. A Voronoi facet (part of the boundary of a Voronoi box) orthogonally bisects the dual edge in the mesh by definition. (It

[1] See Chapter 5 for a description of the Voronoi diagram and the Delaunay properties.

intersects the edge perpendicular at the midpoint.) The suitability of gravity boxes defined by the centroids and edge midpoints as opposed to Voronoi boxes has been investigated for two-dimensional semiconductor device simulation in [165]. Further research will have to show whether or not such a modified discretization scheme based on gravity boxes can be applied successfully in three dimensions and if an additional discretization error is negligible.

Because of the duality between the Voronoi diagram and the Delaunay Triangulation, the mesh generator is required to perform a Delaunay partitioning of which the Voronoi boxes are easily extracted. At the boundary the control volumes must be closed [64]. Voronoi points, which are the circumcenters of the Delaunay tetrahedra, are not allowed to lie outside of the domain. A Voronoi point located outside defines a vertex of an incident Voronoi box which must intersect the boundary as a result. This "cut" box is associated to an interior point (not part of the boundary) and forms an open control volume (Fig. 3.4).

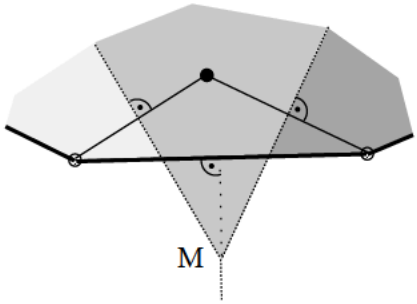


Figure 3.4: A Voronoi box which intersects the boundary and an outside Voronoi point M. The Voronoi regions for each point are shaded differently.

If the Voronoi boxes are indeed used as the control volumes, the requirements directly translate to the following criteria for the triangles of the surface mesh.

Criterion 3.1 (smallest sphere) *Let P be a finite set of points in n-dimensional space R^n and let t be an $(n-1)$-dimensional boundary simplex. t and its n linear independent points $p_{t,i}$ define an infinite number of n-dimensional spheres S_{t,r_c} where each sphere contains the points $p_{t,i}$ on its perimeter. All points p_k in P are associated with a Voronoi box V_k. The smallest sphere $S_{t,r_{\min}}$ contains no other points of P if the Voronoi box V_m does not intersect t for all m with $p_m \notin p_{t,i}$.*

In two dimensions the smallest sphere criterion is also sufficient to guarantee closed control volumes. In three dimensions an empty smallest sphere (equatorial sphere of a boundary triangle) might not guarantee a closed control volume. It can be shown that an additionally applied smallest sphere criterion to the edges of a boundary triangle is in conjunction stronger and suffices.

Criterion 3.2 (smallest sphere, edges and triangle) *Let P be a finite set of points in three-dimensional space R^3 and let t be a boundary triangle defined by three boundary edges $e_{t,i}$. Each $e_{t,i}$ with its two linear independent points $p_{e,j}$ defines a smallest three-dimensional*

sphere $S_{e,r_{\min}}$ where each sphere passes through the points $p_{e,j}$. t defines a fourth smallest sphere $S_{t,r_{\min}}$ (equatorial sphere) which passes through the points $p_{t,l}$ of the triangle. All points p_k in P are associated with a Voronoi box V_k. No Voronoi box V_m intersects t for all m with $p_m \notin p_{t,l}$ if and only if the four spheres $S_{e,r_{\min}}, S_{t,r_{\min}}$ contain no other points of P.

The proof essentially boils down to showing that the volume covered by the four smallest spheres (one triangle and three edges) "swallows" any sphere $S_{x,R}$ defined by a center x and radius R, where x is located anywhere on the triangle and R is such that the sphere $S_{x,R}$ does not contain any of the three vertices of the triangle.

As will be seen in Chapter 5 the smallest sphere criterion (Crit. 3.1) is stronger than the Delaunay criteria (Crit. 5.1 and Crit. 5.2). If no Voronoi box associated with an internal mesh point intersects the boundary, the simplices forming the boundary must be Delaunay simplices. In other words: If one chooses a Delaunay Triangulation to utilize its inherently given Voronoi boxes as control volumes, and if the discretization scheme relies on closed control volumes, the boundary elements must be adapted in one way or another to satisfy the stronger criteria independently of the application. It follows from the well known *Thales circle* that in two dimensions the angle opposite of a boundary edge must not be obtuse. Non-obtuse triangulations can be guaranteed in two dimensions [7]. For three dimensions such a guarantee remains an open problem. In practice the mesh generator is required to construct a Delaunay mesh and to ensure that the surface mesh elements are at least Delaunay.

3.2.2 Requirements for Finite Element Meshes

Such well pronounced requirements based on a different criterion can be formulated for a specific application of the finite element method. The basis is the *maximum principle* which is the most important property of solutions to convection-diffusion equations. In its simplest form it states that both the maximum and the minimum concentrations occur on the boundary or at the initial time. This implies that if the boundary and initial values are positive, the solution must be positive everywhere and concentrations may never reach negative values. It is desirable that the employed discretization also satisfies a maximum principle. As is well known, this is guaranteed, if the system matrix resulting from the discretization is an M-matrix[2] [69, 125].

The system matrix $\mathbf{K}$ for a simple diffusion with a standard Galerkin weighted residual approach, linear elements, and backward Euler time discretization has the following form

$$\mathbf{K} = \frac{1}{\Delta t}\mathbf{M} + D\mathbf{S} \tag{3.8}$$

where $\mathbf{M}$ denotes the mass matrix, $\mathbf{S}$ is the stiffness matrix, and D is the diffusion constant. $\mathbf{K}$ becomes an M-matrix if the mass matrix is lumped and $\mathbf{S}$ is an M-matrix. Since $\mathbf{S}$ only depends on the mesh this condition translates to a constraint on the mesh. The off-diagonal entries $s_{ij}, i \neq j$ of $\mathbf{S}$ must not be positive. These coefficients can be generally expressed as

$$s_{ij} = \sum_{elements} \int_e \nabla N_i \cdot \nabla N_j \, dA \tag{3.9}$$

[2] A real, nonsingular $n \times n$ matrix A where $a_{i,j} \leq 0 \;\; \forall \;\; i \neq j$ and $A^{-1} > 0$.

where N_i, N_j denote the basis functions and A is the area (volume) of element e. The in-product $(\nabla N_i \cdot \nabla N_j)$ has a simple geometrical meaning and leads to an angle criterion for each edge in the mesh, which was recently introduced by [189]. It is an important consideration in three-dimensional finite element mesh generation for diffusion applications with a high concentration gradient.

Criterion 3.3 (sum of dihedral angles) *Let $e_{i,j}$ be an edge with n adjacent tetrahedra t_k. For each t_k two planes exist which do not contain $e_{i,j}$ and which span a dihedral angle θ_k. The two planes share an edge with length l_k. The sum over $k = 1 \ldots n$ of the cotangens of θ_k weighted by l_k must be greater or equal than zero.*

$$\sum_{k=1}^{n} l_k \cot \theta_k \geq 0 \tag{3.10}$$

Figure 3.6 depicts an example where this criterion is violated for the interior edge $e_{i,j}$. Four adjacent tetrahedra exist of which two span a 90° angle. Hence, $\cot \theta_3 = 0$ and $\cot \theta_4 = 0$. As one can see from the figure $\cot \theta_1 = \cot \theta_2 = -\frac{1}{\sqrt{2}}$ (θ_1, θ_2 are obtuse, $\sim 125.3°$) and hence the total sum is negative.

In two dimensions (3.10) can be written as

$$\cot \theta_1 + \cot \theta_2 \geq 0 \tag{3.11}$$

where θ_1 and θ_2 are the angles of two triangles sharing a common edge $e_{i,j}$ as shown in Fig. 3.5. It can be assumed that

$$0 < \theta_{1,2} < 180° \tag{3.12}$$

and therefore

$$\sin \theta_1 \sin \theta_2 > 0 \tag{3.13}$$

Hence, multiplying (3.11) with $\sin \theta_1 \sin \theta_2$ results in

$$\sin \theta_2 \cos \theta_1 + \sin \theta_1 \cos \theta_2 \geq 0 \tag{3.14}$$

which is equivalent to

$$\sin(\theta_1 + \theta_2) \geq 0 \tag{3.15}$$

Due to (3.12) and (3.15) the finite element mesh criterion (Crit. 3.3) can be expressed in two dimensions as

$$\theta_1 + \theta_2 \leq 180° \tag{3.16}$$

In two dimensions (3.7) describes the relation between the circumcircle radius, the edge length, and the opposite angle in a triangle. In three dimensions a relation for the circumsphere radius, the edge length, and the opposite dihedral angle θ_k (which is important for Crit. 3.3) in a tetrahedron does not exist as was explained in Fig. 3.3. This leads to a very interesting conclusion. It can be shown due to the existing relation (3.7) that the finite element mesh requirement (Crit. 3.3) is in two dimensions identical to the box integration mesh requirement which is based on empty circumcircles (Crit. 3.1 and Delaunay Crit. 5.2). To see

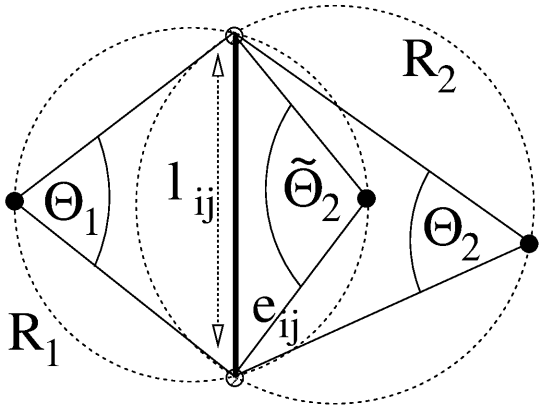


Figure 3.5: Finite element mesh criterion for two dimensions.

this equivalence of the angle condition (3.16) and the Delaunay criterion dependent on (3.7) consider the extreme case where (3.16) becomes

$$\theta_1 + \theta_2 = 180^\circ \tag{3.17}$$

It follows that

$$\sin\theta_1 = \sin\theta_2 \tag{3.18}$$

and furthermore with (3.7)

$$\frac{l_{ij}}{2R_1} = \frac{l_{ij}}{2R_2} \tag{3.19}$$

The two triangles with the common edge $e_{i,j}$ (Fig. 3.5) must possess circumcircles with equally sized radii. Because of (3.17) the circumcircles must be in fact identical. Each circumcircle passes therefore through all four vertices of the two triangles and the Delaunay criterion is "just" fulfilled. With a decreasing sum $(\theta_1 + \theta_2)$ the distance between the two circumcenters (centers of the circumcircles) increases and the Delaunay criterion is definitely satisfied.

As expected, in two dimensions the finite volume and the finite element method lead to the same discretization with identical requirements. They both rely on Delaunay meshes to fulfill the maximum principle. For other finite element applications than diffusion, like stationary problems or problems with less high gradients, the use of Delaunay meshes can be omitted. In three dimensions Crit. 3.3 and the Delaunay criterion are of quite different nature as will be shown with simple examples in the next section. In practice finite element mesh generators may generally try to avoid extremely obtuse (dihedral) angles and badly shaped elements without too much concern on the Delaunay property and without a technique to directly enforce Crit. 3.3. Such a technique remains open to further research.

3.2.3 Simple, Distinctive Mesh Examples

The pure diffusion equation is solved with the finite element and the finite volume method using *AMIGOS* [128]. This allows the comparison of the solutions on identical meshes with the same linear solver. A Gaussian profile is used as the initial distribution. In two dimensions

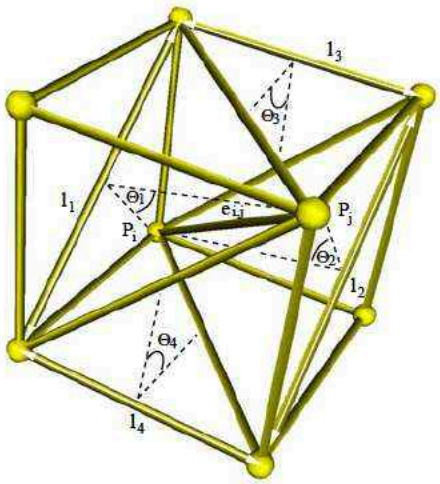


Figure 3.6: T_6 tessellation and Crit. 3.3.

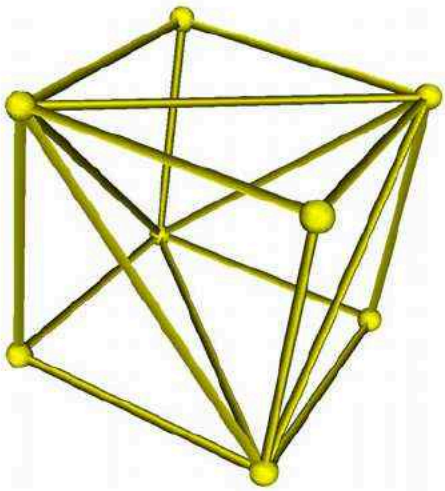


Figure 3.7: T_5 tessellation, no obtuse dihedral angles.

correct and identical results are obtained with both methods on Delaunay meshes. In three dimensions the finite volume method still achieves correct results on a Delaunay mesh as expected. However, the finite element method fails on the same three-dimensional Delaunay mesh. Even for such a simple test problem the finite element solution strongly violates the maximum principle. The resulting concentration reaches negative values in some areas. These areas spread out in time and the absolute value of the emerging negative concentrations is much larger than the minimal initial concentration. The relative error between the solutions of the two methods oscillates strongly and is large in regions where the concentration is negative. These negative concentrations are particularly annoying for diffusion applications in semiconductor process simulation where the concentration varies many orders of magnitude within a small area.

In the following the observed effects are investigated in terms of mesh requirements and simple mesh examples are constructed where the finite element method can be applied successfully to the diffusion problem.

Mesh Example 1: A Delaunay mesh which is not suitable as a finite element mesh for diffusion applications.

Mesh Example 2: A Delaunay mesh which is suitable for finite element diffusion simulation.

Mesh Example 3: A non-Delaunay mesh with obtuse dihedral angles which is still suitable as a finite element mesh.

The three presented meshes illustrate the different scope of the Delaunay criterion and Crit. 3.3. They prove that in three dimensions the Delaunay criterion is *neither sufficient nor necessary* to obtain a correct finite element mesh for diffusion so that the maximum principle is fulfilled. The examples also show that a strict adherence to a sole non-obtuse

angle criterion is not necessary. This important insight complements previous research [89] where one example, a Delaunay mesh insufficient for such applications, was given.

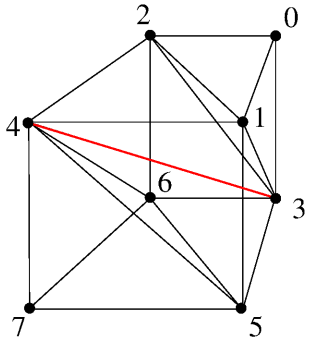
The examples were constructed by exploiting an ortho-product point distribution. A cube defined by eight points can be tetrahedralized in two qualitatively different ways.

T_6 **Tessellation:** A cube is composed of six tetrahedra (Fig. 3.6).

T_5 **Tessellation:** A cube is composed of five tetrahedra (Fig. 3.7).

For comparison purposes a specific tessellation T_6 is used which contains sliver elements with obtuse dihedral angles. The tessellation T_5 on the other hand does not contain such elements. Note that also T_6 tessellations exist which do not contain obtuse angles. The key idea is that all elements of both tessellations fulfill the empty circumsphere Delaunay criterion (Crit. 5.3), because all points lie on the perimeter of a single sphere. On the other hand Fig. 3.6 clearly shows that the finite element mesh requirement (Crit. 3.3) is not met by the chosen T_6 tessellation. It is only met by the T_5 tessellation, because of the total absence of obtuse dihedral angles. A simulation with *AMIGOS* allows to display the stiffness matrix and hence to directly check the sign of the matrix entries. The resulting stiffness matrix of a T_6 tessellation is shown in Fig. 3.8 where the entries for edge $(3, 4)$ of the mesh are underlined.

$$\begin{pmatrix}
5.0 & -1.6 & -1.6 & -1.6 & & & & \\
-1.6 & 8.3 & 1.6 & -1.6 & -5.0 & -1.6 & & \\
-1.6 & 1.6 & 8.3 & -1.6 & -5.0 & & -1.6 & \\
-1.6 & -1.6 & -1.6 & 11.6 & \underline{3.3} & -5.0 & -5.0 & \\
 & -5.0 & -5.0 & \underline{3.3} & 11.6 & -1.6 & -1.6 & -1.6 \\
 & -1.6 & & -5.0 & -1.6 & 8.3 & 1.6 & -1.6 \\
 & & -1.6 & -5.0 & -1.6 & 1.6 & 8.3 & -1.6 \\
 & & & & -1.6 & -1.6 & -1.6 & 4.9
\end{pmatrix}$$



TET: 4 5 6 3

$$\begin{pmatrix}
1.6 & -1.6 & -1.6 & \underline{1.6} \\
-1.6 & 3.3 & 1.6 & -3.3 \\
-1.6 & 1.6 & 3.3 & -3.3 \\
\underline{1.6} & -3.3 & -3.3 & 5
\end{pmatrix}$$

TET: 6 4 3 2

$$\begin{pmatrix}
3.3 & 0 & -1.6 & -1.6 \\
0 & 1.6 & \underline{0} & -1.6 \\
-1.6 & \underline{0} & 1.6 & 0 \\
-1.6 & -1.6 & 0 & 3.3
\end{pmatrix}$$

TET: 3 4 1 2

$$\begin{pmatrix}
1.6 & \underline{1.6} & -1.6 & -1.6 \\
\underline{1.6} & 5 & -3.3 & -3.3 \\
-1.6 & -3.3 & 3.3 & 1.6 \\
-1.6 & -3.3 & 1.6 & 3.3
\end{pmatrix}$$

TET: 4 5 3 1

$$\begin{pmatrix}
1.6 & 0 & \underline{0} & -1.6 \\
0 & 3.3 & -1.6 & -1.6 \\
\underline{0} & -1.6 & 1.6 & 0 \\
-1.6 & -1.6 & 0 & 3.3
\end{pmatrix}$$

Figure 3.8: Global stiffness matrix for a T_6 tessellation and local matrices of those four elements which are adjacent to edge $(3, 4)$.

Suitable meshes for simulation are then built by stacking a large number of such tessellated cubes. The typical characteristics of each tessellation type are thereby conserved. Hence, both meshes are global Delaunay meshes and yet only one satisfies Crit. 3.3. The two fundamentally different meshes based on an *identical* ortho-product point cloud are depicted in Fig. 3.11 and Fig. 3.12. The finite element simulation on the T_6 type Delaunay mesh results in negative concentrations as was previously pointed out. The T_5 type Delaunay mesh which fulfills Crit. 3.3 indeed succeeds to yield the required entries in the stiffness matrix and the concentrations remain positive at any time during the transient simulation.

The most important fact however is shown by the third example. Further exploiting the ortho-product point set and its T_5 type tessellation with slightly shifted points in certain locations results in a *non-Delaunay* mesh which still satisfies Crit. 3.3. Figure 3.9 shows an instance of the mesh consisting of eight cubes. The point in the middle has been shifted. The Delaunay criterion is violated, because the circumspheres of several unmodified tetrahedra contain the shifted point in its interior. The dashed line in the figure marks two of the non-Delaunay triangles. The simulation using *AMIGOS* for the entire mesh (Fig. 3.13) shows, that the requirements for the stiffness matrix are fulfilled. For example one can consider the edge $(14, 10)$ in Fig. 3.9. This edge is shared by elements which contain the shifted point and which are non-Delaunay. The matrix contributions of the six elements which are adjacent to this edge are given in Fig. 3.10. The first two matrices belong to the elements with the shifted point, and indeed possess undesirable positive off-diagonal entries. The last two matrices however belong to very well shaped elements which are able to compensate the overall sum. The resulting entry in the global stiffness matrix for the edge $(14, 10)$ equals $(0.83 + 0.83 + 0 + 0 + (-0.83) + (-0.83) = 0)$. Again, the concentrations do not reach negative values at all times. The shifting of a point introduces obtuse dihedral angles and positive contributions to off-diagonal elements of the stiffness matrix. However, Crit. 3.3 is satisfied and the stiffness matrix remains an M-matrix.

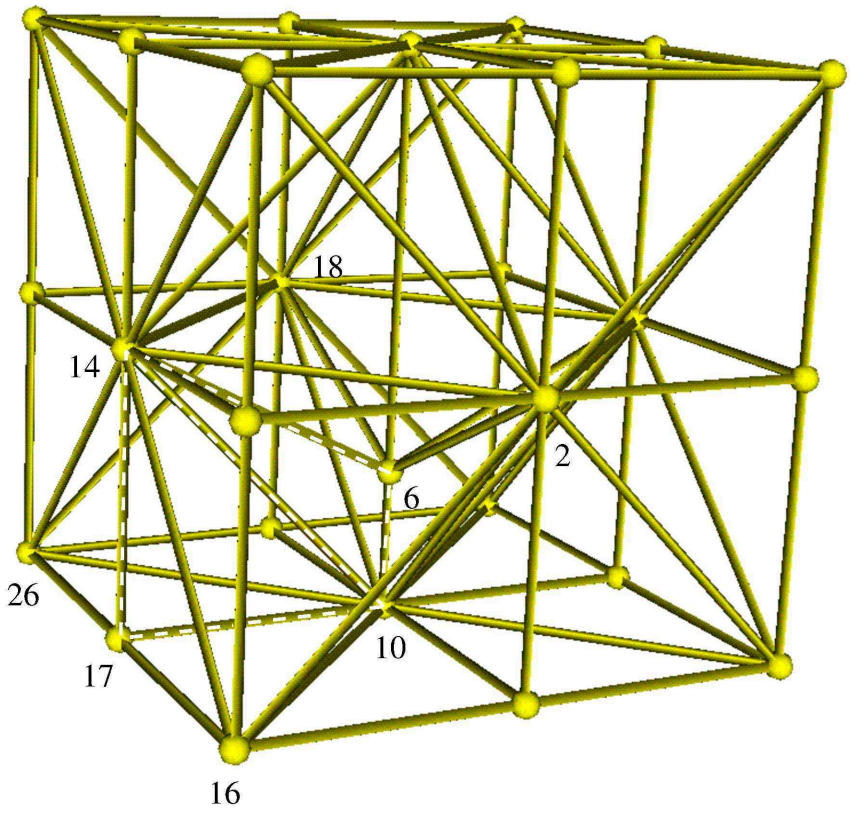


Figure 3.9: T_5 type tessellation with a shifted point.

TET: 10 6 14 2

$$\begin{pmatrix} 5 & -6.\dot{6} & \underline{0.8\dot{3}} & 0.8\dot{3} \\ -6.\dot{6} & 10 & -1.\dot{6} & -1.\dot{6} \\ \underline{0.8\dot{3}} & -1.\dot{6} & 0.8\dot{3} & -0 \\ 0.8\dot{3} & -1.\dot{6} & -0 & 0.8\dot{3} \end{pmatrix}$$

TET: 6 10 14 18

$$\begin{pmatrix} 10 & -6.\dot{6} & -1.\dot{6} & -1.\dot{6} \\ -6.\dot{6} & 5 & \underline{0.8\dot{3}} & 0.8\dot{3} \\ -1.\dot{6} & \underline{0.8\dot{3}} & 0.8\dot{3} & -0 \\ -1.\dot{6} & 0.8\dot{3} & -0 & 0.8\dot{3} \end{pmatrix}$$

TET: 10 17 14 26

$$\begin{pmatrix} 1.\dot{6} & -1.\dot{6} & \underline{0} & 0 \\ -1.\dot{6} & 5 & -1.\dot{6} & -1.\dot{6} \\ \underline{0} & -1.\dot{6} & 1.\dot{6} & -0 \\ 0 & -1.\dot{6} & -0 & 1.\dot{6} \end{pmatrix}$$

TET: 10 14 17 16

$$\begin{pmatrix} 1.\dot{6} & \underline{0} & -1.\dot{6} & 0 \\ \underline{0} & 1.\dot{6} & -1.\dot{6} & 0 \\ -1.\dot{6} & -1.\dot{6} & 5 & -1.\dot{6} \\ 0 & 0 & -1.\dot{6} & 1.\dot{6} \end{pmatrix}$$

TET: 2 10 16 14

$$\begin{pmatrix} 2.5 & -0.8\dot{3} & -0.8\dot{3} & -0.8\dot{3} \\ -0.8\dot{3} & 2.5 & -0.8\dot{3} & \underline{-0.8\dot{3}} \\ -0.8\dot{3} & -0.8\dot{3} & 2.5 & -0.8\dot{3} \\ -0.8\dot{3} & \underline{-0.8\dot{3}} & -0.8\dot{3} & 2.5 \end{pmatrix}$$

TET: 10 18 26 14

$$\begin{pmatrix} 2.5 & -0.8\dot{3} & -0.8\dot{3} & \underline{-0.8\dot{3}} \\ -0.8\dot{3} & 2.5 & -0.8\dot{3} & -0.8\dot{3} \\ -0.8\dot{3} & -0.8\dot{3} & 2.5 & -0.8\dot{3} \\ \underline{-0.8\dot{3}} & -0.8\dot{3} & -0.8\dot{3} & 2.5 \end{pmatrix}$$

Figure 3.10: Element matrices which contribute to the entry in the global stiffness matrix for the edge $(14, 10)$. Due to the symmetry of the mesh the three matrices on the left and on the right side possess the same entries.

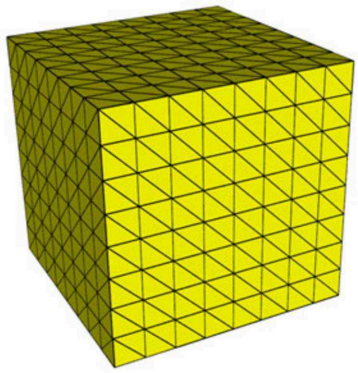


Figure 3.11: Delaunay mesh (T_6), 3072 tetrahedra.

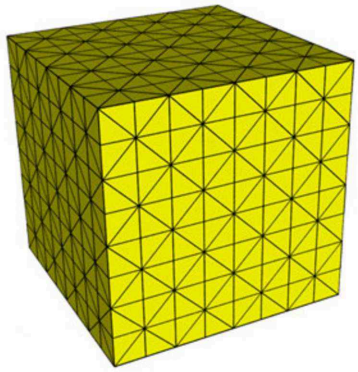


Figure 3.12: Delaunay mesh (T_5), 2560 tetrahedra.

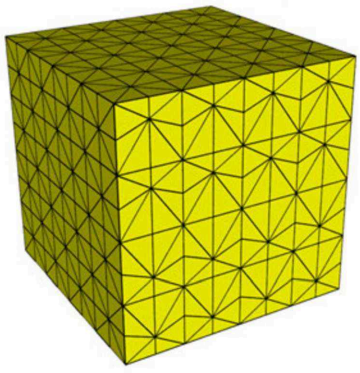


Figure 3.13: Non-Delaunay mesh, 2560 tetrahedra.

3.3 Control Space

In order for a specific application to influence the meshing process aside from geometrical concerns a *control function* must be defined. The stepsize h specifies the desired mesh spacing at a given location and a given direction.

$$h = \mathcal{F}(\mathbf{x}, \vec{d}) \tag{3.20}$$

For an isotropic mesh density the dependence on $\vec{d}$ can obviously be omitted. The control function is not likely to be prescribed analytically or manually. It usually depends on a key variable of the physical problem. Hence, the control function itself must be defined on a different mesh. This *background mesh* is used to evaluate the control function $\mathcal{F}$ of the discrete key variable given at the mesh points. The background mesh must at least cover the entire simulation domain and is present during the generation of the actual boundary consistent simulation mesh. Such a background mesh is often provided by a simple ortho-product grid or an octree structure. The conjunction of the control function and the background mesh is called the *control space*[3]. For example, in semiconductor device simulation a typical key variable is the electron concentration n. Let n be defined on a simple structural grid which is not consistent with the boundary. An appropriate definition of $\mathcal{F} = g(n)$ governs the mesh density. To achieve higher refinement and smaller elements in regions where the gradient is large one can define

$$\mathcal{F}(\mathbf{x}, \vec{d}) = \frac{\Delta_{\max}}{\alpha + |\vec{d} \cdot \nabla n|} \tag{3.21}$$

where α is a small regularizing term and $\Delta_{\max}$ is an approximate parameter for the maximum allowed increase of n within a mesh element. The spacing of mesh points h_x in the direction of the x-coordinate axis ($\vec{d} = \vec{e}_x$) should be

$$h_x = \mathcal{F}(\mathbf{x}, \vec{e}_x) < h_x^B \frac{\Delta_{\max}}{|\Delta n|} \tag{3.22}$$

where h_x^B is the spacing in the direction of the x-coordinate axis of the background mesh and Δn is $(n(\mathbf{x}) - n(\mathbf{x} + h_x^B))$.

The aim to accurately discretize the solution quantities or to choose a control function $\mathcal{F}$ which depends on the solution itself leads to the problem that the required element size is not known a priori. A previous simulation mesh with an existing solution must be used as a background mesh to judge whether or not the mesh density needs to be increased. There exists no knowledge of how pronounced this increase should be. A solution dependent measure $Q_{sol}(\mathbf{x}, \vec{d})$ must be defined to assess the existing mesh. To achieve an anisotropic mesh spacing this metric Q_{sol} must be capable to judge the mesh/solution with respect to the direction $\vec{d}$. Typically, Q_{sol} reflects interpolation errors, error estimates, second order derivatives, and error indicators. Numerous approaches can be found in literature [6, 9, 17]

$$\int_e |\nabla(u_{quad} - u_{lin})|^2 \, dx \tag{3.23}$$

$$\int_e (u_{quad} - u_{lin})^2 \, dx \tag{3.24}$$

[3] A slightly different concept of control space is given in [55].

$$\int_e |u_{ref} - u_{lin}|\, dx \tag{3.25}$$

where u_{quad} denotes a quadratic interpolant, u_{lin} a linear interpolant, and u_{ref} a reference solution. Error indicators or a posteriori error estimates for $(u - u_{lin})$ can be developed if specific properties of the solution, e.g. in the case of elliptic or parabolic problems, are known [10]. For anisotropic control the error contribution of each edge of the mesh element must be separately evaluated.

The control function $\mathcal{F}$ can be formally expressed as

$$h = \mathcal{F}(\mathbf{x}, \vec{d}) = g(h^B(\mathbf{x}, \vec{d}), Q_{sol}(\mathbf{x}, \vec{d})) \tag{3.26}$$

where h^B is the local mesh spacing of the background mesh. For some applications, e.g. periodic systems, such an absolute value for h serves as an upper bound to guarantee a certain accuracy. An explicit function for h is also needed for *global adaptation*. The entire domain is remeshed with new elements which are not created from simple refinement of the old elements. The old mesh is used purely as background mesh providing the old solution and not for the actual construction of the adapted mesh. The meshing process guided by a measure Q_{sol} becomes an iterative process which starts on an initial coarse mesh with a first solution. The initial mesh can only be controlled by geometrical constraints and/or key variables which are independent of the solution. However, most practical codes perform a less sophisticated adaptation. Adaptation usually means then *local adaptation* by means of local refinement. The function $\mathcal{F}$ is not known, instead threshold decisions based on Q_{sol} determine whether or not an element is locally split in half. The mesh is only adapted isotropically and locally to the solution.

3.4 Local Adaptation

Local adaptation is necessary to achieve accurate solutions with an acceptable effort in terms of simulation time and memory consumption. The local refinement, coarsening, or smoothing steps are performed to enhance purely geometrical quality aspects as discussed in Section 3.1, or are guided by a control function as explained in the last section. In the first case the refined regions concentrate around areas where the local feature size is small. Large elements which resolve small geometrical features are usually badly shaped and require refinement. In the latter case the mesh density is adapted to a stationary solution or dynamically for each timestep of a transient simulation. The regions of refinement have to migrate as the characteristics of the transient solution change over the domain. Essentially, local refining in some regions as well as local coarsening in other regions becomes necessary to avoid meshing the entire domain repeatedly. The here discussed refinement is often referred to as *h-refinement* which results in a decrease of the stepsize h. On the other hand are *p-refinement* techniques which increase the order of the polynomial form functions of the finite element approximation. Topics and techniques of local mesh adaptation are discussed in the following paragraphs.

3.4.1 Moving Boundaries

Element deformation, moving mesh points, and changing structure boundaries occur for instance during the oxidation step in semiconductor process simulation. The situation is often

referred to as a local adaptation problem, because portions of the mesh remain unchanged and should be reused. It is not feasible to repeatedly mesh the entire domain with every slight change of the structure boundary and to transfer the data to the completely new set of mesh points. However, difficulties arise when the device structure changes qualitatively. The existing surface mesh topology may not be suitable to represent the new boundaries and interfaces correctly. More global techniques might be necessary to extract a new surface mesh in such cases (see Section 3.5). In this sense moving boundaries during oxidation are related to moving surfaces during topography simulation of etching and deposition steps. Although in the latter case no internal mesh points are moving and the translation velocities are only applied to points of the surface, techniques from etching and deposition tools might successfully be applied to oxidation.

When the mesh is only distorted the following local adaptation steps suffice to maintain mesh quality and consistency.

1. The mesh points are moved and the elements are deformed. Ideally this happens only inasmuch as certain constraints are fulfilled. An effective and simple to implement constraint is to check the sign of the volume of the mesh element to avoid folding and overlapping elements. A mesh point can then only be moved as far as none of the volumes of the incident elements reaches a too small value. It might be necessary to cover the exterior of the structure with an outside mesh to fully detect all possible areas of collision.

2. Through a linear scan too small, too large, badly shaped, or extremely distorted elements are tagged. If no constraints enforcing consistency were applied in the previous stage, the detection of overlapping elements would be much more costly at this stage. If the mesh is "folded", elements with a negative volume could be detected easily, but their removal does not restore the consistency. A large number of positive elements would have to be checked as well for possible overlappings.

3. Tagged elements are removed with the usual techniques of refinement and local transformation (see following paragraphs). This changes the internal mesh topology (not the topology of the surface) so that a further translation of the mesh points including previously constrained points becomes possible. It is returned to step one and the process is repeated as long as changes are significant or the end time of the simulation is not reached.

3.4.2 Hierarchical Meshes

The initial mesh is coarse. Local refinement is performed by recursive subdivision that follows varying simple patterns. All recursion levels are stored in a tree data structure. Local coarsening of a previously refined area can be performed easily by traversing the tree. Such hierarchical meshes are typically employed for multigrid methods [171] which require changing from coarse meshes to refined meshes and vice versa iteratively. The tradeoff lies in a refinement limited to simple patterns, the higher memory consumption, and the more complex mesh management. The last issue is especially important for moving boundary situations. A carefully chosen tree data structure and a well designed implementation are

important to unify mesh updates for all levels of the mesh hierarchy. It can be advantageous to store only one mesh and to rely on fast remeshing techniques for the purposes of adjusting to a moved boundary as well as coarsening and refining.

3.4.3 Bisection and Projection

Refinement techniques mostly involve the bisection of mesh edges. It is a simple procedure which in practice achieves a smoother grading of the mesh density than is acquired by techniques like the orthogonal projection of a vertex onto the opposite edge of a triangle. The reason can be seen easily in two dimensions. If the angles in a triangular mesh are assumed to be not extremely obtuse, bisection guarantees that the minimum spacing between mesh points does not decrease much more pronounced than by a factor of 0.5 and no mesh points lie too close together. Obtuse angles might be introduced, but could be eliminated by topological modifications as dicussed later on. Orthogonal projection avoids bad angles, but even under the assumption of an average element quality it cannot be guaranteed that mesh points have a sufficient distance. Too close mesh points induce a high refinement in that region without that the physical problem calls for it. Self-induced refinement may lead to convergence problems of the meshing process if not properly addressed as discussed in Section 6.3.

A new bisection-based refinement method has been introduced by Liu and Joe [92, 93]. Interesting research on topics like the longest edge bisection and its "propagation path" has been accomplished by Rivara and Hitschfeld [67, 131, 132]. The order in which several refinement steps are performed has a great effect on the mesh. A lot of research has still to be done in three dimensions.

3.4.4 Red-Green Refinement

Red-Green refinement is based on the bisection of all edges of a simplex in one step. The resulting mesh topology which defines the connectivity of the split points is chosen such that the refined elements and the original element are self-similar. This is possible for the split element itself (*Red* refinement) but not for the neighbor elements of which not all edges are split (*Green* refinement). In such a way the original geometrical element quality can be preserved, but not improved. Hence, such a refinement is only justified when the mesh density needs to be increased according to the control space. The name "Red-Green refinement" in literature often refers to the two-dimensional case where the simplex is a triangle. The triangle is split into four triangles of the same shape and the adjacent triangles are each split into two triangles. The same effect could be accomplished with a more universal bisection technique combined with local transformations. Only the desired edges are split. With several adaptation steps including topological modifications the same refinement pattern can be reached as results from the Red-Green technique.

In three dimensions Red-Green refinement with mixed elements has been investigated by [88]. Splitting all edges of the three-dimensional simplex (tetrahedron) introduces six refinement points which define an octahedron (Fig. 3.14). The neighbor elements (Green region) form the transition from the refined area to the unrefined area. The implementation in *AMIGOS* combines this refinement pattern with the concept of hierarchical meshes as

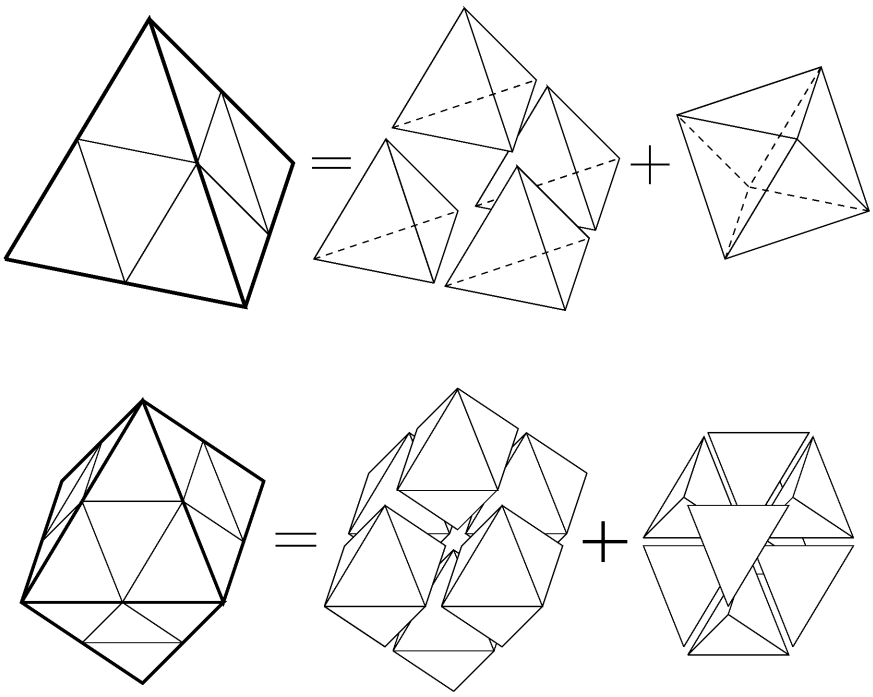


Figure 3.14: Red-Green refinement using mixed elements in three dimensions.

discussed above. The elements in the Green region are thereby not permanently fixed and are never refined themselves. They will be discarded and replaced by elements forming a Red pattern when further refinement is required.

3.4.5 Full Freedom Point Insertion and Removal

Arbitrary refinement can be performed by inserting new mesh points at any desired location not limited to e.g. the mid-points of element edges. If the topology connecting the new mesh point is not limited to a fixed pattern, more general algorithms for the local remeshing will have to be employed. If the new point does not a priori belong to the mesh element which motivates the refinement, a point location method will be required to determine the element that contains the point. Coarsening meshes which are not of the hierarchical type by means of point removal has less practical relevance. The most important application is to collapse too close points and to coarsen a too fine surface mesh (see Section 3.5). Such fully flexible techniques are required to improve the geometrical quality of the mesh elements or to introduce anisotropy. An important refinement method for angle bounded triangulations by means of arbitrary point insertion is based on the Delaunay criterion and will be discussed in Section 5.6.

3.4.6 Mesh Smoothing

Another way of adapting a mesh is the relaxation or smoothing of the mesh points. The position of the mesh points is modified, but the mesh topology remains unchanged. Shifting points can have a drastic effect on the quality of a mesh and it is more efficient than refinement and collapsing points especially when the translation amplitudes are small. A global optimum is reached by sequentially updating the mesh points in a Gauss-Seidel-type iteration [9, 49]. The location of each mesh point is derived from a local optimization of a certain criterion while holding all other mesh points fixed. Usually, only a few sweeps through the mesh points are required to achieve convergence, and the local optimization problem is solved with a damped Newton method.

More advanced approaches define the local optimum as a maximum of a combination of a geometrical quality and a solution dependent quality [9]. The local displacement of a mesh point is then naturally constrained by the geometrical quality of the incident elements. Other approaches use translation forces to define attraction and repulsion between mesh points depending on the distance and direction. The force function is crucial for convergence and is often defined similarly to the physical binding forces between atoms [190]. Consistency checks are necessary to ensure that the displacement of a mesh point does not result in overlapping or zero volume elements.

A simple and straight-forward method can be derived from a finite difference approximation of the Laplace operator and is called *Laplacian smoothing* [62]. A mesh point is moved to the centroid of the surrounding mesh points which are topologically connected.

$$P = \frac{1}{n} \sum_{i=1}^{n} \alpha_i P_i \tag{3.27}$$

Several sweeps through all mesh points are required. If a set of surrounding mesh points forms an extremely non-convex polyhedron/polygon its center of mass may lie outside and overlapping elements would be generated. In such cases the mesh point is skipped. The weighting factors α_i could be set to one, or for example be derived from the volume/area of the incident elements where $\sum \alpha_i$ is equal to n. Laplacian smoothing results in more homogeneous meshes and is not suitable to achieve or maintain an anisotropic mesh density. Only for the trivial case of an homogeneous anisotropic mesh, e.g. an ortho-product grid with constant spacing h_x, h_y over the domain where $h_x \ll h_y$, would the anisotropy remain unaffected after smoothing.

Another simple method aims to equally distribute the angle of the elements incident at each mesh point [129]. A general limitation of smoothing techniques is a bad mesh topology. If for example in two dimensions the number of edges incident at a given mesh point is too large or too small, it will become impossible to improve the angles. Most powerful will be a combination of refinement, smoothing, and local transformations.

3.4.7 Local Transformations

The edges in a mesh are locally modified while the mesh points are left unchanged. Such topological transformations of the mesh elements are an important technique to improve the

mesh and to get rid of imperfections in an efficient and fairly straightforward manner. In two dimensions transformations are simple and can in fact be restricted to a single type of operation on a set of two triangles. The diagonal edge is flipped which is often also called *edge swapping*. The same technique applied to surface triangulations in three dimensions is defined more precisely in Section 6.3. Topological changes for volume elements in three dimensions are much more sophisticated [71, 72, 75]. A transformation generally changes the number of involved tetrahedra and several types of flip operations have to be defined. The basic operation is a 2-3 or 3-2 flip where facet swapping respectively introduces or eliminates a third tetrahedron (Fig. 3.15). More complex operations involve five or more tetrahedra. As

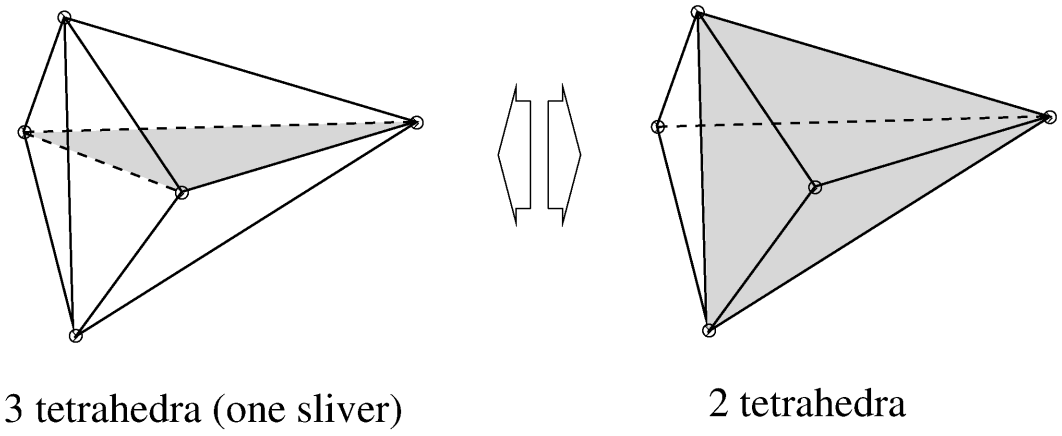


Figure 3.15: 3-2 or 2-3 local transformation. The internal facet which is being swapped is drawn shaded.

can be seen from Fig. 3.15 local transformations can be useful to remove slivers which only exist in three or higher dimensions. It should be noted that an algorithm performing only local transformations may get stuck in local optima.

3.5 Surface Mesh

The generation of a suitable surface mesh is not simple. The difficulty is that the surface description of a CAD model and surface meshing are two different tasks. The purpose of the first is purely the definition of the structure. The latter will require the first, but it will also depend on the generation type of the volume mesh.

An input model can be described in very different ways. Constructive solid geometry *CSG* uses boolean operations on solids to define a structure [99]. Alternatively, the same structure can be defined by a boundary representation *BREP*. The enclosing surface, in two dimensions the contour, is given through splines, bezier curves, non-uniform rational B-spline (*NURBS*) patches, or edges and polygons. The transition from a mere structure description to an actual surface mesh involves several preprocessing steps. A first conversion into a meshable surface description, as already mentioned earlier, is not trivial and depends on the type of solid modeling [97]. Figure 3.16 shows an example of a structure description which makes intersection calculations necessary to derive the boundary representation. After extracting

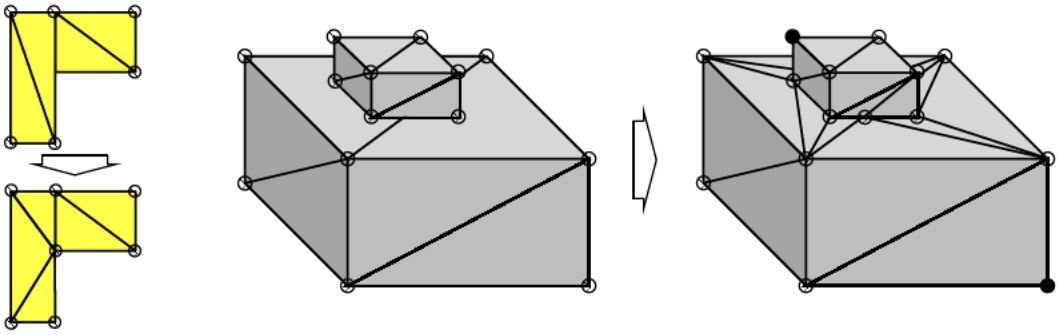


Figure 3.16: Splitting an edge to ensure a well connected surface topology. Correctly splitting a polygon requires an expensive calculation of all intersections.

the well connected surface topology, it must be adjusted to work with the generation type of the volume mesh.

A method based on graph theory and network flow techniques for bidirected flow problems has been developed by [111]. A boundary given by curved polygons is refined into quadrilaterals (four point mesh elements for surfaces in three dimensions) so that the resulting surface mesh is suitable for numerical analysis. The theoretical approach avoids typical standard traps where patches stay unresolved because the number of remaining points and edges does not allow a tessellation into quadrilaterals with the available templates.

Many tetrahedral mesh generators, e.g. most advancing front and Delaunay methods, require a surface triangulation. Furthermore, providing a refined, coarsened, or smoothened surface triangulation will prove efficient prior to the generation of the volume mesh. In some cases the elements of the surface mesh are required to fulfill certain mesh criteria as for example Crit. 3.1. Such special surface triangulations can be generated through complex refinement procedures. A new technique will be described in detail in Section 6.3. It will also be useful if the volume meshing method is capable to adapt the surface mesh a posteriori. Other methods, e.g. some cartesian and octree techniques, generate the surface mesh indirectly by calculating the intersections of the surface representation with the elements of the volume mesh. Again, a surface which is represented by a triangulation may simplify this process.

Where in computational fluid dynamics and related fields the triangulation of *NURBS* patches is important [2, 120], quite different solid modeling techniques are employed in semiconductor process and device simulation [188]. The structures might be defined through layout data [101]. Or they might be rasterized and composed of a large number of equally sized small cells (cellular data). Such an output, e.g. from topography simulation, contains too many cells to be directly used for the surface or the volume mesh. Simple cartesian solid modelers produce staircase-like surfaces where the normal vector of each facet is parallel to a coordinate axis (Fig. 3.17). These cases unnecessarily complicate the meshing process, and can be avoided with a preceding data reduction and/or smoothing step. Such techniques also gain importance for level set methods. An iso surface of a continous volumetric data set forms the structure description. Extracting a surface mesh from such an iso surface may not be straightforward because the data set is itself discretized on a mesh. Coarse mesh elements

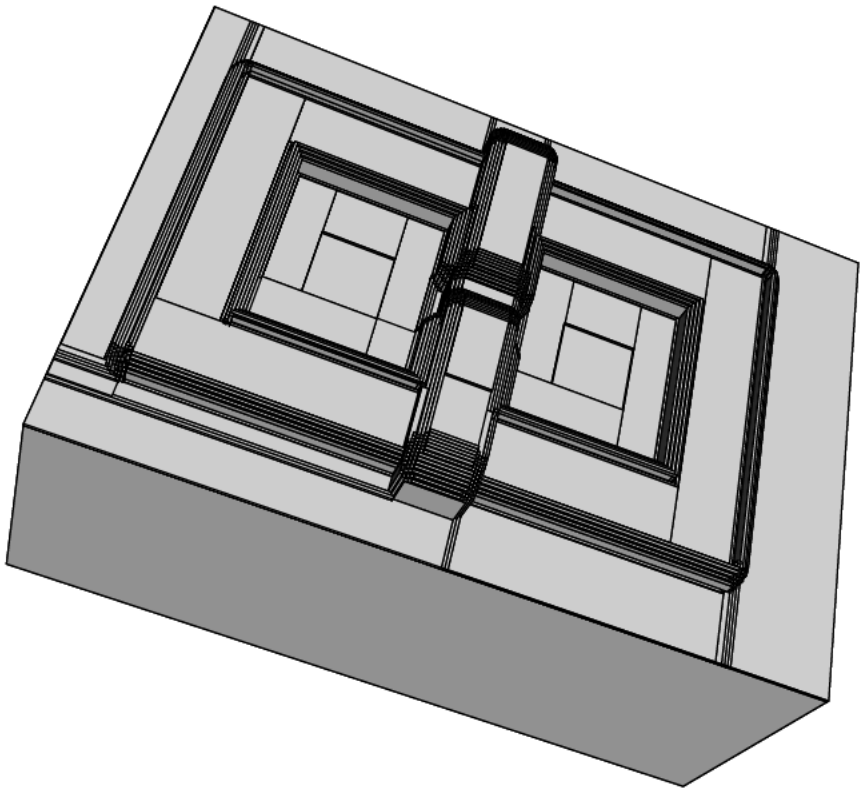


Figure 3.17: Staircase effects approximate slopes and result in unnecessary large meshes.

and interpolation can cause an extremely rough and unsuitable surface mesh.

3.5.1 Surface Extraction From Cellular Data

For certain applications a cellular data representation[4] enables the use of problem adapted and efficient algorithms. During the simulation of etching and deposition processes topological changes of the surface structure are handled without special surface trace algorithms or algorithms to avoid the formation of surface loops. At the cost of a higher memory consumption due to the sampling of the structure moving boundary situations are managed in a straightforward manner with image processing techniques. The polygonal surface description is later retrieved through the marching cubes algorithm [95]. The original method which gen-

[4] An image which consists of pixels in two and voxels in three dimensions.

erates fairly smooth iso surfaces for continous data is adapted in such a way that it constructs the interfaces between a discrete distribution of several materials as shown in Fig. 3.18 and Fig. 3.19. The value of the magnitude on the cell corners is thereby not arbitrary but rather an integer reflecting the material type. The cell edges are never split differently than through bisection which results in a less optimal representation with angles limited to a multiple of 45°. The selection of the templates depending on the material type at each cell corner is not always straightforward (Fig. 3.19). A detailed description of such an implementation can be found in [85]. During the sampling and retrieval process it will be desireable to preserve as much of the polygonal information as possible in areas where the boundary remains unchanged [110].

It is interesting to see that the original marching cubes algorithm which operates on continous data can be utilized to extract surfaces from level set computations on orthoproduct grids. Often such level set methods are performed on tetrahedral meshes and a to the marching cubes analogous algorithm becomes necessary.

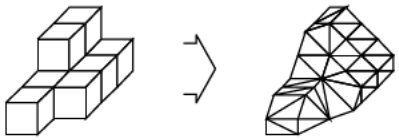


Figure 3.18: Marching cubes algorithm applied to discrete data which describes the distribution of a finite number of materials.

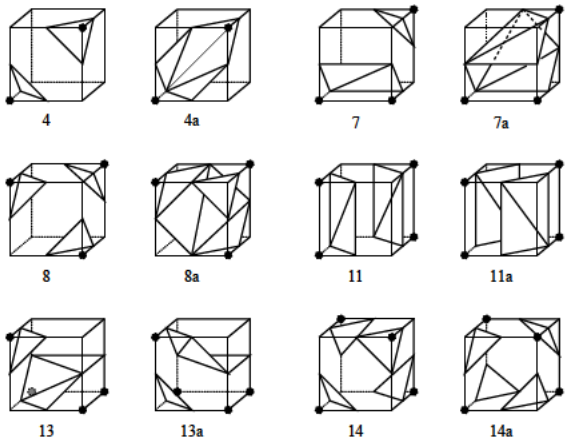


Figure 3.19: Pathological cases and alternative templates.

3.5.2 Surface Coarsening, Data Reduction

Surface triangulations can consist of a too large number of very small triangles which do not convey important geometrical information. For example surface triangles which have been extracted from cellular data are of the same size as the sampling cells. In this case data reduction is mandatory. An algorithm based on the standard method as proposed in [153] allows such a decimation of a general triangulation including edges with more than two adjacent facets [85]. By taking various parameters like the aspect ratio and the distance of a point to a plane into account, redundant points are discarded together with incident triangles. The resulting hole (ring) is triangulated. A smoothing effect can be observed under certain conditions but it cannot be guaranteed. The local operation and an example is depicted in Fig. 3.20. At the top of the trench the original structural edge has been converted into a less smooth surface. Depending on the locally applied parameters either the reduction effect is limited or ripple effects might occur. More global techniques, e.g. extracting all contour and

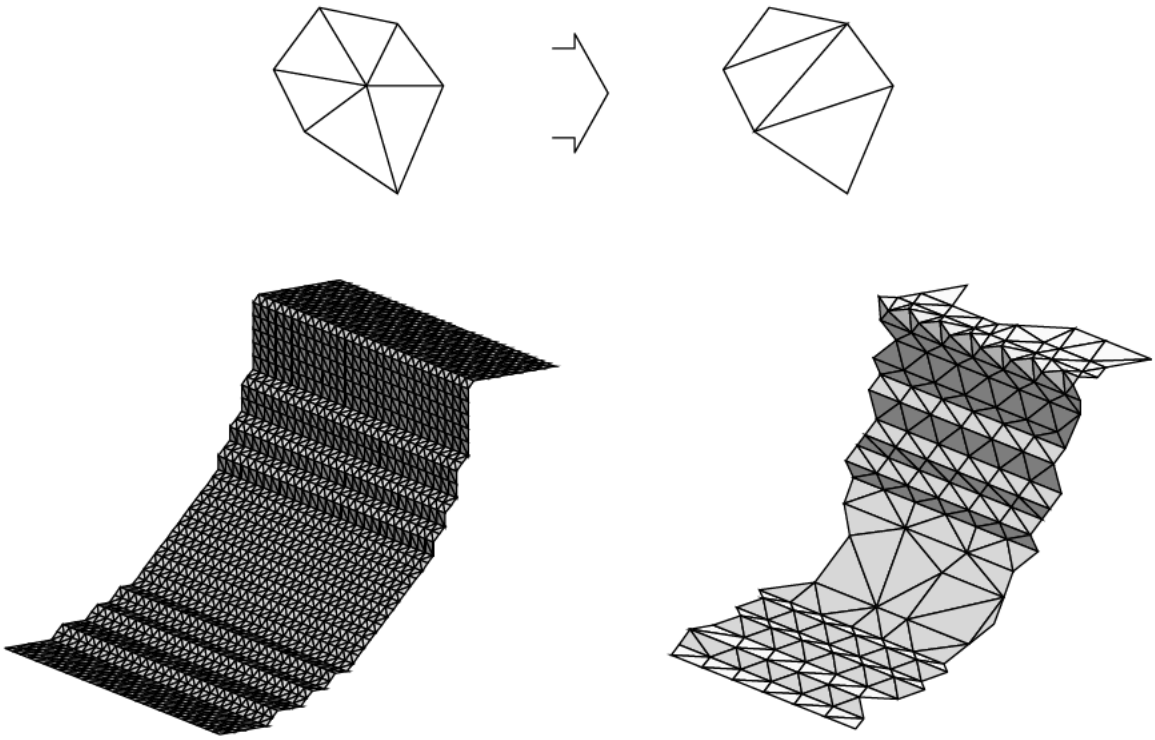


Figure 3.20: Detail of a trench consisting of 2912 triangles before and 288 triangles after data reduction by locally discarding points.

structural information beforehand (see Section 6.3) or generally elaborated schemes utilizing for instance energy functions [68] can be employed.

3.5.3 Surface Smoothing

Surface smoothing for the above mentioned purposes is similar to the function of a low pass filter. The critical question is how to get rid of ripples and staircase effects while conserving structure corners. Obviously this can only be achieved by either user interaction through parameters, or by preserving a complete information flow from the solid modeler to all preprocessors and to the mesh generator.

3.6 Point Placement

The distribution of the mesh points should only depend on the application and the device structure. In practice it is trivial to see that it also depends on the chosen mesh generation method. It is difficult to find and to generate with an algorithm the optimal placement of points without following some trial and error scheme. The mesh points are often sprayed over the simulation domain with a simple method and only afterwards adapted to the application/solution. Many existing and applied methods are not fully flexible with regard to

the placement of the mesh points. Two negative and frequently encountered effects can be observed:

- Rotating and translating a model relative to the coordinate system can have a heavy impact on the resulting mesh.
- The location of the mesh points is limited to certain positions within the coordinate system.

Some methods might even fail if the model is slightly tilted relative to the coordinate system. Other methods, e.g. advancing front methods, are fairly independent of such an alignment and produce better boundary-fitted meshes. Unstructured meshes do not possess a regular topology by definition. Some unstructured methods, e.g. cartesian and octree methods, exhibit a certain regularity concerning the distribution of the mesh points. Therefore it is useful to further distinguish unstructured methods with regard to the point placement mechanism. Throughout this text the term *fully unstructured* will refer to methods which allow an arbitrary placement of the mesh points.

Chapter 4

Methodologies

THE enormous variety of existing meshing algorithms and the many different hybrid methods make an extensive overview difficult. This chapter is an attempt to give a complete classification of the main techniques which form the basis for practical mesh generation methods. Some of the techniques are suitable for tetrahedral as well as hexahedral mesh generation. Hexahedral meshes are created from, or result in quadrilateral surface meshes as for example depicted in Fig. 4.1. Purely hexahedral methods like whisker weaving and the spatial twist continuum [180] for the generation of unstructured hexahedral meshes are not discussed.

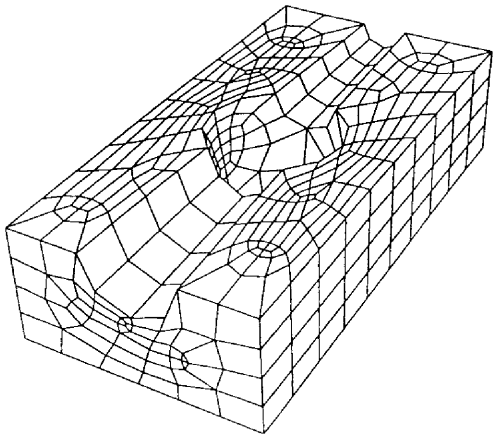


Figure 4.1: Unstructured quadrilateral surface mesh, MENTAT II [98].

4.1 Structured Grid Generation

Structured grids have a regular topology where the neighborhood relation between all points is captured with a two- or three-dimensional array. By incrementing or decrementing the array index the point neighbors can be directly accessed. Historically, structured grid generation was a first attempt to automate the meshing process for simple shapes. Through

mathematical transformations an ortho-product grid of the unit square/cube is mapped onto the simulation domain. In literature it is often referred to as *numerical grid generation* [24, 60, 78, 184]. *Multi block structured grid generation* combines several simple blocks to form more complex shapes. Cutting the simulation domain into these blocks is strictly speaking the same task as unstructured mesh generation and poses the same difficulties. The *Chimera* or *overlapping* approach allows overlapping blocks. Each structured grid component can be generated independently from the other parts. It must only be guaranteed that the blocks overlap sufficiently to allow interpolation between the component grids. A more detailed overview can be found in [61]. The two mathematically different mapping techniques are shortly summarized in the next paragraphs.

4.1.1 Algebraic Method

The parametrized boundary surfaces are algebraically combined to create the interior grid. The mapping of the unit square/cube with coordinates u, v, w onto the domain is performed by the transfinite interpolation [184]. In two dimensions it can be written as

$$\begin{aligned}\mathbf{M}(u,v) &= (1-v)\mathbf{B}_{bottom}(u) + u\mathbf{B}_{right}(v) + v\mathbf{B}_{top}(u) + (1-u)\mathbf{B}_{left}(v) \\ &\quad -(1-u)(1-v)\mathbf{V}_1 - u(1-v)\mathbf{V}_2 - uv\mathbf{V}_3 - (1-u)v\mathbf{V}_4\end{aligned}$$

where $\mathbf{B}$ are the boundaries, $\mathbf{V}$ the four corner vertices, and $\mathbf{M}$ is the mapping function to calculate the coordinates x, y in the real domain. The mapping function satisfies

$$\begin{aligned}\mathbf{M}(u,0) &= \mathbf{B}_{bottom}(u) \\ \mathbf{M}(u,1) &= \mathbf{B}_{top}(u) \\ \mathbf{M}(0,v) &= \mathbf{B}_{left}(v) \\ \mathbf{M}(1,v) &= \mathbf{B}_{right}(v) \\ \mathbf{M}(0,0) &= \mathbf{V}_1 \\ \mathbf{M}(1,0) &= \mathbf{V}_2 \\ \mathbf{M}(1,1) &= \mathbf{V}_3 \\ \mathbf{M}(0,1) &= \mathbf{V}_4\end{aligned}$$

The resulting grid is useful if the boundaries are convex or not too twisted. A three-dimensional example of the mesh of a LOCOS which was generated with a package described in [13] is shown in Fig. 4.2-a.

4.1.2 PDE Method

The mapping is a transformation based on a system of partial differential equations. The system is solved on a reference mesh to generate the structured grid. Depending on the type of the equations two classes can be distinguished.

- Elliptic grid generation
- Hyperbolic grid generation

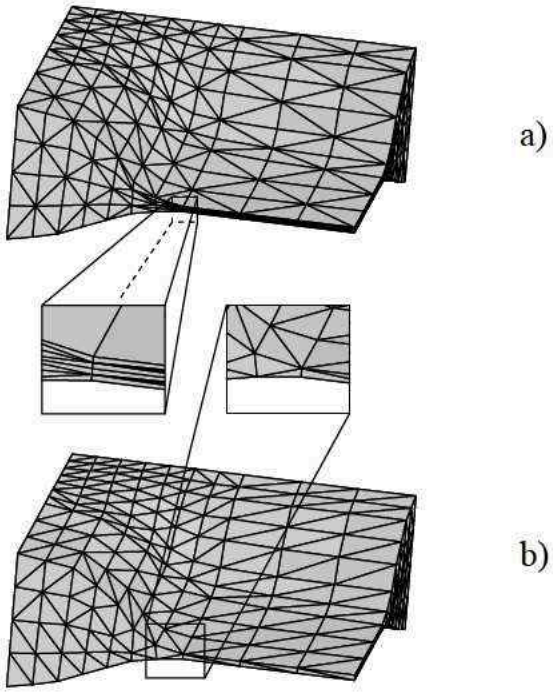


Figure 4.2: LOCOS: (a) structured mesh, 2000 tetrahedra (b) unstructured mesh, 957 tetrahedra.

The latter uses hyperbolic operators and is suitable to grow a structured grid from a boundary. The first commonly utilizes the regularizing properties of the Laplace operator [103, 168, 183]. In two dimensions two systems with corresponding boundary conditions are solved.

$$u_{xx} + u_{yy} = 0$$

$$v_{xx} + v_{yy} = 0$$

Extra control of the grid spacing and orthogonality can be introduced by using Poisson equations. The resulting grids are of high quality, boundary-fitted, and possess good orthogonality. The application of conformal mapping techniques to semiconductor device simulation, where boundary-fitted and possibly orthogonal meshes are a great concern, has been tested so far in two dimensions [26]. Non-planar thin layers (geometrical anisotropy) and protection layers (physical anisotropy near boundaries) should be manageable in three dimensions. In practice the structured grid must often become a part of a larger mesh with regions of different

anisotropic requirements. The region between the various structured grid parts could for example be filled with an unstructured mesh.

4.2 Product Methods

Product methods generate a n-dimensional mesh by multiplying a mesh of dimension $(n-1)$ with a one-dimensional mesh. Numerous variations can be found which exploit application dependent structural characteristics. The multiplication is performed by rotation (cylindrical topology) or by extrusion in the direction of one coordinate axis. The mesh of dimension $(n-1)$ can be of any type with triangle or quadrilateral elements. From the many possible combinations one particular approach has proven useful for interconnect simulations. Structures defined through layout data exhibit characteristics which can be exploited by the layer-based product method.

4.2.1 Layer-Based Method

The layered structure of many devices allows a simplified generation of three-dimensional meshes. Overlaying all layer descriptions into a two-dimensional graph unifies all lateral structural information (Fig. 4.3). An unstructured triangular mesh is generated for this

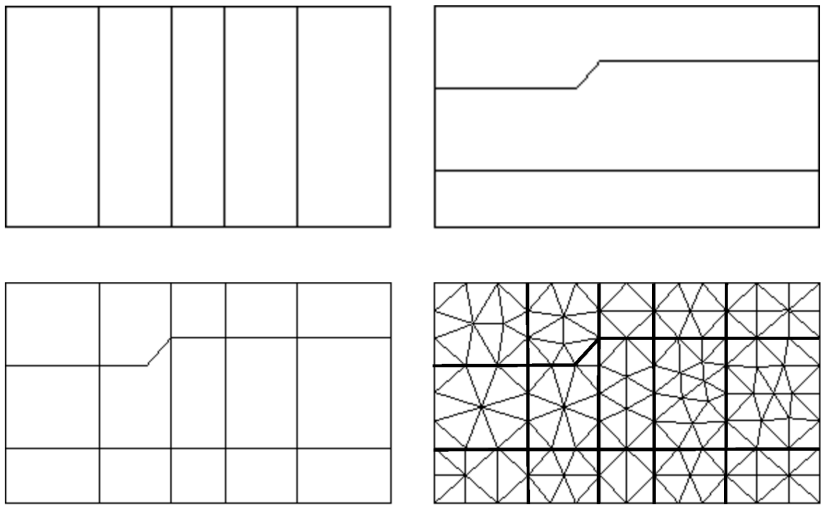


Figure 4.3: Overlaying two layer descriptions and lateral two-dimensional unstructured mesh.

graph. The resulting triangles are extruded into the third dimension and multiplied for every layer of the device. A following tessellation of the prisms yields the three-dimensional tetrahedral mesh. Local adaptation can be applied to the lateral unstructured mesh, but it will be propagated throughout the third dimension. A detailed description of this method can be found in [13]. Figures 4.4 and 4.5 show an example where a part of the interconnect layout is analyzed for parasitic effects. Regarding the suitability for anisotropy, lateral planar thin layers can be managed in a trivial way. The orthogonality of the elements resolving

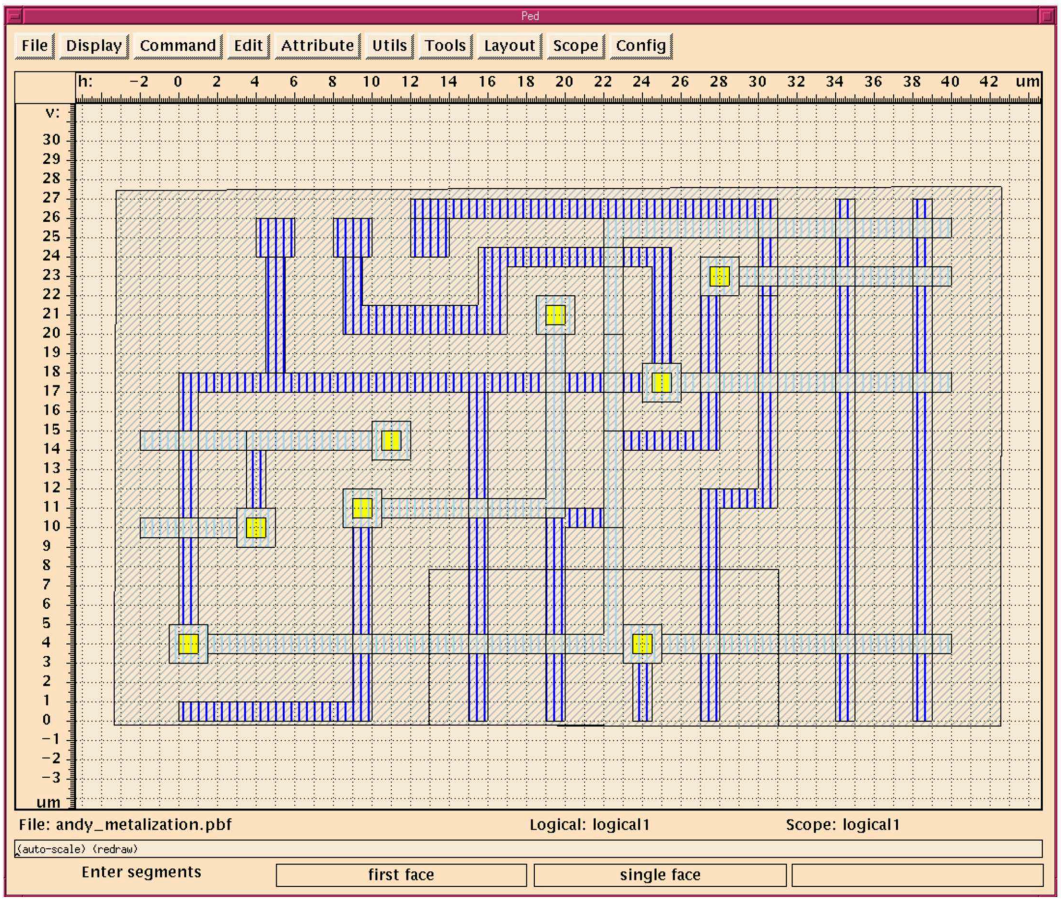


Figure 4.4: Layout structure description.

such a lateral thin layer is a trivial and welcome by-product of this method. With some limitations thin layers formed by parallel planes which are normal to the lateral expansion — they lie in the direction of the extrusion — can be managed as well: If the two-dimensional unstructured method is not capable of producing anisotropic two-dimensional elements for the lateral structure, the resulting lateral mesh will contain a large number of very small triangles. Due to the missing third dimension the number will still be at limits manageable. The following extrusion will introduce a number of needle elements which has the same order of magnitude as the number of small triangles. Hence, the thin layer which is normal to the lateral orientation is not ideally meshed but managed. Arbitrary thin layers (non-planar, non-parallel, non-aligned) cannot be handled with this method.

4.3 Cartesian and Octree Methods

A naive and intuitive approach is to generate a simple regular grid which entirely covers up the simulation domain. For example, an ortho-product point cloud can be used to fill the bounding box which is defined by the minimum and maximum of each coordinate with mesh

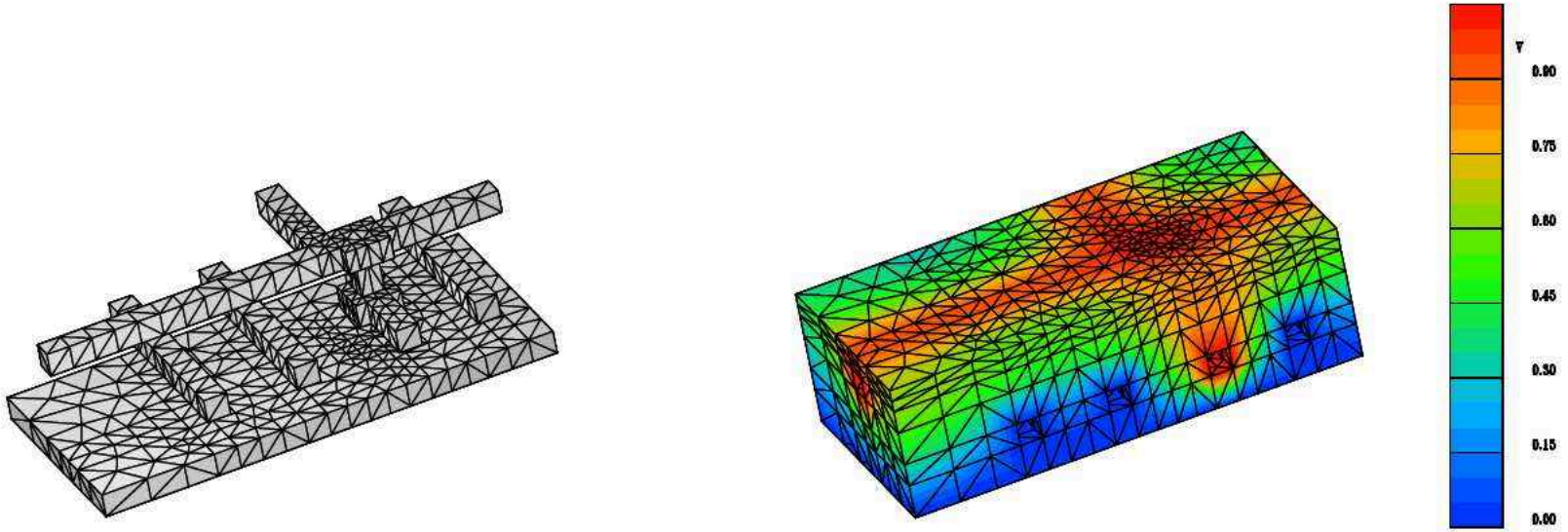


Figure 4.5: Interconnect simulation of a part of the layout using a layer-based product mesh.

points. The cells are then classified according to their location.

- The cells in the interior of the structure boundaries are used as mesh elements.
- The cells which completely lie outside of the simulation domain are suppressed.
- The cells which intersect the boundaries have to be split into mesh elements and exterior parts.

The selection of the stepsize or cell size is crucial to resolve features of the boundary and to limit the complexity of the split cells near the boundaries. Unfortunately, the resulting surface mesh cannot be controlled well. It is a matter of luck how the boundaries intersect the cells. Sharp intersections or cells which are cut off close to their corners result in very thin or degenerate elements. This situation becomes worse when the boundary is slightly tilted relative to the coordinate system (Fig. 4.6). Depending on the type of solver and discretiza-

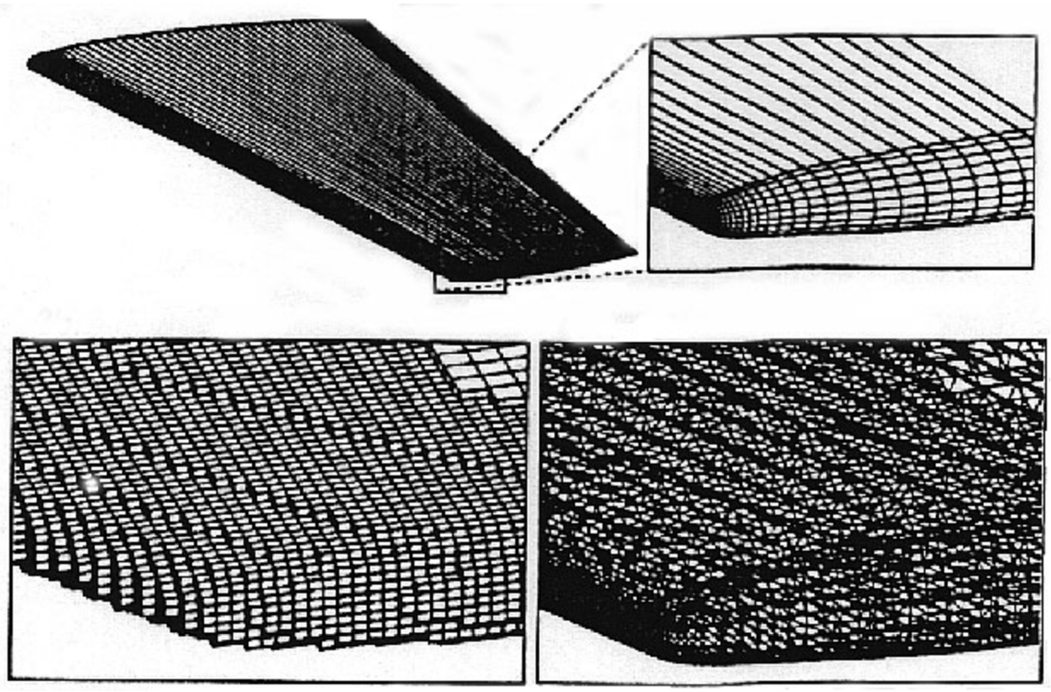


Figure 4.6: Intersection of the cartesian cells with the boundary, M. Berger et al. [2].

tion the cell elements must be further processed. Interior cells can be split into tetrahedra according to a set of templates. For the cells near the boundary templates and distortion (warping, [141]) might not be sufficient and a more general subdivision into tetrahedra may become necessary. The evolving mesh is boundary consistent but not boundary-fitted, and its topology will generally be of an unstructured nature.

An important remedy to the evenly sized mesh across the entire simulation domain is to allow the local subdivision into smaller cells. This leads to octree decomposition techniques with tetrahedral and even hexahedral [148] mesh templates. Spatial tree data structures

provide great efficiency in storing and locating geometrical information [142]. A set of rules is applied to each node of the tree until a state of subdivision is reached where the leafs of the tree contain single entities of the boundary. These rules must be designed carefully, especially under finite precision arithmetics, to avoid deadlocks and to guarantee a subdivision process which terminates. The *order of irregularity* of the tree is defined as the maximum difference between the levels of subdivision of neighboring leafs. For instance an order of one guarantees that a cell's edge cannot be smaller or greater than half or twice the length of the neighbor cell.

A tradeoff which is quite important to observe lies in the selection of the irregularity of the tree. A one irregular tree enables the use of less complex templates to further tessellate the cells. It will be easier to construct elements with non-obtuse angles [116]. For example an interior quad cell (not intersected by the boundary) with four edges of which any combination is split in half can be fitted with non-obtuse triangular templates. Note that this is valid only for square quads and not general rectangles. This advantage is inherited from ortho-product grids where non-obtuse triangular/tetrahedral templates are trivial due to the rectangular cells. However, a one irregular tree induces a very high number of mesh elements, because the refinement propagates through large portions of the mesh. In fact it is not unusual that the number of mesh points becomes unmanageable as experiments with point clouds generated from one irregular octrees have shown [192]. Higher irregular trees complicate the vocabulary of templates to fit all possible cell constellations and make the creation of good elements difficult.

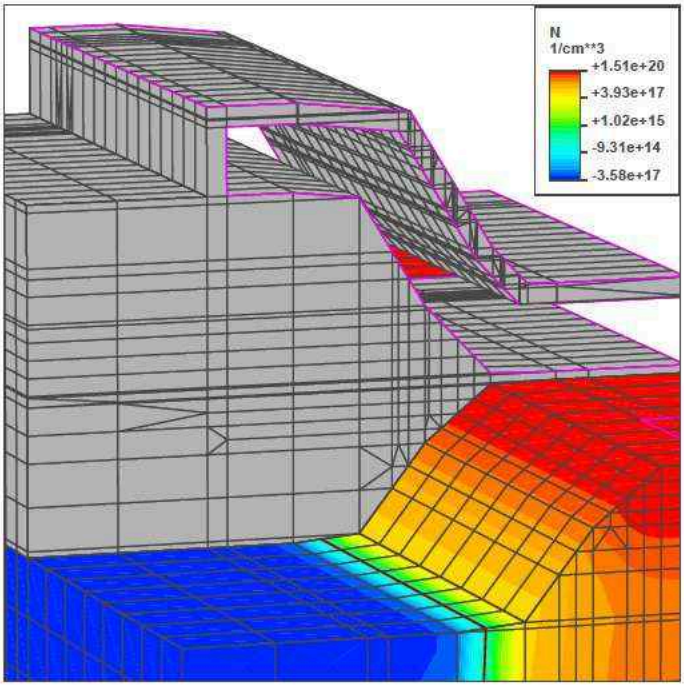


Figure 4.7: Intersection based octree mesh of Flash EEPROM, ISE ETH [52].

The long history of octree methods has lead to quite elaborated schemes [22, 152, 159]. However, the difficulties to control the element quality near boundaries also with regard to the mesh requirements for surface triangles are inherent. Boundary-fitted meshes cannot be achieved. Anisotropy can only be implemented in a limited way by intersection based instead of bisection based splitting of the cells. Thereby, it is difficult to maintain the good properties of the original octree and some advantages like the possibility to define non-obtuse triangular/tetrahedral templates are lost. Thin layers can only be managed as long as they are planar, parallel, and have a normal vector of which two components are equal to zero (Fig. 4.7). Thin layers with a normal vector of which one component is zero can theoretically be managed similar to the layer-based method with needle elements. As long as these structural requirements are met, octree methods will be of practical relevance. Quite advanced methods which can handle such a limited kind of anisotropy are applied in practice. However, it is questionable how much further such methods can be developed and how significant the improvements can be.

4.4 Advancing Front Methods

The mesh is constructed by progressively adding mesh elements starting at the boundaries. This iteration results in a propagation of a front which is the border (internal boundary) between the meshed and the unmeshed region. The difficulty with this method lies in the merging of the advancing front. A new triangular/tetrahedral mesh element is added by inserting one new point[1]. The location of this point is crucial and is determined by the following criteria.

- The quality of the resulting element.
- The desired mesh spacing given by the control space.
- Neighborhood constraints like other parts of the boundary/front.
- The point must be inside of the domain.

Adjusting the location of a point to meet all of these criteria requires a very complex analysis of the surrounding region. A previously generated point which belongs to a different part of the front may have to be preferred. In such a case no new point will be created and the two parts of the front merge. This results in a decrease of the size of the active front. The final step of the mesh iteration is a full merge where no active front remains. A few example situations are depicted in Fig. 4.8. The last two criteria must be checked by extensive search algorithms and intersection tests.

- None of the created edges which are connected to the new point may intersect any edge/facet of the two-dimensional/three-dimensional front.
- None of the created facets containing the new point may intersect any edge of the three-dimensional front.

[1] Quadrilateral/hexahedral elements require more points.

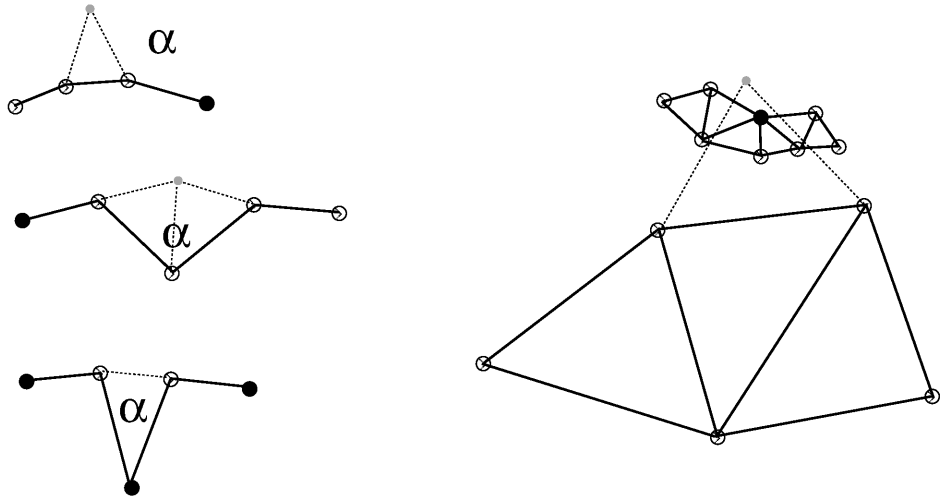


Figure 4.8: Various situations in two dimensions and different patterns depending on the angle α.

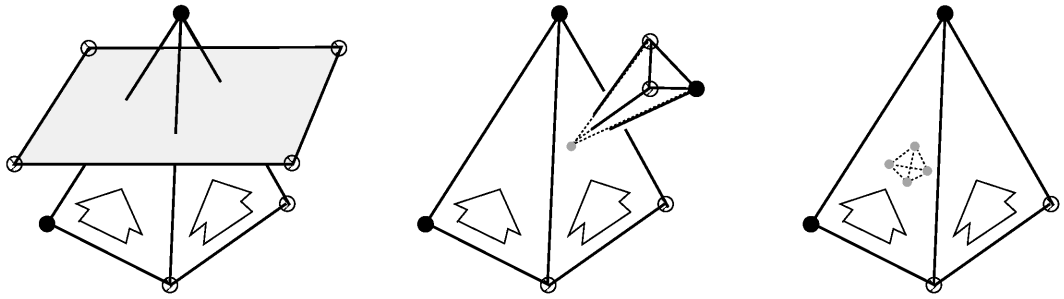


Figure 4.9: Three necessary tests to avoid collisions in three dimensions. The arrows show the direction of the advancing front and the tetrahedron which is tested and built.

- The new element must not contain any other points.

In three dimensions the second test is required additionally to the first. All three cases are illustrated in Fig. 4.9. If a more or less ideal initial guess for the location of a new point fails the intersection tests, the question how to alter the position successfully will arise. Sacrifices with regard to element quality and mesh spacing have to be made. A background mesh as explained in Section 3.3 can be utilized to evaluate the neighborhood constraints and to help the decision process. For example a volume mesh of the boundary/front without internal mesh points, a *boundary mesh*, contains much required information like distances and closest neighbors. This "background boundary mesh" must be locally updated each time the front advances. A Delaunay mesh seems most suitable for such purposes and was integrated as a background boundary mesh into an advancing front method by [50] as already briefly discussed in Section 2.3.

The surface mesh of the boundary from which the advancing front departs has a great influence on the volume mesh. A high quality surface triangulation is required for the generation of a tetrahedral mesh. Which and in what order the surface triangles are effectively

used as launching seeds for the advancing front is another important opportunity to influence the final mesh. In fact, it is the advantage of advancing front methods that the elements near the boundary can be controlled directly. Protection layers around physically crucial boundaries can be constructed by departing from those boundaries with several very small stepsizes. However, orthogonality is not a priori guaranteed by the original advancing front method. Enhancements to the algorithm are needed to provide perpendicular and parallel grid lines near boundaries. Arbitrary thin layers pose no difficulties in theory, but in practice require intelligent mechanisms to evaluate the neighborhood constraints. The advancing front method produces fully unstructured meshes and allows general anisotropic mesh elements.

4.5 Delaunay Methods

The Delaunay Triangulation which will be discussed in detail in Chapter 5 can be efficiently utilized as robust tetrahedralization engine for practical meshing applications. A Delaunay based meshing approach is a concept which consists of two tasks. One addresses the mesh topography which is defined through the placement of mesh points. The other task is to create the mesh topology by performing the Delaunay Triangulation for a known point set. The sequence in which the two tasks are carried out is a matter of choice and therefore two classes can be distinguished.

- The mesh points are first created by a variety of techniques. The Delaunay Triangulation is performed afterwards on the complete set of points.
- The Delaunay Triangulation is first computed for the boundary without internal points (boundary mesh). The mesh points are then inserted incrementally into the triangulation/tetrahedralization and the topology is updated according to the Delaunay definition.

The two approaches can also be combined. Both gain advantages from the Delaunay Triangulation. The creation of an initial set of points by e.g. advancing front, octree, or structured methods is much easier, because the constraint to avoid overlapping or inconsistent elements (in this case only points) is much simpler to handle. All topological issues will be addressed later by the Delaunay Triangulation in a rigorous and precisely defined way. On the other hand a Delaunay boundary mesh provides a lot of information on the structure. Internal mesh points can be added in an intelligent way (in what order and where) based on the size and location of the elements of the boundary mesh[2]. The local topology update after the insertion of a point poses no problem and is taken care of by many Delaunay algorithms[3].

A tetrahedralization engine based on the Delaunay Triangulation allows a fast generation of elements without for instance the expensive intersection tests of advancing front methods. The reason can be seen in the global definition of the topology[4]. It is therefore not necessary to check and compare one region of the mesh with every other region to verify the mesh

[2] For instance see Section 5.6.
[3] Section 5.3.
[4] This is strictly only valid for a unique Delaunay Triangulation, Section 5.4 and 6.4.3.

consistency. Instead only a locally confined test of the Delaunay definition is required and the knowledge of other regions is implicit.

Fully unstructured meshes with anisotropic capabilities can be generated. Figure 4.10 depicts a detail of the Flash EEPROM in comparison to Fig. 4.7. However, the concept where

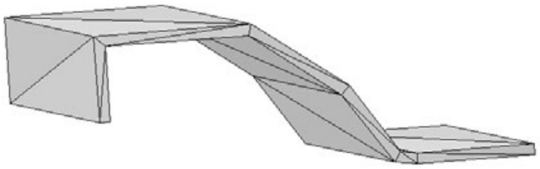


Figure 4.10: Boundary mesh of the floating gate structure, 36 tetrahedra.

the mesh topography is dealt with separately from the mesh topology is for anisotropic applications not always advantageous. The topology for a given set of mesh points might not be as expected and might destroy the desired anisotropy. For the last of the following anisotropic schemes the construction of protection layers depends on the ability of the Delaunay method to cope with thin layers.

- The mesh points are for example distributed in an advancing front style with a small stepsize compared to the lateral distances. The tetrahedralization engine is supplied with the point set. Adverse effects can occur when additional points are inserted and the topology is updated.
- In two dimensions the triangulation engine is supplied with grid lines to enforce a certain topology. It can be advantageous if these grid lines are not split. A Delaunay Triangulation can be employed without the insertion of additional points[5]. The Delaunay criterion for elements near those grid lines and at the boundaries is not necessarily fulfilled.
- The triangulation/tetrahedralization engine is supplied with grid lines/facets. The Delaunay Triangulation results in a refinement and the Delaunay criterion is fulfilled. An anisotropic refinement technique which can handle thin layers is mandatory.

All types of thin layers including planar and bounding box aligned layers require sophisticated algorithms to be managed well. The ideal case is when a minimum of refinement is induced which occurs at corner regions of thin layers to satisfy the Delaunay criterion. How many refinement points are required and how close they must lie to a corner vertex depends on the angles formed by a non-planar thin layer in relation to its thickness. In the example of Fig. 4.10 absolutely no refinement was necessary. A thinner layer containing sharper angles complicates the situation.

[5]The constrained Delaunay Triangulation can only be guaranteed in two dimensions, see Section 5.5.

Another characteristic of Delaunay methods which is important for meshing applications is the global optimization of the element quality[6]. Unfortunately it is not always identical to the desired mesh quality, e.g. non-obtuse dihedral angles or more specific mesh requirements as described in Section 3.2. However, interesting results have been obtained which show that a three-dimensional Delaunay Triangulation combined with a local optimization of the required mesh quality is superior to methods which only perform local optimizations without concern of the Delaunay criterion [71, 73, 100]. Local improvements of the dihedral angle results in a topology which can be globally far from optimal. Alternating between the Delaunay criterion and a minimum maximum dihedral angle criterion can achieve better results.

[6]The max-min and min-max-min Delaunay properties are discussed in Section 5.1.

Chapter 5

Delaunay Triangulation

5.1 Tetrahedralization of a Point Set

Euler's formula is a fundamental corollary in homology theory and describes for three-dimensional complexes the relationship between the number of vertices n, the number of edges e, the number of facets f, and the number of solids s.

$$n - e + f - s = 0$$

The number of solids includes the infinite solid "outside" of the complex. In this sense all facets are boundaries between solids. For the example of a cube it becomes $(8-12+6-2=0)$. Splitting one facet of the cube into two triangles results in $(8-13+7-2=0)$. For a general tetrahedralization Euler's formula can be written as

$$n - e + t - T = 1$$

where t is the number of triangles and T the number of tetrahedra. Based on this equation simple bounds on the number of tetrahedra can be deduced

$$n - 3 \leq T \leq \begin{pmatrix} n-1 \\ 2 \end{pmatrix} - n_{hull} + 2$$

where n_{hull} is the number of points on the convex hull. A detailed combinatorial analysis and the investigation of various types of tetrahedralizations with different characteristics, worst case scenarios, as well as construction algorithms can be found in [40].

The three-dimensional Delaunay Triangulation is a special type of tetrahedralization[36]. In two dimensions a Delaunay Triangulation is known to minimize the largest circumcircle, to minimize the largest minimum-containment circle, and to maximize the minimum angle of all triangulations. The minimum-containment circle of a triangle is the smallest circle which contains the triangle. The circumcircle of an obtuse triangle is larger. The optimization of the angle seems to be due to the in two dimensions existing relation between edge length, circumcircle, and opposite angle (3.7). In three dimensions the Delaunay Triangulation is only known to minimize the largest minimum-containment sphere [15, 130]. An important

difference between two and three dimensions is the number of triangles/tetrahedra as a function of the number of points n. While the number of triangles in any triangulation grows with $O(n)$, the number of Delaunay tetrahedra in a tetrahedralization can grow with $O(n^2)$ [123].

5.2 Definition and Delaunay Properties

The definition of the Delaunay Triangulation is based on the Voronoi diagram through the principle of duality (Fig. 5.1). Voronoi diagrams and their application in an enormous amount of fields are described in detail with many original references in [117].

Definition 5.1 (Voronoi) *Let $P = \{p_1, \ldots, p_k\}$ be a finite set of points in the n-dimensional space R^n and their location vectors $\mathbf{x}_i \neq \mathbf{x}_j \; \forall \, i \neq j$. The region given by*

$$V(p_i) = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}_i\| \leq \|\mathbf{x} - \mathbf{x}_j\| \; \forall \, j \neq i\}$$

is called Voronoi region (Voronoi box) associated with p_i and

$$\mathcal{V}(P) = \bigcup_{i=1}^{k} V(p_i)$$

is said to be the Voronoi diagram of P.

A Voronoi box is formed through the intersection of planes and is therefore a general irregular polyhedron. The facets of the Voronoi boxes correspond in the dual graph to the Delaunay edges which connect the points of P.

Criterion 5.1 (Delaunay edge) *Let P be a finite set of points in a sub-domain Ω^n of the n-dimensional space R^n. Two points p_i and p_j are connected by a Delaunay edge e if and only if there exists a location $x \in \Omega^n$ which is equally close to p_i and p_j and closer to p_i, p_j than to any other $p_k \in P$. The location x is the center of an n-dimensional sphere which passes through the points p_i, p_j and which contains no other points p_k of P.*

$$\begin{aligned} e_{Delaunay}(p_i, p_j) \Longleftrightarrow \exists x & \\ x \in \Omega^n \quad & \wedge \\ \|x - p_i\| = \|x - p_j\| \quad & \wedge \\ \forall \, k \neq i, j \quad \|x - p_i\| < \|x - p_k\| & \end{aligned}$$

Combining this criterion for the three edges of a triangle and furthermore for the four triangles of a tetrahedron leads to the following criteria for Delaunay simplices. A Delaunay triangle is thereby the dual of a Voronoi edge.

Criterion 5.2 (Delaunay triangle) *Let P be a finite set of points in a sub-domain Ω^n of the n-dimensional space R^n. Three non-collinear points p_i, p_j and p_k form a Delaunay triangle t if and only if there exists a location $x \in \Omega^n$ which is equally close to p_i, p_j and*

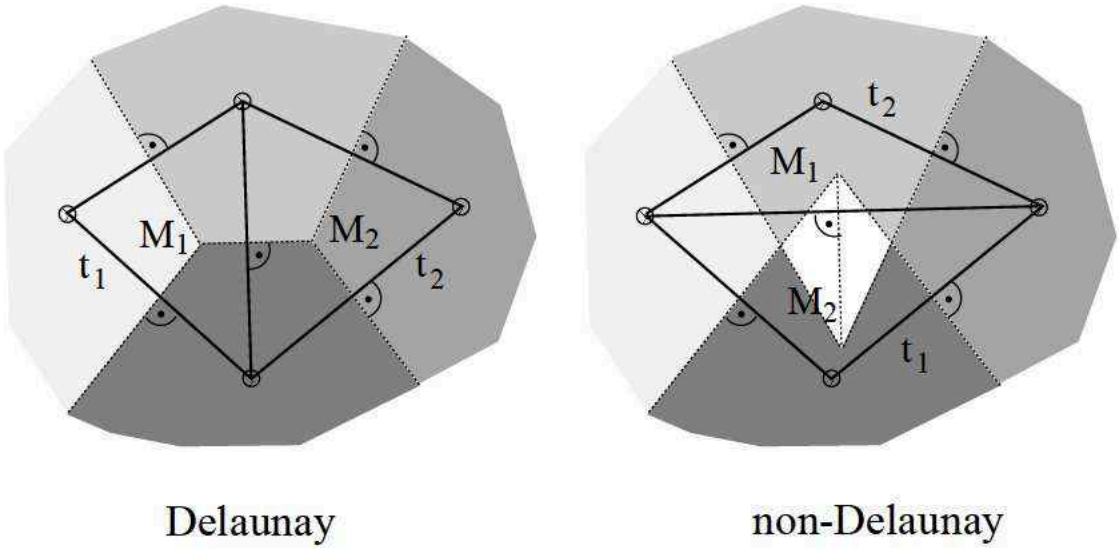


Figure 5.1: Each Voronoi box associated with a point is differently shaded. Two triangles t_1, t_2 with their circumcenters M_1, M_2 which are the vertices of the Voronoi boxes are depicted for the correct Delaunay case and for the non-Delaunay case. Incorrect Voronoi boxes which are derived from non-Delaunay triangles overlap.

p_k and closer to p_i, p_j, p_k than to any other $p_m \in P$. The location x is the center of an n-dimensional sphere which passes through the points p_i, p_j, p_k and which contains no other points p_m of P. For $n = 2$ only one such sphere exists which is the circumcircle of t.

$$
\begin{aligned}
t_{Delaunay}(p_i, p_j, p_k) \Longleftrightarrow \exists x \\
x \in \Omega^n \quad \wedge \\
\|x - p_i\| = \|x - p_j\| = \|x - p_k\| \quad \wedge \\
\forall\, m \neq i, j, k \quad \|x - p_i\| < \|x - p_m\|
\end{aligned}
$$

Crit. 5.2 implies that an empty circumcircle is necessary but not sufficient for Delaunay surface triangles in three dimensions. This is the reason why a two-dimensional Delaunay Triangulation code is of limited use to construct a three-dimensional Delaunay surface triangulation. The Delaunay edge and Delaunay triangle criteria are depicted in Fig. 5.2. A Delaunay tetrahedron corresponds to a point in the Voronoi diagram, which is the vertex of four[1] incident Voronoi boxes.

Criterion 5.3 (Delaunay tetrahedron) *Let P be a finite set of points in a sub-domain Ω^n of the n-dimensional space R^n, where $n \geq 3$. Four non-coplanar points p_i, p_j, p_k and p_l form a Delaunay tetrahedron T if and only if there exists a location $x \in \Omega^n$ which is equally close to p_i, p_j, p_k and p_l and closer to p_i, p_j, p_k, p_l than to any other $p_m \in P$. The location x is the center of an n-dimensional sphere which passes through the points p_i, p_j, p_k, p_l and*

[1] If in three dimensions more than four Voronoi boxes are incident at a Voronoi point, the Delaunay Triangulation is non-unique.

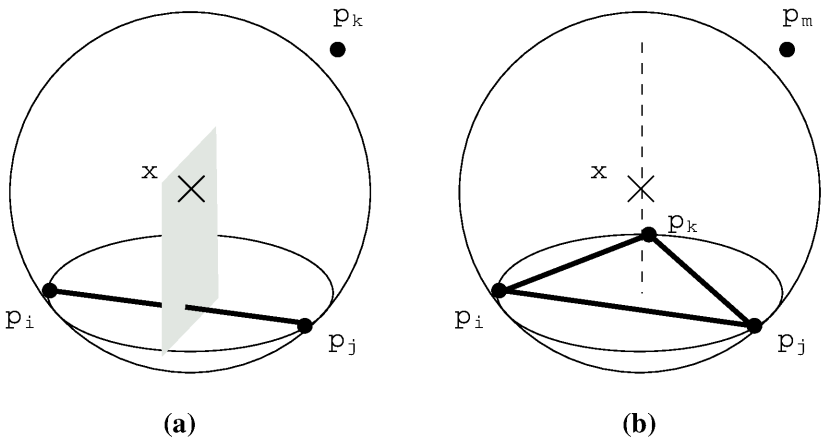


Figure 5.2: The Delaunay edge (a) and Delaunay triangle (b) criteria.

which contains no other points p_m of P. For $n = 3$ only one such sphere exists which is the circumsphere of T.

$$\begin{aligned} T_{Delaunay}(p_i, p_j, p_k, p_l) \iff \exists x \\ x \in \Omega^n \quad \wedge \\ \|x - p_i\| = \|x - p_j\| = \|x - p_k\| = \|x - p_l\| \quad \wedge \\ \forall\, m \neq i, j, k, l \quad \|x - p_i\| < \|x - p_m\| \end{aligned}$$

A Delaunay tetrahedron must consist of Delaunay edges and Delaunay triangles. The edge and triangle criteria are implicit, because the existence of the n-dimensional sphere in Crit. 5.1 and in Crit. 5.2 is guaranteed by the sphere in Crit. 5.3.

An *anisotropic Delaunay Triangulation* [20] can be defined through a simple linear transformation. The empty circumcircle criterion is applied in the transformed space and results in an empty ellipse in the mesh space. The transformation must be allowed to change over the domain to be practically useful. This leads to difficulties in the grading of the mesh. For an extremely inhomogeneous transformation the Delaunay Triangulation cannot be applied in a consistent way and the excellent property to always guarantee a valid tessellation without special consistency checks is lost.

5.3 Algorithms for Constructing a Delaunay Triangulation

This section briefly overviews Delaunay Triangulation algorithms for a given point set P without constraining boundaries. In two dimensions a naive edge swapping approach is less optimal, because the number of required flip operations grows with $O(n^2)$ where n is the number of points. Optimal algorithms run in $O(n \log n)$ time and depend on efficient data structures and point bucketing schemes. For randomized point sets in two dimensions algorithms with expected linear running time exist. In three dimensions the number of tetrahedra grows in the worst case quadratically with the number of points. In practice for

normal point sets $O(n \log n)$ algorithms are possible. It again depends on the implementation of special data structures, search trees, and sorting. A detailed comparison of Delaunay Triangulation algorithms can be found in [15, 47, 174]. The following paragraphs provide sketches of the algorithms with simplified explanations for the two-dimensional case. The three-dimensional versions are analogous. The divide-and-conquer and the sweepline method have been thoroughly tested in two dimensions. For a practical extension to three dimensions the incremental methods seem most suitable.

5.3.1 Divide-and-Conquer

The point set P is recursively divided into halves until the subsets contain a minimum number of points. These smallest sets of two or three points can be linked yielding edges or triangles. The following conquer step merges the subsets. Thereby the convex hull of a subset is traversed and linked to the other subset. Re-connecting the points by e.g. flip operations to satisfy the Delaunay criterion is equivalent to finding the dividing polygonal chain between the Voronoi diagrams of the two subsets. The dividing chain which consists of Voronoi edges can be constructed in linear time which results in an overall $O(n \log n)$ performance [123].

5.3.2 Sweepline

Sweepline methods form another general class of algorithms in computational geometry as do the divide-and-conquer techniques. For example in two dimensions a vertical line is swept from left to right. The sweepline is halted at so called event locations where the status of the sweepline is updated. Between events the sweepline does not have to be halted because its status does not change. In such a way the domain is partitioned into stripes which are sequentially processed. The status of the sweepline and the type of events depend on the application. For the construction of the Delaunay Triangulation such an algorithm has been implemented by [46]. The boundary edges of the current (incomplete) state of the mesh are stored in a tree data structure. An event occurs when the sweepline reaches a point of P or when it passes a circle formed by three adjacent vertices of the current mesh boundary. New elements are created and the status of the boundary edges is updated.

5.3.3 Incremental Construction

An initial triangulation which covers the domain is constructed. For example the bounding box is split into two triangles. The points of P are incrementally inserted into the triangulation. The triangle which contains the inserted point is first located and then split. Two variations to this algorithm exist. The topology around the inserted point can be updated by flip operations to restore the Delaunay property [84]. Alternatively, all triangles whose circumcircles contain the inserted point are removed and the resulting cavity is triangulated by linking the inserted point to *all* vertices of the cavity boundary. This simple linking scheme automatically guarantees the Delaunay property of the new elements. The technique is referred to as the Bowyer/Watson algorithm because it was simultaneously published [21, 186]. The cavity is *star-shaped* [123] because at least one location exists (the location of the newly

inserted point) from which straight line segments can be drawn to all vertices of the cavity without intersecting the cavity boundary.

5.3.4 Incremental Search

Starting with an initial Delaunay edge the Delaunay Triangulation is incrementally constructed by attaching triangles to the current boundary of the triangulation. The search for the correct point to be linked to the current edge so that the circumcircle of the resulting triangle contains no other points constitutes the main factor in the overall performance of the algorithm. From an initial edge the triangulation grows until the convex hull of the point set P is triangulated. It can be observed that the growing boundary of the incomplete triangulation forms an advancing front. Nevertheless, one must be aware of the differences between an advancing front meshing method as described in Section 4.4 and an advancing front style triangulation algorithm [104, 106] for a given point set P. The incremental search algorithm appears to originate from [63] and [179]. It forms the foundation of the modified advancing front algorithm for constraining boundaries which will be discussed in detail in Section 6.4.

5.3.5 Convex Hull

An n-dimensional Delaunay Triangulation can be deduced from a computation of the convex hull in $(n+1)$ dimensions. For example a two-dimensional Delaunay Triangulation of a point set P is equivalent to the projection of the convex hull of a three-dimensional point set $\tilde{P}$ which is derived from P through a lifting transformation [15, 114].

$$\begin{aligned} \tilde{x} &= x \\ \tilde{y} &= y \\ \tilde{z} &= x^2 + y^2 \end{aligned}$$

A detailed discussion of convex hull algorithms can be found in [123].

5.4 Non-Uniqueness

The definition of the Delaunay Triangulation for a given point set P usually results in a unique triangulation/tetrahedralization of P. No two different triangulations/tetrahedralizations exist for the same point set which both satisfy the Delaunay property. However, degenerate subsets of points in P can be formed which lead to non-uniqueness.

Definition 5.2 (cospherical point set) *Let P be a finite set of points in n-dimensional space R^n. At least $n+2$ points p_i are said to be cospherical if and only if they are located on the perimeter of an n-dimensional sphere S where S does not contain any other points in P.*

In such cases the length of a Voronoi edge or the area of a Voronoi facet is zero. Hence, the corresponding Delaunay edges and facets are missing in the dual graph, and non-simplicial polygons/polyhedra are formed. These can be arbitrarily triangulated/tetrahedralized, because their topology remains undefined by the Delaunay criterion.

In two dimensions such point sets are often called cocircular. However, one should be aware that cocircularity in three dimensions refers to a more specific degeneracy. It is useful to distinguish this special degenerate case which only occurs in dimensions higher than two.

Definition 5.3 (cocircular point set) *Let P be a finite set of points in three-dimensional space R^3. More than three coplanar points are said to be cocircular in R^3 if and only if they are located on the perimeter of a two-dimensional circle s in R^3 where s defines a disk which contains no other points in P.*

In fact a cocircular point set in three dimensions implies the existence of two intersecting cospherical point sets. This is true as long as at least one other point exists in each half space defined by the plane of the cocircular point set.

Degenerate cases require specially enhanced algorithms to ensure a robust Delaunay Triangulation engine. None of the above listed Delaunay algorithms remain entirely unaffected. For the incremental search based algorithm the consequences are discussed in detail in Section 6.4.3 and the implementation of a new and unconventional solution is presented. A more typical approach to avoid degenerate cases is to apply perturbations to the location of the points [81, 114, 115]. However, this technique is not as straightforward as it appears and other difficulties arise.

- It must be guaranteed that the perturbations destroy all cospherical subsets and that they do not create new cospherical subsets.
- It can be shown that the number of sliver elements increases greatly when cospherical point sets are perturbed. Additive noise does not affect a random point set by nature, but the quality of a tetrahedralization of an ortho-product point set is significantly reduced.
- If it is not permitted to dislocate certain points, e.g. vertices of the boundary, the temporary additive noise must be removed when the topology is set which is *after* the triangulation/tetrahedralization. This is not easy because the sliver elements which have been caused by the additive noise reach zero volume when the points are repositioned to their original location.

5.5 Boundary Integrity

Constraining boundaries are usually defined as a set I of vertices, edges, and facets which are *closed* under intersection. All intersections between entities in I must be present in I. The facets are bounded by edges of I. All vertices of facets and edges must be contained in I. In two dimensions such a set is called a *planar straight line graph* ($PSLG$) and in higher dimensions *piecewise linear complex* (PLC). A Delaunay Triangulation of the vertices of I will generally not be conform with the edges and facets of I (Fig. 5.3-a). Special means are required to incorporate I into the Delaunay Triangulation. Two different theoretical concepts exist which extend the definition of the Delaunay Triangulation for boundaries. Furthermore, two different approaches exist when to incorporate the boundaries from an algorithmic point of view.

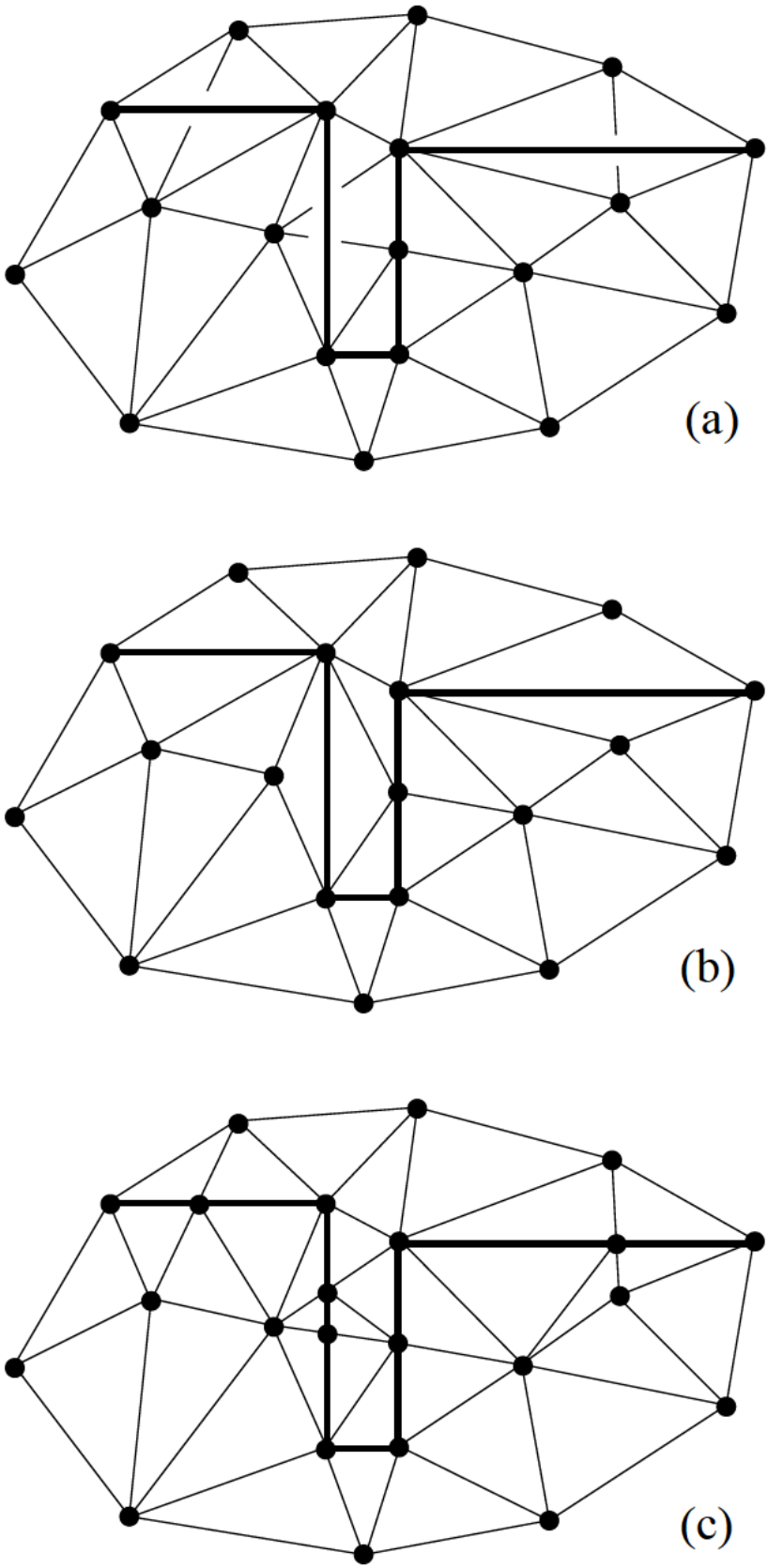


Figure 5.3: (a) Boundaries which are not conform with the Delaunay Triangulation (b) A constrained Delaunay Triangulation (c) A conforming Delaunay Triangulation

5.5.1 Constrained Delaunay Triangulation

A constrained Delaunay Triangulation (*CDT*) [155] is depicted in Fig. 5.3-b. The boundary edges are preserved and not split into smaller edges by avoiding the insertion of additional points. Their presence in the *CDT* is ensured through local modifications where Delaunay edges are removed or flipped. The resulting Delaunay Triangulation is said to be constrained by the boundary. Elements near the boundary are *not* guaranteed to satisfy the Delaunay criterion. Even though the smallest sphere criterion (Crit. 3.1) for boundary edges is stronger than the Delaunay criterion, it can paradoxically be of advantage for meshing applications to use *CDT*. The reason lies in the for meshing applications different perception of proximity as can be seen in Fig. 5.4. If one defines as medium for visibility the mesh elements, the point P in Fig. 5.4 is invisible from the edge e. Therefore, for the mesh element adjacent at edge e the point P can be ignored and the smallest sphere criterion is in this sense fulfilled, although the edge e is not even a Delaunay edge. If the boundary depicted in the figure were an interface between the meshes of two regions, the point P would certainly inflict the smallest sphere criterion of edge e, and the Delaunay criterion for the interface edges would not be sufficient but necessary.

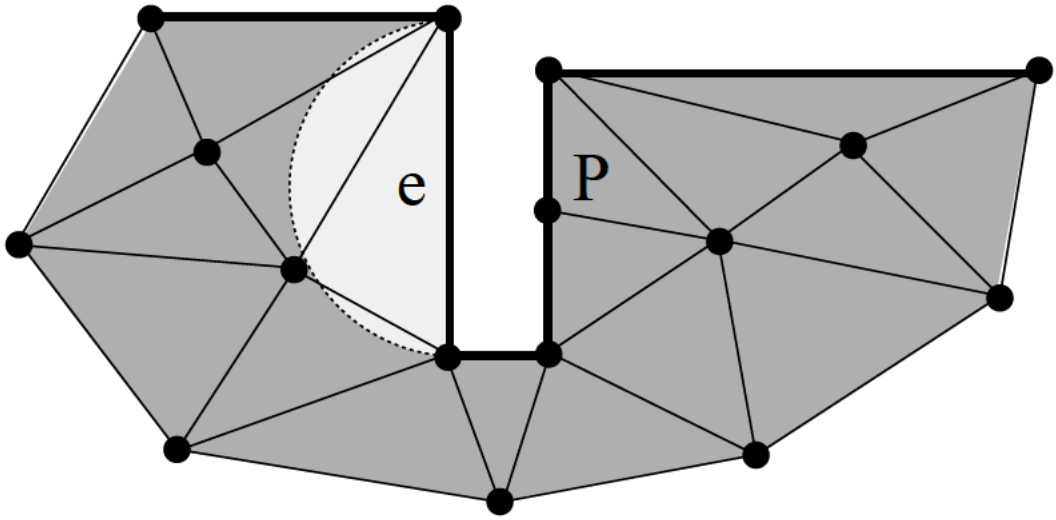


Figure 5.4: A constrained Delaunay Triangulation with a non-Delaunay edge e. The point P does not affect edge e. The half of the smallest sphere which lies inside the mesh is highlighted.

Contrary to the two-dimensional case a constrained Delaunay Triangulation does not exist a priori in three dimensions. Constellations of boundary facets exist which cannot be tetrahedralized without the insertion of additional points on the boundary. In other words no tetrahedralization can be found regardless of Delaunay criteria which allows the preservation of the boundary facets. An example of such a constellation is the *twisted prism* or *Schönhardt polyhedron* [149] which requires the insertion of at least one additional vertex (Fig. 5.5). No tetrahedron can be attached to the bottom facet with any of the top three vertices as fourth point without that it intersects one of the diagonal side edges. This leads to the important question how to determine whether or not additional vertices are required

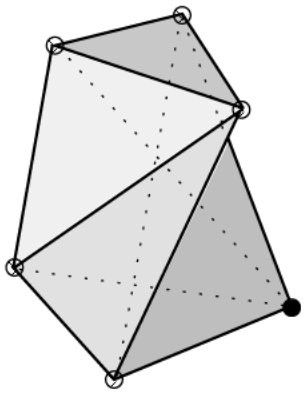


Figure 5.5: An untetrahedralizable twisted prism where the diagonals of the three side facets almost intersect.

to tessellate a general polyhedron. Unfortunately it has been shown that this problem is NP-complete[2] [14, 136, 137]. A condition for the existence of a higher dimensional *CDT* is examined in [163]. An important question is how to insert the additional vertices to guarantee a tessellation without further refinement of the boundary.

5.5.2 Conforming Delaunay Triangulation

A conforming Delaunay Triangulation (*RDT*) is shown in Fig. 5.3-c. The boundary edges are split into smaller edges by inserting additional points. The refinement of the boundary extends the initial set of vertices. The Delaunay Triangulation of this extended set of points is conform with the boundary edges. All elements satisfy the Delaunay criterion. A key question is how to insert those additional points to ensure that all boundary edges are contained in the Delaunay Triangulation and that the number of required points is minimal [59, 139]. It is especially in three dimensions a challenge to avoid overrefinement due to small boundary features. The insertion of points can induce further refinement in other areas and an endless feedback loop can evolve. The problem to guarantee a bound on refinement has not been solved for arbitrary three-dimensional inputs I and one must rely on heuristic techniques [162]. If the facets of I form dihedral angles of no less than 90° the complexity of the situation

[2]Many outstanding and famous problems exist in computer science for which no deterministic polynomial time algorithms are known. Exponential time algorithms are useless regardless of the speed of a computer, because finding a solution is in any case too expensive. Problems which can be solved by deterministic algorithms in polynomial time are said to be in P. Problems which can only be solved by nondeterministic algorithms in polynomial time are said to be in NP. In other words if the solution is not efficiently found but guessed and then checked for validity by a polynomial time algorithm the problem is in NP. The trouble is that no one has been able to prove that a problem is in NP and *not* in P. It is unclear whether an efficient deterministic polynomial time algorithm for a problem in NP remains undiscovered or simply does not exist. Problems exist which can be proven to represent all problems in NP. These problems are said to be NP-complete. If an NP-complete problem could be solved by a deterministic polynomial time algorithm, it would be proven that P=NP. This would mean that to all outstanding and famous problems efficient but undiscovered solutions exist. Otherwise one must rely on heuristics and hope that they do not fail for most practical instances of the problem. Some well known NP-complete problems are traveling-salesman, Hamilton-cycle, satisfiability, and longest-path.

alleviates [108, 164]. However, for small dihedral angles the heuristics remain to be optimized. A special refining scheme designed for sharp angles often exhibited by semiconductor devices is presented in Section 6.3. It should be noted that due to the lack of existence of a three-dimensional *CDT* the refinement of the boundary cannot be entirely avoided for any method which aims to integrate a boundary into a Delaunay Triangulation.

One can distinguish two approaches when to perform the refinement of the boundary.

Convex Hull and Segmentation This is the commonly used approach where a Delaunay Triangulation algorithm is first applied to the points of I. This results in a mesh which covers the convex hull. Afterwards follows a refinement to recover the missing edges and facets of I. At last a segmentation step is necessary to carve out those parts of the mesh of the convex hull which form the desired tessellation of the geometrical model.

Modified Advancing Front The boundary refinement precedes the tetrahedralization. It is guided by the Delaunay criterion which is applied to the edges and facets of I. The following step is the construction of the Delaunay Triangulation by an incremental search/advancing front style algorithm. The mesh is grown from the boundary as initial front only in regions where it is required (Section 6.4). A mesh which covers the convex hull and the necessary segmentation step is avoided.

Regarding the first approach the location of refinement points is often derived from intersections between the Delaunay Triangulation and the not yet recovered edges and facets. This is not always ideal and a refinement based on geometrical quality measures as for example discussed in the next section is of advantage. The triangulation of a refined edge of I is trivially unique. However, the triangulation of a refined facet of I can be non-unique which complicates the identification and recovery into a three-dimensional Delaunay Triangulation.

Although the second approach seems more obvious and possesses some advantages it is not common at all. This is probably due to three reasons. First, the refinement cannot be based on intersections of the boundary with the Delaunay Triangulation, because it is performed prior to the tetrahedralization. This is not a big disadvantage, because anyhow such a refinement proves to be not ideal for complex inputs in three dimensions as was mentioned above. Second, the advancing front style Delaunay algorithm is not common itself. Its performance depends heavily on the efficiency of the point location. Third, a robust implementation of the advancing front style Delaunay algorithm especially for degenerate cases under finite precision arithmetics is a quite difficult task. Chapter 6 addresses these issues. The developed boundary refinement scheme and the robust implementation of the modified advancing front algorithm combined with a fast point location will be discussed in detail.

5.6 Steiner Points and Steiner Triangulation

In the context of a Delaunay Triangulation and other optimal triangulations/tetrahedralizations *Steiner points* refer to points which are added to the set of vertices of the input *PSLG*/*PLC* I. The name does not indicate a specific location for the added point. While refinement is quite naturally considered for mesh generation purposes, the addition of Steiner

points to a Delaunay Triangulation is a powerful concept in computational geometry which allows quite theoretical investigations. It forms the basis for many provable optimal triangulation algorithms for various quality criteria [15, 16, 134].

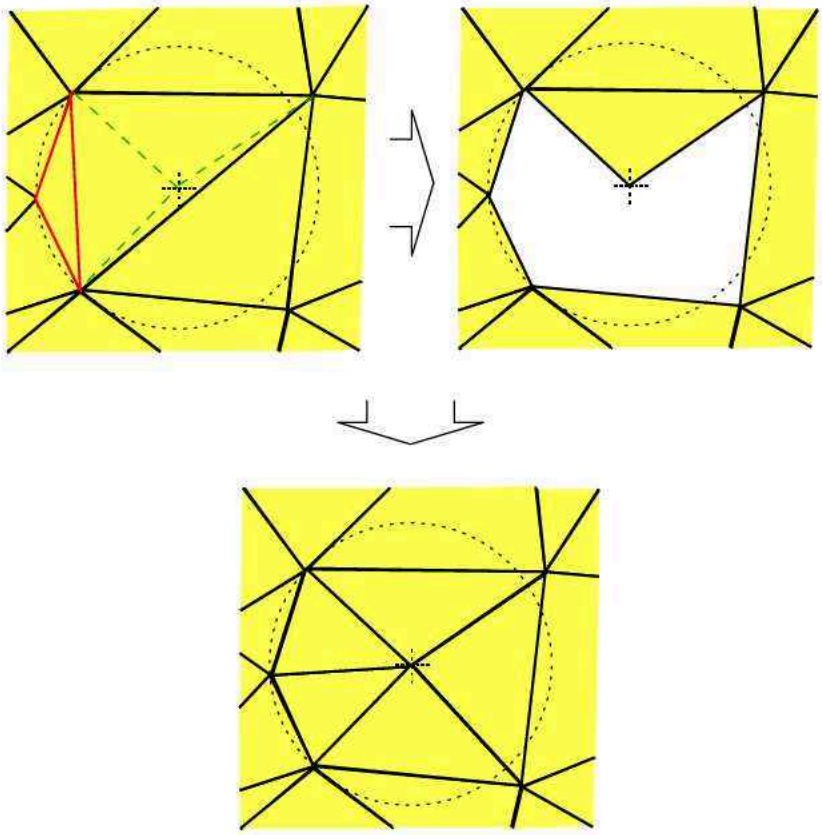


Figure 5.6: Steiner point insertion at the circumcenter, removal of non-Delaunay elements, and triangulation of the resulting cavity.

One of the most important techniques is the insertion of a Steiner point at the *circumcenter* of a badly shaped element. The element is not necessarily refined itself, because its circumcenter might be located in an adjacent element. In a subsequent step the Delaunay property is restored which ensures the destruction of the undesired element because its circumsphere is not empty. The minimum distance between two points cannot be reduced unproportionally. The circumsphere of a Delaunay element contains no other points, hence the inserted circumcenter cannot lie arbitrarily close to another point. This allows a proof of good grading. The refinement can be bounded by disallowing the insertion of circumcenters for circumspheres smaller than a given limit. Then, the insertion process must terminate, because it runs out of space.

For example in two dimensions a scheme can be devised to guarantee angles between 30° and 120° as long as the length of the boundary edges is within a specified range [28]. Figure 5.6 shows an obtuse triangle which is removed by inserting its circumcenter and restoration of the Delaunay property. An example triangulation is shown in Fig. 5.7. The key idea is

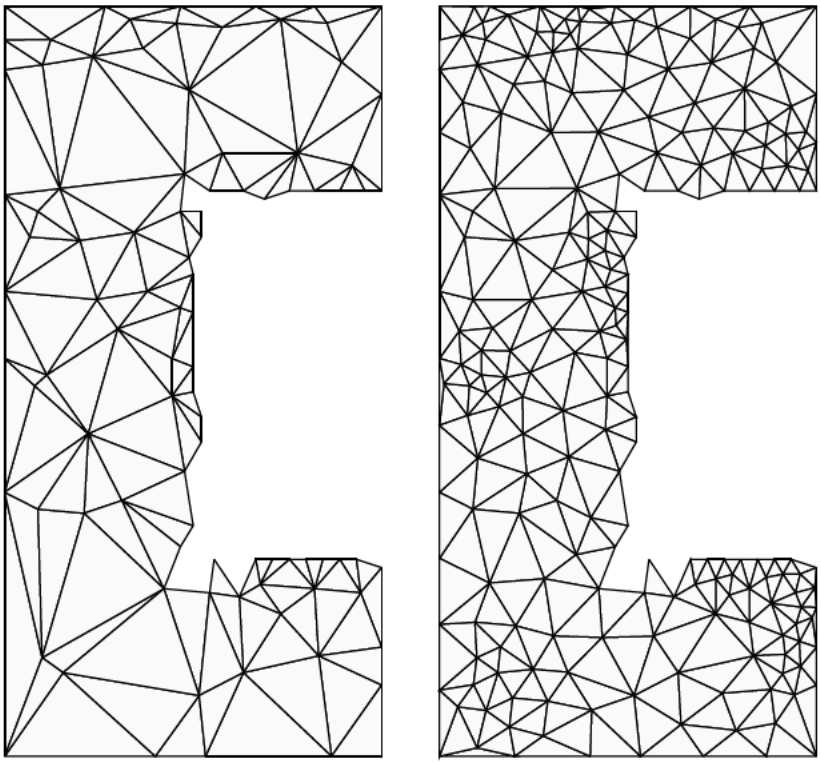


Figure 5.7: Delaunay Triangulation vs. quality improved Steiner Triangulation. The original 130 triangles (94 points) were refined with 128 Steiner points resulting in 376 triangles.

that a Steiner point at the circumcenter directly affects the shortest edge to circumradius ratio quality measure Q_1 (3.1). To see this it is assumed without loss of generality that the minimum distance between two points of the input I is b. Steiner points are inserted as long as elements exist with circumradius greater than b. The restoration of the Delaunay property after each point insertion is possible with flip operations. Hence, it can be guaranteed due to the Delaunay property that the inserted points are also not closer than b to any other point. Simultaneously it is guaranteed that no circumradius is greater than b. Termination is guaranteed because each point defines a disk with radius equal to the minimum distance b in which no other points may be located. All such disks sooner or later cover the entire available space and no more points can be inserted. The resulting edge length ranges between the minimum distance b and the maximum $2b$ which is possible for the maximum circumcircle with radius b. Hence, the worst (minimal) quality is $Q_1 = \frac{l_{\min}}{R} = \frac{b}{b} = 1$ (Fig. 5.8). Because of (3.7) the minimum angle is $\alpha = \arcsin \frac{b}{2b} = 30°$.

An analysis which includes the boundary is more complex [29, 135]. Certain restrictions have to be applied to the edges and facets of I. As already mentioned the edge lengths must be within a certain range or the edges must form angles of e.g. no less than 90°. Naturally, edges of the input I can form a sharp angle which is forced into the triangulation and which cannot be resolved. If a Steiner point lies outside of the domain defined by I, it cannot be inserted. Figure 5.9 shows an obtuse element of which the longest edge is a boundary edge

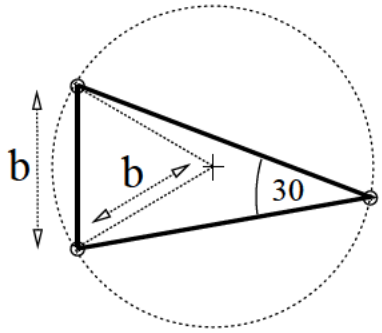


Figure 5.8: The worst case element with a 30° angle and minimum edge length b. The largest circumcircle has radius b.

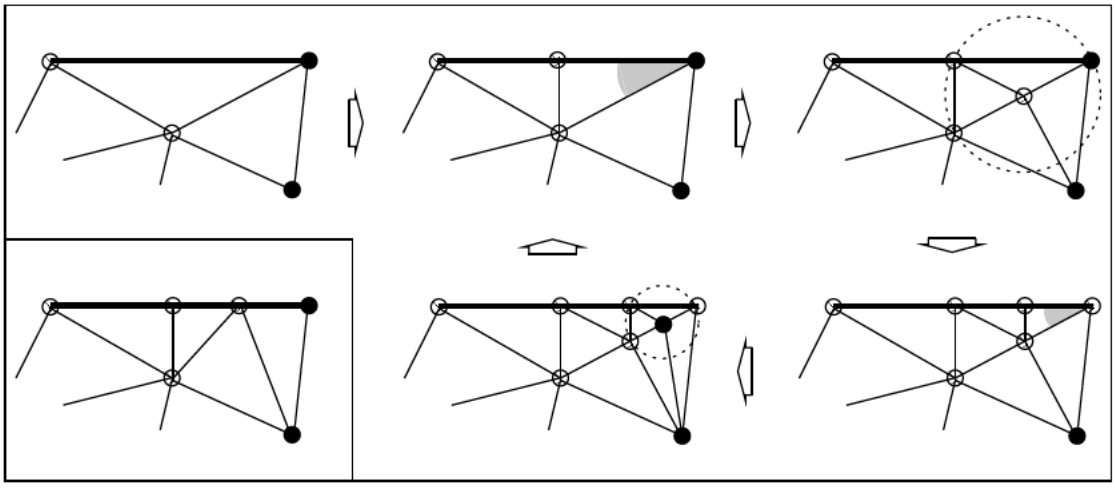


Figure 5.9: A naive approach where the bisection of boundary edges and the insertion of circumcenters runs into an endless loop. The small angle which causes the insertion of a Steiner point at the center of the dotted circumcircle is shaded. A better solution can be obtained and is shown in the bottom left corner.

and the circumcenter is outside. An intuitive approach to bisect the longest edge to destroy the obtuse angle combined with a minimum angle criterion enforced through Steiner point insertion at circumcenters may run into an endless loop (Fig. 5.9). Alternately the boundary edge is further bisected and a Steiner point added at the circumcenter of the new element. The boundary edge can never be flipped and the angle never improves. A solution is to check whether or not a Steiner point candidate inflicts the smallest sphere criterion of a boundary edge. As can be seen in Fig. 5.9 the inserted circumcenters are always located inside of the smallest circle passing through the two vertices of the boundary edge. In such cases the Steiner point candidate should be discarded and the boundary edge should be instead refined itself. Generally, a Steiner point insertion algorithm must stop the attempt to resolve bad angles in impossible situations due to special constellations of the edges and facets of I, rather then to cause excessive refinement. Restrictions on I are not acceptable for practical implementations.

A detailed comparison of Steiner Triangulation algorithms is given in [162]. Depending on how the boundary is handled different angle bounds can be achieved [28, 29, 135]. Often it is experienced that a theoretical minimum angle bound of e.g. 20° can be increased in practice up to 30°. A Steiner Triangulation with circumcenters is just as powerful in three dimensions to improve the shortest edge to circumradius quality criterion Q_1 [30, 37]. However, as was pointed out in Section 3.1 sliver elements are not captured by Q_1 and are not removed by inserting circumcenters as Steiner points. A three-dimensional method which promises to avoid all badly shaped elements and which is based on a two-dimensional approach with Steiner points [16] is proposed in [109].

5.7 Delaunay Slivers

A sliver has been introduced in Section 3.1 as an element with a very specific bad shape in the context of mesh generation. They are sometimes also called *flat tetrahedra*. Here, in the context of a Delaunay Triangulation and for the following chapter it is worthwhile to distinguish three sliver versions (Fig. 5.10).

- A sliver tetrahedron which does not satisfy the Delaunay criterion.
- A sliver tetrahedron which satisfies the Delaunay criterion in a strict sense. Its vertices are not part of a set of cospherical points and the Delaunay Triangulation around the sliver is unique.
- A sliver tetrahedron which satisfies the Delaunay criterion, but which is composed of vertices out of a set of cospherical points. The Delaunay Triangulation is not unique.

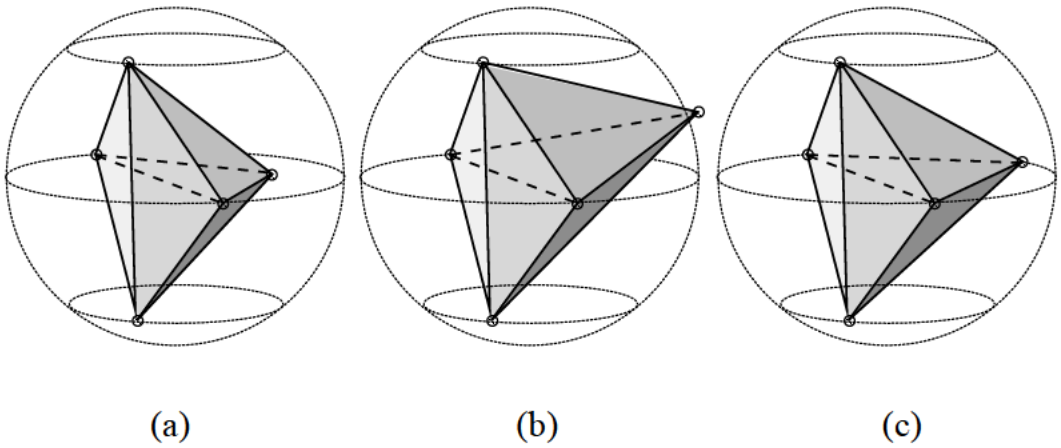


Figure 5.10: (a) Non-Delaunay sliver with circumsphere and two adjacent tetrahedra in the back (b) Strict sense Delaunay sliver with an empty circumsphere (c) Delaunay sliver with a cospherical point set.

The first type is of less importance and will clearly be absent in a Delaunay Triangulation. The second type, a Delaunay sliver, unfortunately exists in a Delaunay Triangulation for

a given point set P in three dimensions. The various optimality properties of a Delaunay Triangulation were given in Section 5.1. The minimum dihedral angle is not guaranteed to be optimal in three dimensions. A different non-Delaunay tetrahedralization might exist which avoids slivers with small dihedral angles but which is in other sense less optimal. In theory manipulation of P by constructing a Steiner Triangulation with Steiner points at circumcenters does not help to eliminate Delaunay slivers. The question arises whether a different type of Steiner point insertion is capable to manipulate P such that a Delaunay Triangulation results which does not contain Delaunay slivers. Alternatively, it would be of interest to examine how pronounced the occurrence of Delaunay slivers is in a practical mesh and how often they really survive the insertion of Steiner points at circumcenters in practice. Also, there is a chance that the third type sliver which is less critical exists much more often than a strict sense Delaunay sliver. Local transformations can be applied while maintaining the Delaunay property. The Delaunay Triangulation is not unique. The mesh examples in Section 3.2.3 have shown two different Delaunay Triangulations of an identical ortho-product point set where only one tetrahedralization contained sliver elements. A similar distinction of types can be made for the twisted prism. Its relation to slivers will be important in the next chapter (Section 6.4.3, Fig. 6.29).

Chapter 6

Architecture and Implementation

6.1 Meshing Strategy and Overall Concept

The implemented meshing strategy follows the concept of Delaunay methods as described in Chapter 4. The design consists of four parts respectively placing mesh points, preprocessing the surface, tetrahedralization, and further insertion of Steiner points for adaptation purposes and quality improvement (Fig. 6.1). No restrictions are imposed on the generated and/or provided initial mesh point distribution. Mesh points may overlap the structure boundary or cover the entire bounding box. Fully unstructured meshes can be constructed. The tetrahedralization engine does not require to remove or change mesh points. Among the advantages of such a logical separation in a place points and link [169] approach are stability, modularity, and a higher flexibility in combining different techniques for various applications. An example for the success of this approach in two dimensions is the triangulation engine described in [160].

The goal of surface preprocessing is the actual management of complex structure boundaries in an intelligent way, consistency checks, conversion issues, and the generation of a high quality Delaunay surface mesh. The minimum information defining a multi-segment geometry such as one unsorted list of general polygons is a sufficient input. In fact, only the non-convex regions of the structure boundaries have to be specified. It is not required to provide a closed surface. For example the surface of the top region of a semiconductor structure after etching and deposition simulation without the bulk boundary suffices to generate a volume mesh. Various discretization criteria for surface elements as discussed in Section 3.2 can be applied. The initial distribution of mesh points is taken into account by the surface preprocessor. Situations where unconnected points are too close to the boundary or collapse with the boundary can be cleaned up on the fly. The developed surface refinement algorithm ensures the integration of the boundary into the Delaunay mesh.

The tetrahedralization engine uses the modified advancing front Delaunay algorithm as briefly introduced in Section 5.5. The preprocessed surface description provides the initial front. The mesh generation process is constrained to the regions of interest by the Delaunay boundary representation. The tetrahedralization of the convex hull of the vertices and mesh points is not even temporarily necessary. It also means that initial mesh points which

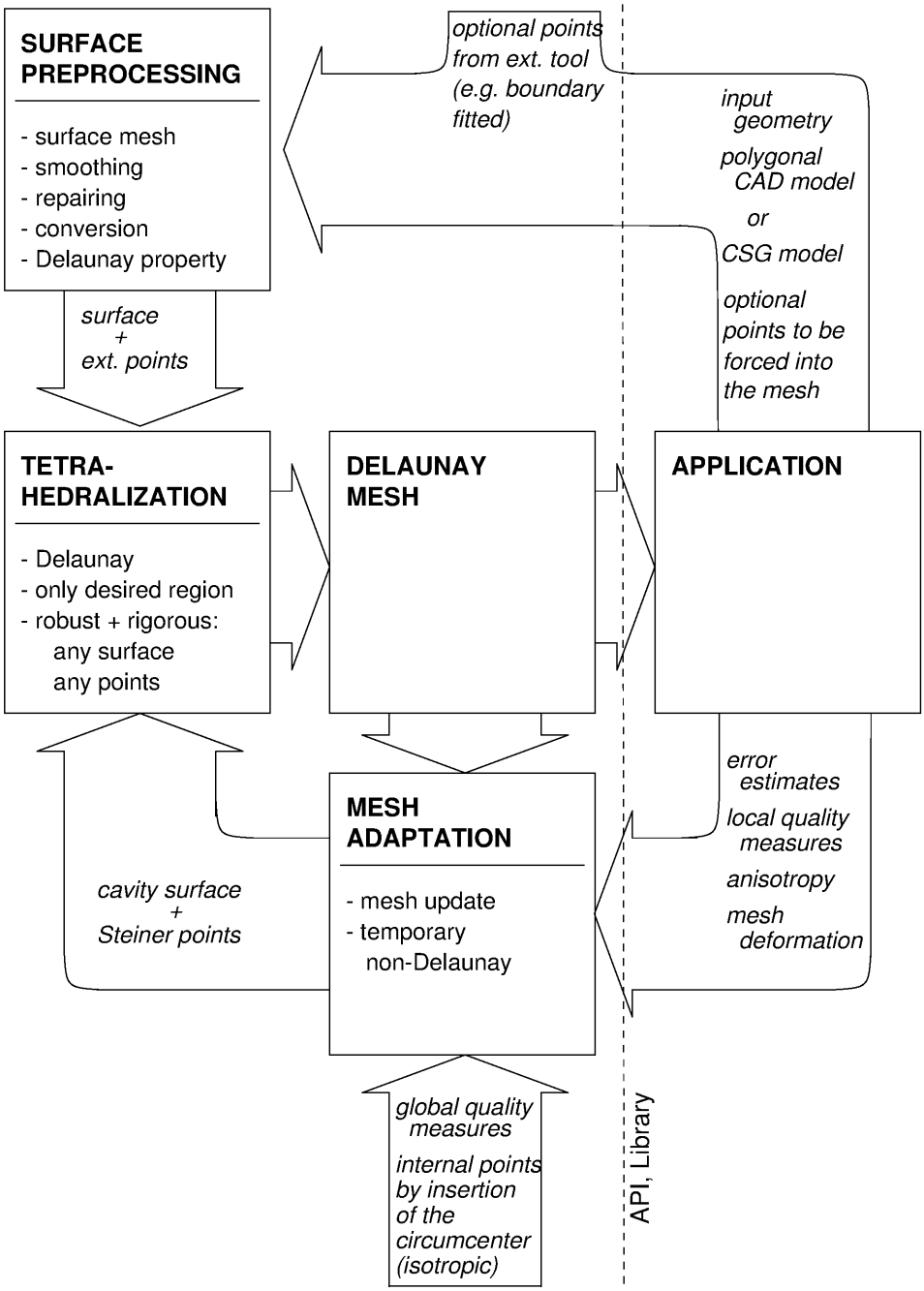


Figure 6.1: Overall concept

are located outside of structure boundaries never affect the meshing procedure. A robust implementation was developed for degenerate cases combined with an efficient non-uniform point bucketing scheme. Many existing variations of the advancing front style Delaunay algorithm are either two-dimensional or experience difficulties with cospherical point sets [31, 33, 39, 42, 43, 122]. A recent three-dimensional implementation uses a perturbation approach to handle degenerate cases [81].

Other advantages of the modified advancing front algorithm are an easier parallelization [32] and the possible real time visualization of the meshing process for debugging purposes. The process can be interrupted at any time and the resulting snapshot is a valid mesh of part of the domain. Those parts of the domain which are not of interest are never meshed which saves computation time especially for extremely non-convex domains. The approach is therefore also well suited for local mesh adaptation. The cavities in the mesh where elements have been discarded due to local modifications form local fronts which can directly be used as advancing fronts to grow tetrahedra and to remesh the gaps. In such a way Steiner points can be inserted to improve geometrical quality measures.

6.2 Initial Point Generation

Many different styles to generate an initial mesh point distribution can be conceived. Corresponding algorithms are more easy to design, because of the lack of restrictions applied to the point set.

- The octree-type generation can be extended to use special point templates for each octree leaf instead of using corner points exclusively. The edges of the octree leafs as shown in Fig. 6.2 can be bisected, or more complex patterns similar to crystal lattices can be utilized.
- An advancing front technique to place the mesh points can produce boundary-fitted meshes.
- Random point distributions to fit inhomogeneous density requirements can be additionally smoothened.

Note that the generation of a Steiner point distribution as described in Section 5.6 does not require an initial point set and is based on a boundary mesh. A detailed investigation of finite octree point generation can be found in [192]. A test example of the finite octree and the resulting mesh is shown in Fig. 6.2 and Fig. 6.3.

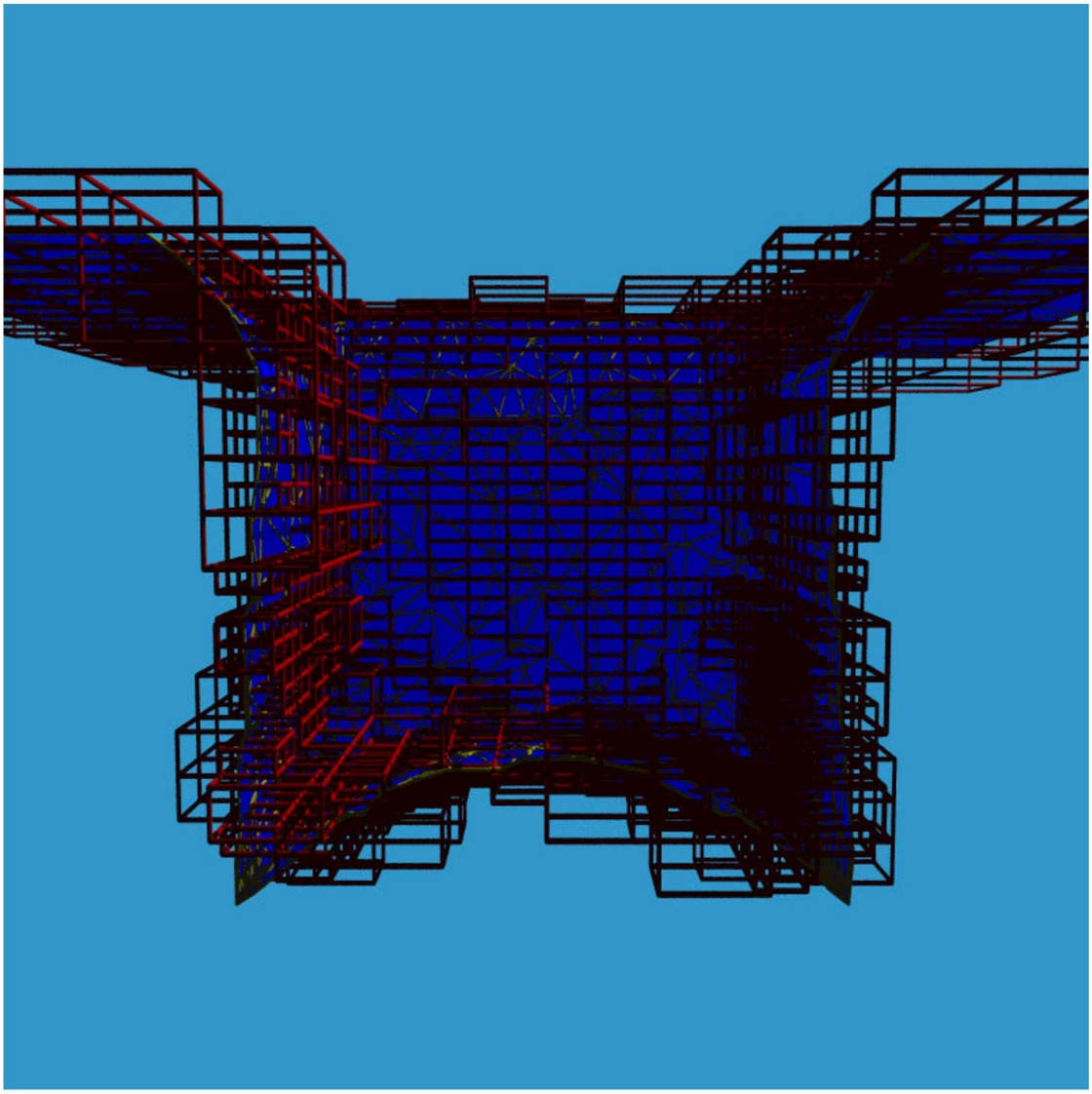


Figure 6.2: Finite octree point generation.

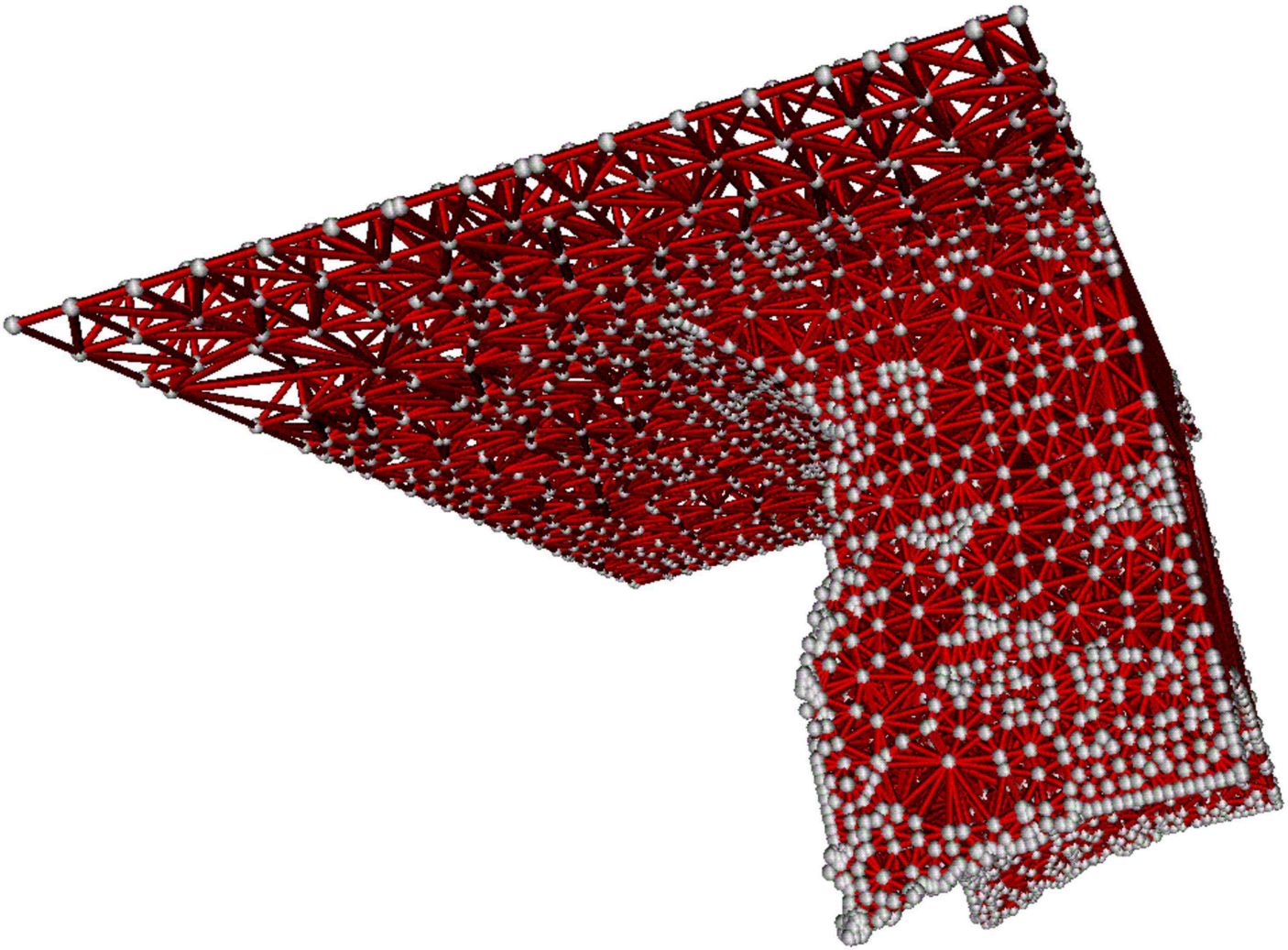


Figure 6.3: Mesh for the octree point distribution, 25253 tetrahedra.

6.3 Surface Triangulation

Given a set of constraining input facets I (polygonal representation, Fig. 6.11) a surface triangulation is derived. All polygons are triangulated by a recursive splitting algorithm. The polygons may be non-convex, more than two polygons may share an edge, and they do not have to form a closed surface. Depending on the choice of mesh and discretization a refinement algorithm is further applied to either enforce the smallest sphere or the Delaunay criteria (Crit. 5.1, Crit. 5.2, Crit. 3.1, and Crit. 3.2). An important operation is defined to reduce the number of refinement points.

Definition 6.1 (flip-able triangle) *Two triangles t_1, t_2 which are in conjunction convex and which share one common edge e_c are said to be flipped if they are replaced by two triangles $\tilde{t}_1, \tilde{t}_2$ forming the same outline but sharing a different edge $\tilde{e}_c \neq e_c$. This operation is extended to non-planar triangles: Let t_1, t_2 be two triangles sharing a common edge e_c which is not part of any other triangle $t_i, i \neq 1, 2$. If t_1, t_2 form a dihedral angle $\alpha = 180° + \epsilon$ where $|\epsilon| \leq \epsilon_{acc}$ and ϵ_{acc} is a user controlled accuracy, and if the equatorial sphere of t_1 contains a vertex of t_2 which does not belong to t_1, the triangle t_1 may be flipped as well and will be called flip-able triangle.*

The term *flip saturation* is introduced. A triangle may not satisfy the smallest sphere criterion and may not be flip-able. The triangle may become flip-able after a certain number of flip operations are applied on other triangles (Fig. 6.4). This situation is said to be *unsaturated*. It is desired to reach the state of flip saturation before unnecessary refinement occurs. This is achieved by employing the recursive triangle flip.

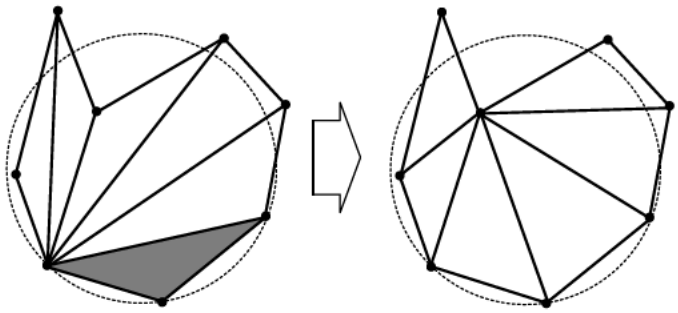


Figure 6.4: A triangle which is at first not flip-able and the state of flip saturation.

Recursive Triangle Flip

Two triangles t_1, t_2 are flipped. The resulting triangles $\tilde{t}_1$ and $\tilde{t}_2$ are each checked if they are flip-able. If any of the two triangles $\tilde{t}_1, \tilde{t}_2$ is flip-able with a triangle t_i where $i \notin 1, 2$ it will be flipped as well. Repeat for the flipped triangle $\tilde{t}_i$ until no further flip operations are possible.

This operation is performed once on each triangle during a linear scan over all existing triangles. The procedure during which no refinement is allowed is only necessary at the start to ensure flip saturation. Afterwards, during refinement the flip saturation is maintained after insertion of a point by applying the recursive flip locally on the affected triangle only.

It follows a definition of structural and flip-able edges.

Definition 6.2 (flip-able and structural edges) *If two triangles t_1, t_2 sharing an edge e_c form a dihedral angle $\alpha = 180° + \gamma$ where $|\gamma| > \epsilon_{acc}$ and ϵ_{acc} is a user controlled accuracy, the edge e_c is called a structural edge or S-edge. If $|\gamma| \leq \epsilon_{acc}$ the edge e_c is called flip-able edge. If a triangle t possesses an edge e_{open} with no existent adjacent triangle, the edge e_{open} will be called a structural edge as well. If a triangle t possesses an edge e_{multi} with more than one existent adjacent triangles, the edge e_{multi} will be called a structural edge as well.*

An example of an edge shared by more than two triangles is shown in Fig. 6.5. Edges with

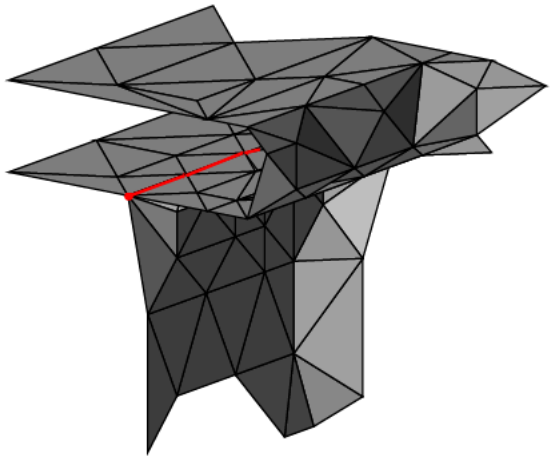


Figure 6.5: Multiple connected edges.

only one adjacent facet are structural and a potential inconsistency. They may indicate an undesired gap in the surface representation. The current implementation allows to repair such faults by tracing these edges and linking them to construct a three-dimensional ring. If this ring consists of a large number of edges, it can be assumed that the boundary does not form a closed surface intentionally and no action will be taken. On the other hand if the ring is relatively small, it can be assumed that a hole in the surface has been detected. The hole can be patched by a non-planar triangulation of the ring. This is performed by the general polygon triangulation algorithm which was used for the facets of the input I.

Note that two triangles need not be flip-able if they share a flip-able edge (Fig. 6.6).

All *S-edges* are detected and stored in a special data structure for further processing. The smallest sphere criterion will be enforced for *S-edges* by a novel refinement scheme presented in this section (Fig. 6.7). The idea is to avoid overrefinement due to a feedback situation where an inserted point causes further point insertions for other *S-edges*. This is especially important when incident *S-edges* span angles of less than 90°. The *S-edges* are drawn in bold,

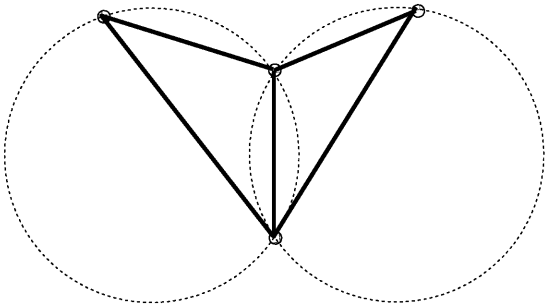


Figure 6.6: Non-convex coplanar triangles. The common edge is by definition flip-able, while the triangles are not.

the dashed circles illustrate the minimal edge length up to which refinement is permitted, and the solid circles symbolize sphere tests. There are two possible types how a refinement point is derived from points which inflict the sphere criterion. Such points will be called *disturbing* points.

Type P **Insertion** An orthogonal projection of the disturbing point onto the *S-edge* is used.

Type R **Insertion** The disturbing point is rotated onto the *S-edge*. The rotation axis passes through the point at which the two *S-edges* are incident.

If more than one disturbing point exists, a best candidate is selected by comparing the relative distance and choosing the closest. The simple cases are *P I and R I*. In *P II, R II, and R IV* the minimal edge length is limited and the inserted point receives an offset. If the ratio between the length of the two *S-edges* is close to one, the most complex situation case *R III* evolves. An offset as in case *R IV* would not produce a good result and the sphere test depicted by the left solid circle in case *R IV* could fail due to the inserted point. An overall improved situation results from an intended first order feedback where actually two points will be inserted in two consecutive steps *R III* + *R I*. The circle for the sphere test which causes the insertion in the first place is omitted in all sketches. Once the location of the new point has been determined and prior to its insertion, sphere tests are performed for some of the new edges to avoid feedback (solid circles). Note that the algorithm is designed for three dimensions in spite of the two-dimensional sketches. Figure 6.8 shows the result for the trivial case of a planar polygon. The *S-edges* (outline) satisfy the smallest sphere criterion after refinement.

The triangles are processed after the *S-edges*. If desirable, the refinement can be reduced by omitting the smallest sphere criterion for triangles (Crit. 3.1). The weaker Delaunay criterion (Crit. 5.2) cannot be checked easily, because the number of spheres to test can theoretically be infinite as the criterion requires a sphere of any size to be empty. Hence, the insphere test would have to be repeated for different sizes until an empty sphere has been found. An equivalent criterion was devised which needs at most two insphere tests. It uses a metric λ which will be described later on in Section 6.4.

Criterion 6.1 (double sphere) *Let P be a finite set of points in three-dimensional space R^3 and let t be a boundary triangle with a non-empty smallest sphere $S_{\min}$. The point $p_{k,\lambda_{\min}}$*

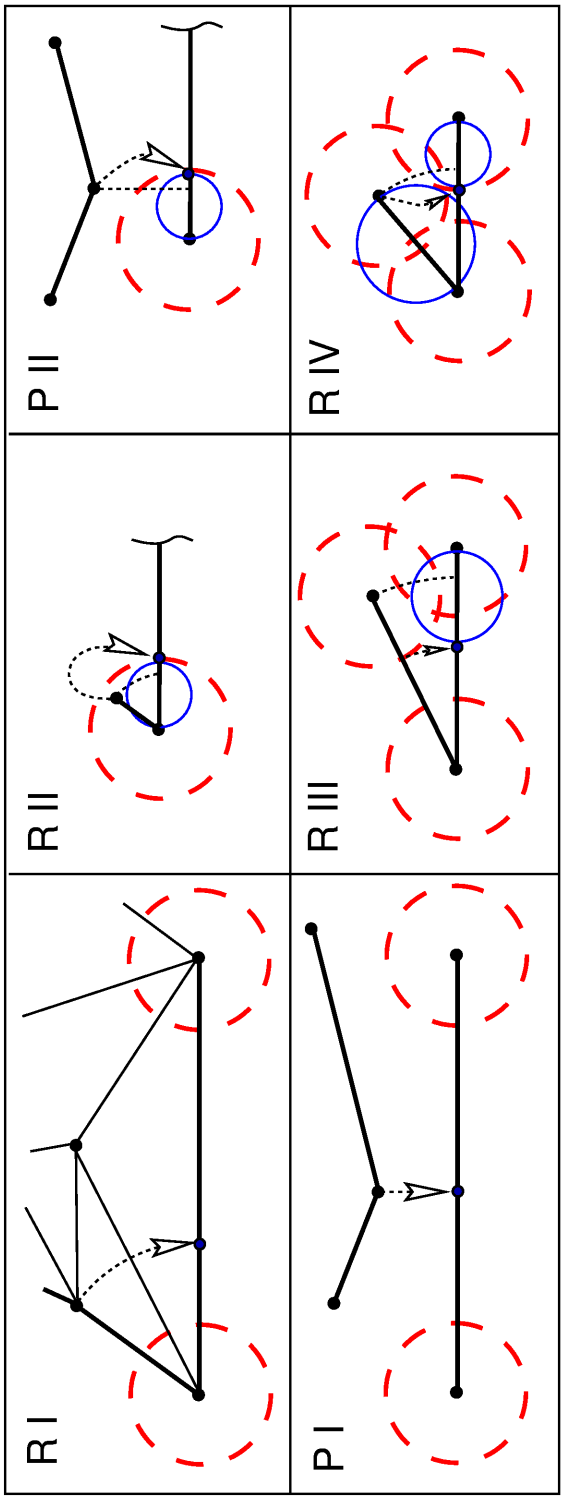


Figure 6.7: Refinement types for structural edges.

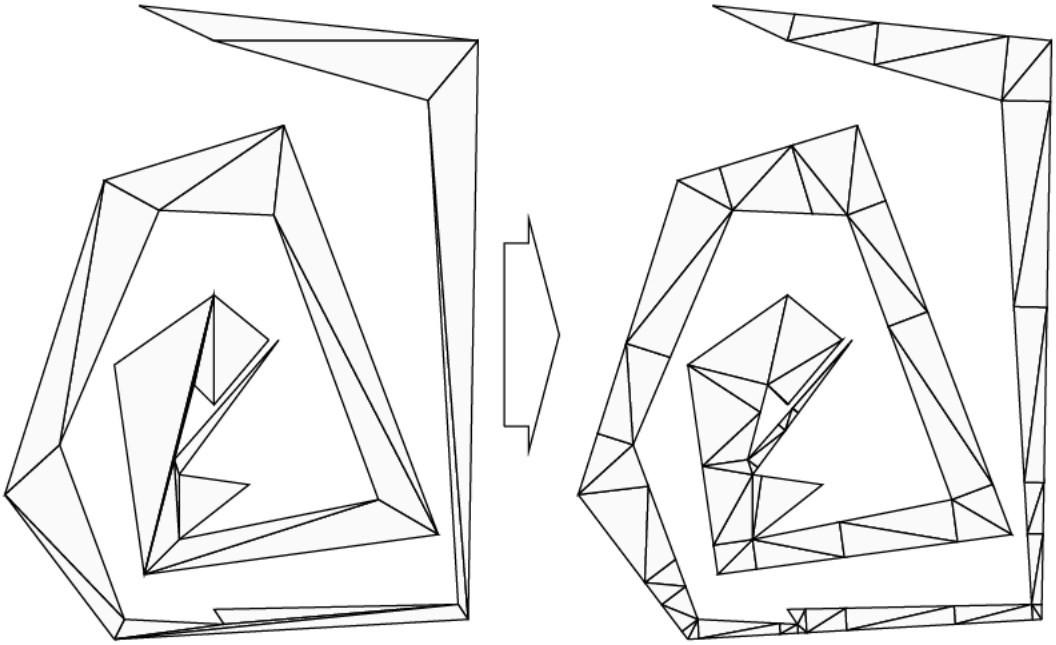


Figure 6.8: Refining structural edges for the trivial case of a planar polygon.

is contained in $S_{\min}$ *and minimizes a metric* $\lambda = H\vec{M}_k \cdot \vec{n}$. *The triangle* t *is a Delaunay triangle if and only if the sphere* S_{double} *defined by* t *and* $p_{k,\lambda_{\min}}$ *contains no other point in* P.

A non-Delaunay triangle and a Delaunay triangle with two disturbing points located inside of the smallest sphere and the empty double sphere are depicted in Fig. 6.9.

A key idea for Steiner point refinement with provable bounds was discussed in Section 5.6. The circumcenter of a triangle is a very well suited location for a new point to be inserted. The triangle which will actually be split can be different from the triangle which causes the refinement. Hence, a triangle search by e.g. a jump and walk algorithm is required. An algorithm was implemented which extends this concept for non-planar surfaces (Fig. 6.10). The location of the new point S is derived from the circumcenter M by an orthogonal projection. S is the point M projected onto the surface ($S = M_{proj}$). It is checked if the refinement point S really violates the sphere criterion of the bad triangle. The point S might fall outside of the sphere in which case a Steiner point refinement would not be justified, because the bad triangle would survive the Delaunay update step. A point location is necessary to find the triangle containing S for a given M. The direction of projection is orthogonal to the plane spanned by the bad triangle. The triangle search can be performed by a walk algorithm.

Locate Triangle

Starting with a triangle $t = t_{bad}$ and a given point P a triangle is searched for which contains P_{proj}. (Usually P is the circumcenter M.) The direction of projection is defined by the normal vector $\vec{n}_{bad}$ of t_{bad}. An edge e of t and $\vec{n}_{bad}$ define a plane which separates two half spaces. The edge e is said to *face* the point P, if P and t do not lie

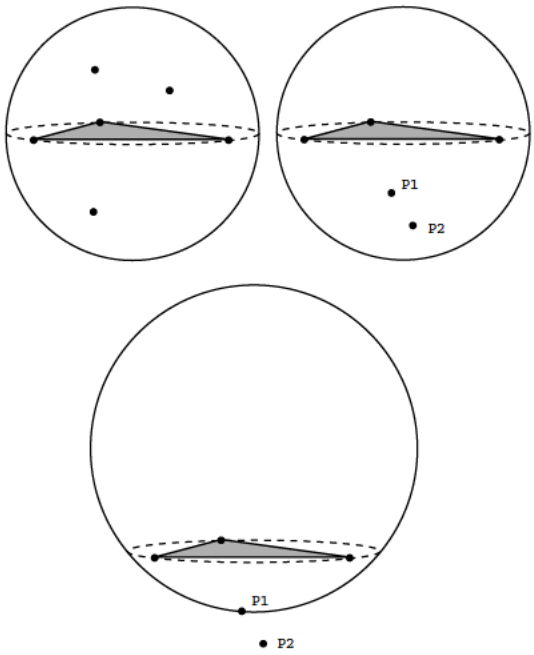


Figure 6.9: Double sphere criterion.

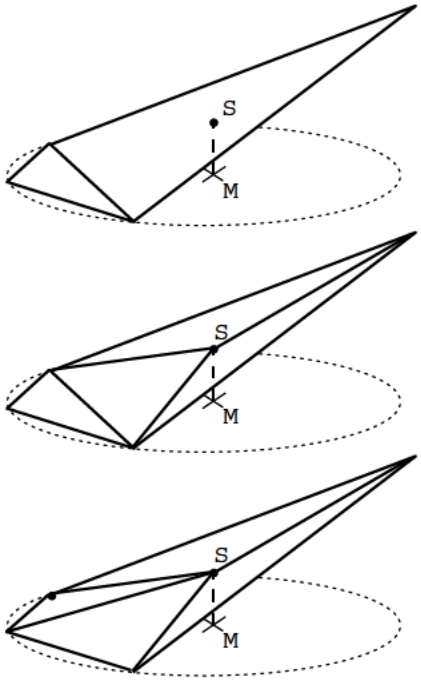


Figure 6.10: Locating the triangle which contains the projection of the circumcenter and flipping of the non-structural edge.

in the same half space. Determine one edge e of t which faces P and which is a flip-able edge (Def. 6.2). Follow edge e by finding the adjacent triangle t_{adj}. Repeat for $t = t_{adj}$ until an *S-edge* has been encountered or until the triangle t possesses no edge e which faces P. In either case the search is terminates. In the latter case t contains P_{proj}. The Steiner point $S = P_{proj}$ on t is calculated by intersecting the found triangle t with the line defined by P and the direction of projection.

The point P was used in the description to show that the algorithm can search for arbitrary points where $P \neq M$. This is important when Steiner points are derived from disturbing points instead of circumcenters.

If the triangle is not extremely obtuse, M will be contained in the triangle itself or in an adjacent triangle. Then, no iterations or only a few will be required. If an *S-edge* is encountered the triangle will not be refined. Instead the *S-edge* will be refined as described above. This is important for several reasons. Insertion of S is usually followed by triangle flip operations which eliminate the bad triangle. Hence, the edge to follow must be flip-able. Also, it is of limited sense to project M onto the surface for regions with sharp dihedral angles.

Originally, this type of refinement should only be applied to Delaunay triangles with an empty circumsphere to improve quality angle criteria. If the circumsphere is not empty, points could be inserted too close to each other. In the present case this is not necessary. In fact the Steiner point refinement of *non-Delaunay* triangles proves successful for enforcing

the Delaunay criterion as long as the state of flip saturation is guaranteed and maintained after each point insertion.

At the same time it is possible to define minimum angle and maximum area criteria upon which Delaunay surface triangles are adapted by the insertion of Steiner points. These can be applied inhomogeneously according to a control function. For the example in Fig. 6.11 the structural edges are shown in Fig. 6.12, the Delaunay surface triangulation in Fig. 6.13, and the adapted surface mesh in Fig. 6.14.

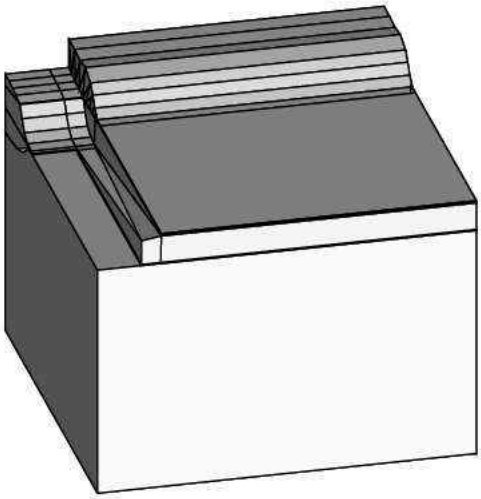


Figure 6.11: Polygonal boundary description of a MOS Transistor with a spacer.

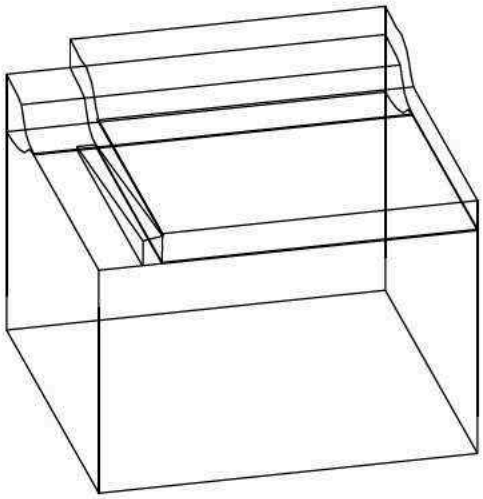


Figure 6.12: Structural edges of the MOS transistor example.

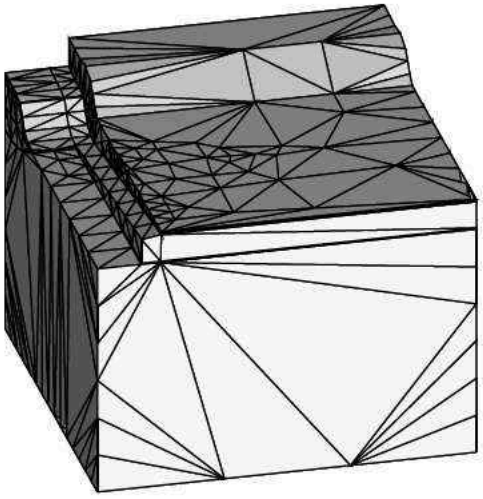


Figure 6.13: Delaunay surface triangulation.

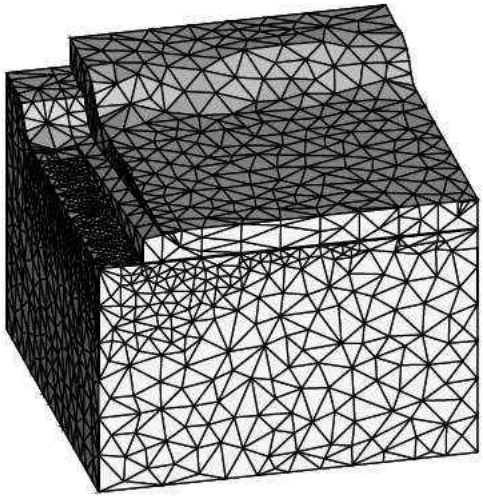


Figure 6.14: Adapted surface mesh after Steiner point insertions.

6.4 Volume Tetrahedralization

6.4.1 Algorithm Overview

Figure 6.15 shows the flow diagram of the modified advancing front algorithm. Delaunay triangles are extracted from the surface triangulation to form an oriented initial front. These triangles have the purpose of a seed which is inserted into a queue to "grow" tetrahedra. The

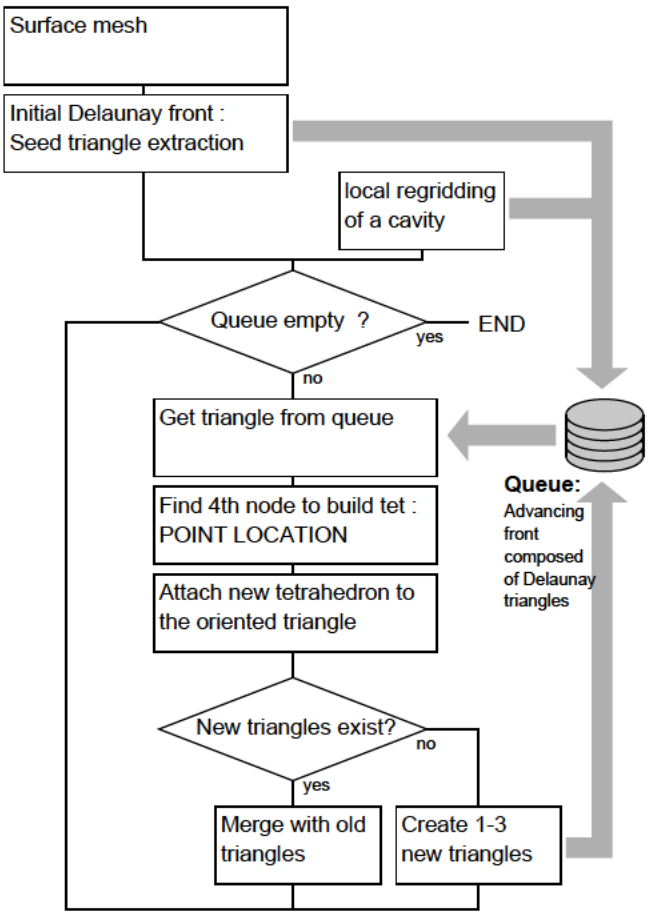


Figure 6.15: Modified advancing front algorithm.

algorithm starts with a non-empty queue. It does not require the queue to hold all surface triangles which represent the boundary. One triangle per enclosed segment is sufficient. The surface triangulation has two purposes.

1. It provides the initial front for the advancing front algorithm to start with. One triangle per segment is enough and is called a seed triangle.
2. It provides a border for the advancing front algorithm which cannot be passed.

The triangles of the initial front and all later generated triangles of the advancing front have an orientation defined by the order of their vertices. Given a seed triangle (which is taken

from the queue) a tetrahedron is attached to that side of the triangle which faces the half space in which its normal vector is directed. The tetrahedron is constructed with a a fourth point which has a positive distance to the seed triangle relative to its normal vector. In this way a *front side* and a *back side* of triangles is distinguished. Repeatedly attaching tetrahedra

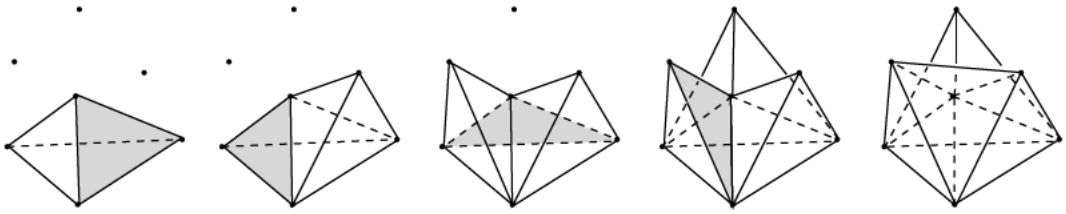


Figure 6.16: The triangle to which the next tetrahedron is attached is shaded for each step.

to the front sides of the triangles of the queue, removing them from the queue when they have been processed, and inserting new triangles into the queue leads to a growth process of tetrahedra (Fig. 6.16, Fig. 6.17). Hence, the queued triangles form the advancing front at all times. It advances when a new tetrahedron (attached to a triangle which is removed from the queue) generates new triangles which are inserted into the queue. Generally, a created tetrahedron can produce any number between 0 and 3 new triangles. During the beginning of the tetrahedralization process each created tetrahedron will more likely produce 3 new triangles and the size of the queue will increase rapidly. Later on the advancing front will merge with itself or parts of the surface triangulation and fewer new triangles are produced. A tetrahedron consists of n new triangles, $(3 - n)$ previously generated triangles, and the triangle to which it is attached. The $(3 - n)$ previously generated triangles must have been previously inserted into the queue or belong to the initial surface triangulation. They are part of the advancing front. When they are encountered during the creation of a new tetrahedron, they are removed from the queue and the advancing front is stopped. If n is zero the creation of the tetrahedron results in a decrease of the size of the queue. When the queue has run empty the meshing process terminates.

The following questions arise:

1. What degree of freedom does the algorithm have to choose a tetrahedron to be attached to a triangle of the queue ? What kind of tetrahedron will be chosen ?

2. How is it guaranteed that the advancing front does not pass through itself or the given boundary triangulation ?

3. How is it guaranteed that no part of the domain remains untetrahedralized ?

First it is assumed that the Delaunay Triangulation is unique and degenerate point sets do not exist. Given a fixed mesh point distribution and an oriented triangle only one point can complement the triangle to form a valid Delaunay tetrahedron to be attached to the triangle's front side. The meshing algorithm will choose to create exactly this tetrahedron. Intrinsically it is not able to create non-Delaunay tetrahedra as will be seen later on.

To answer the second question the following is observed.

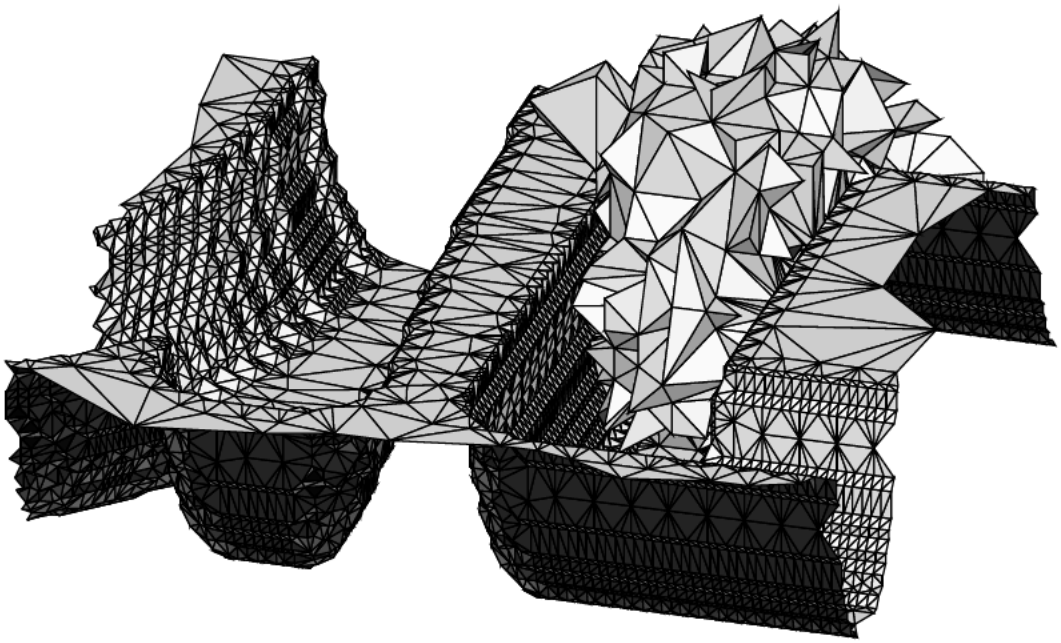


Figure 6.17: A snapshot of the growing mesh generated by the modified advancing front algorithm.

Lemma 6.1 *The advancing front cannot build loops, intersect or overlap with itself, or pass through Delaunay borders as long as it is guaranteed that identical triangles are never stored more than once.*

Proof 6.1 *Under the assumption of a unique Delaunay Triangulation any Delaunay triangle must be present within the tetrahedralization. No Delaunay triangle exists which is not part of the tetrahedralization. If one or more such triangles would exist the Delaunay Triangulation would not be unique. Given the facts that the algorithm only produces Delaunay tetrahedra and the advancing front separates the tetrahedralized domain from the untetrahedralized domain, it follows that at all times the advancing front consists entirely of Delaunay triangles. Assume that one part of the advancing front passes through another part of the advancing front. Then, part of the front must be located in the interior of a tetrahedralized region. Since the front consists of Delaunay triangles they must be part of the unique tetrahedralization. Hence, during the tetrahedralization Delaunay triangles must have been produced and stored where identical copies already existed as part of the front.* ■

Such a test whether or not an identical triangle already exists is trivial and does not cost performance. It guarantees that the algorithm terminates. Since no Delaunay triangles are generated more than once and the overall number of Delaunay triangles t in a Delaunay Triangulation is finite, the algorithm terminates after at most t steps.

To see that complex structures are rigorously meshed regardless of whether or not they are multiple connected the following is examined.

Lemma 6.2 *No volume is left untetrahedralized unless it is enclosed by a surface triangulation to which no seed was provided.*

Proof 6.2 *Assume that part of the domain remains untetrahedralized and the algorithm has terminated. The algorithm must only terminate if the queue holding the advancing front is empty. The untetrahedralized region must be separated from the tetrahedralized region or the outside of the domain by a surface triangulation T_{sep}. First, assume that T_{sep} contains at least one triangle that is part of a tetrahedron and was generated during the tetrahedralization process. Then, this triangle must have been inserted into the queue. It can be attached to at most one tetrahedron, because T_{sep} was defined to be the border between the tetrahedralized and the untetrahedralized region. Since it would have only been removed from the queue if a second tetrahedron would have been attached to it, the queue cannot be empty and the algorithm cannot have terminated. Second, assume that T_{sep} does not contain triangles produced during the tetrahedralization. Then, it must entirely consist of triangles present in the given boundary surface triangulation. Since it is a closed region, the boundary defines a segment. Under the requirement that at least one boundary triangle per segment is inserted into the queue the algorithm cannot have terminated.* ■

6.4.2 Point Location

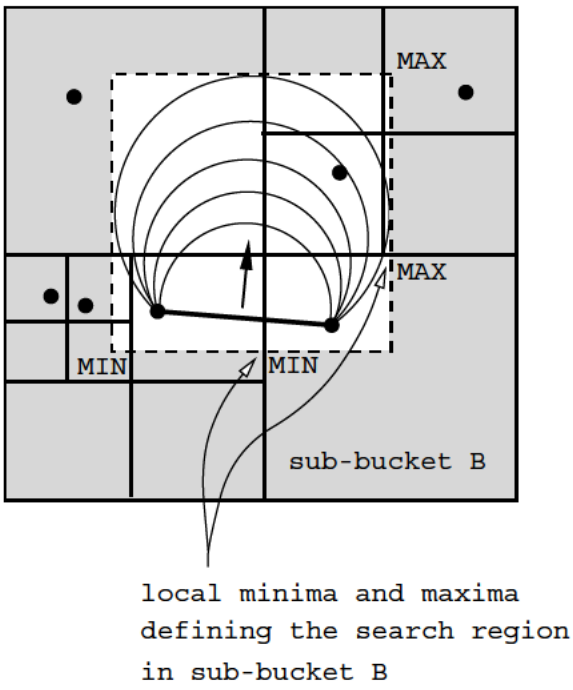


Figure 6.18: A non-uniform point bucketing scheme and a rectangular search region which is aligned with the bounding box and which is associated with a circle and a given λ value (two-dimensional analogy).

An efficient data structure for point bucketing is crucial to quickly locate the fourth vertex among all mesh points which completes a triangle from the queue to form a Delaunay tetrahedron. The currently processed triangle which was taken from the queue is called a *base triangle*. The advancing front holds Delaunay triangles, hence any base triangle satisfies

the Delaunay criterion. A local region around the base triangle needs to be searched for to find the correct fourth point. The border of this region is defined by a sphere containing the circumcircle of the base triangle. The radius of the sphere will be increased as long as the sphere does not contain other points. A sketch for the two-dimensional case is depicted in Fig. 6.18 where an edge corresponds to the base triangle. It is evident that one sphere must exist which does not hold any other points due to the Delaunay property of the base triangle. In fact, the way the region around the base triangle is examined resembles closely the Delaunay sphere criterion. The size of the sphere is increased so that exactly one other point (than the three points of the base triangle) is located on the perimeter of the sphere. This point is the seeked fourth vertex of the tetrahedron. It is evident that the tetrahedron must satisfy the Delaunay criterion, because its circumsphere does not contain any other point. For one base triangle two spheres exist that correspond to a tetrahedron on each side. The algorithm will only attach a tetrahedron to the front side of the base triangle as previously explained. It will therefore increase the size of the sphere into the direction of the normal vector of the base triangle. This is expressed by a signed increase of the normal distance λ between the center M of the sphere and the base triangle. The normal distance can be positive or negative. The point H denotes the circumcenter of the base triangle and $\vec{n}$ its unit length normal vector. The smallest sphere is characterized by $\lambda = 0$. A negative λ denotes a sphere which covers a larger area on the back side than on the front side of the triangle.

$$\lambda := \vec{HM} \cdot \vec{n}$$

The spherical search region fits into a cubic search region which can always be aligned with the bounding box (Fig. 6.18). This allows for an efficient non-uniform point bucketing scheme to locate points within the search region. All points are stored and maintained at all times in a point bucket octree [142]. Any rectangular area aligned with the bounding box can be searched by traversing the tree data structure.

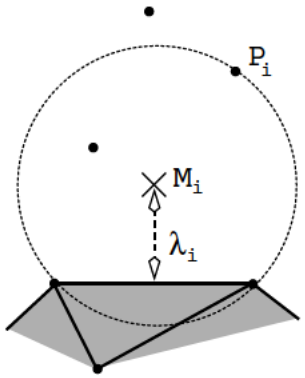


Figure 6.19: Every found point P_i in the search region defines a λ_i.

The gradual increase of the size of the search region does not cost much performance. If the sphere is too small no points will be located inside the search region and the effort of traversing the octree is minimal. The case when exactly one point (the fourth vertex of the tetrahedron) is found, is ideal. Usually, the search region will contain several points. In such a case the size of the sphere defining the search region will not be reduced until only a single

point is contained. Instead λ_i is computed from M_i which is the center of the sphere defined by the point P_i and the base triangle (Fig. 6.19). $\vec{X}$ is the location vector of one of the three vertices of the base triangle.

$$\lambda_i = (\vec{M}_i - \vec{X}) \cdot \vec{n}$$

From the collection of points P_i the correct fourth point can be chosen by minimizing λ_i.

Criterion 6.2 (Minimum λ) *Let P_i with corresponding λ_i be a complete set of all points which are located in a spherical search region in R^3 defined by a Delaunay triangle t and a value $\lambda_{\max} \geq \lambda_i$. The point P_j and the Delaunay triangle t form a Delaunay tetrahedron, if and only if $\lambda_j = \lambda_{\min}$.*

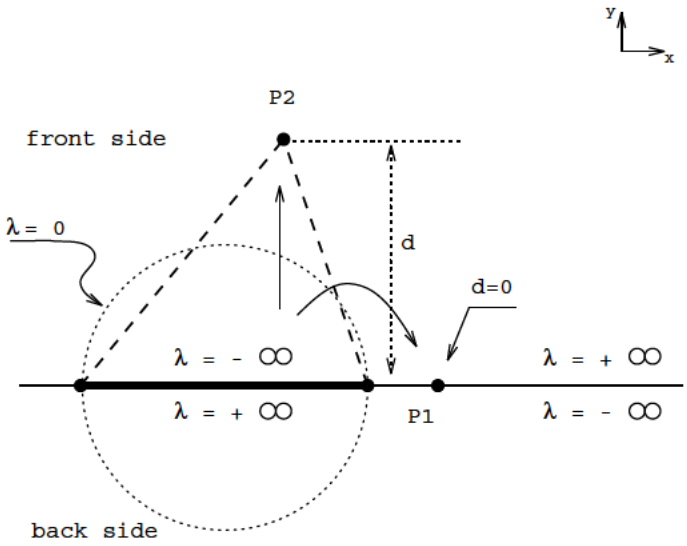


Figure 6.20: The scope of λ for the two-dimensional case. Depending on the location of a point its λ value can reach a critical value. Singular regions of λ are indicated.

The scope of the possible λ values is numerically a delicate matter. The robust implementation applies an additional insphere test. Figure 6.20 shows the two-dimensional analogy. The normal distance of a point P to the plane containing the base triangle (in two dimensions an edge) is d. The point P_1 has $d = 0$ and the point P_2 has a positive d. The algorithm has to determine which of the two points P_1 and P_2 is the correct point. The correct triangle in the two-dimensional case is drawn with dashed lines. The various possible locations of a point are denoted with the corresponding λ values. The location of points with $\lambda = 0$ is shown by the dashed circle. The point P_1 is on a numerical critical location:

$$\lim_{d \to 0^+} \lambda_{P_1} = +\infty$$
$$\lim_{d \to 0^-} \lambda_{P_1} = -\infty$$

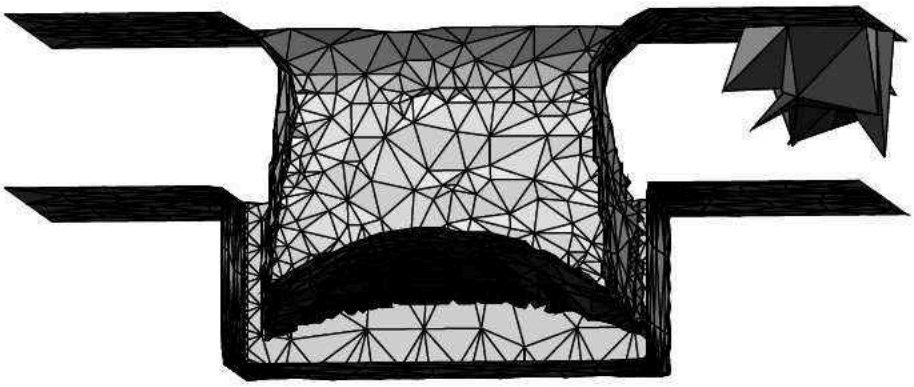


Figure 6.21: An open surface description and the growing mesh.

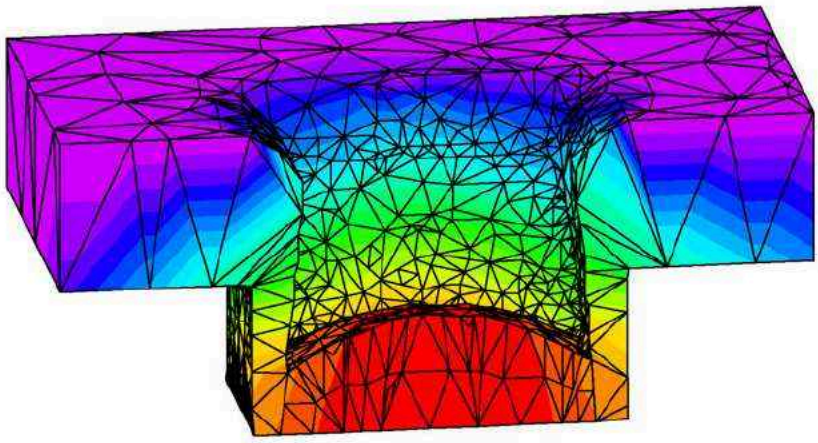


Figure 6.22: Complete boundary representation after tetrahedralization.

A more flexible algorithm allows boundaries which do not form a closed surface. This generalization is implemented as follows. If no fourth vertex can be found after gradually increasing the search region up to the size of the bounding box, the base triangle will be converted and stored as a boundary triangle. In this way new boundary triangles can be generated during the tetrahedralization process at locally convex regions of the mesh point distribution. Figure 6.21 shows an example of an open surface description and a snapshot of the meshing process. Figure 6.22 depicts the complete boundary including the generated boundary triangles after tetrahedralization.

The octree search and the λ criterion introduce the logarithm into the overall time complexity $O(n \log n)$ where n is the number of tetrahedra. For randomized point distributions the number of tetrahedra grows approximately linear with the number of points (Fig. 6.23). The algorithm runs in optimal time. The performance is given in Tab. 6.1.

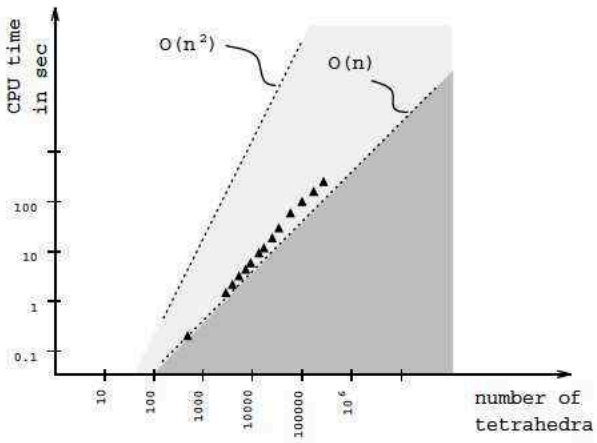


Figure 6.23: Runtime on an HP 9000-735/100Mhz

6.4.3 Degenerate Point Sets

For a practical tetrahedralization engine any restriction on the location of mesh points is unacceptable. The assumption of a unique Delaunay Triangulation is now removed. A scheme is devised how to extend the modified advancing front algorithm for degenerate cases as defined in Section 5.4.

Cospherical Point Sets

Under exact arithmetics degenerate cases result in a compound of points $P_{c,i}$ with identical λ_c. The fourth vertex cannot anymore be determined by minimizing λ (Crit. 6.2). Given a set of points $P_{c,i}$ with λ_c the original algorithm results in overlapping tetrahedra and crossed edges. Several two-dimensional situations are depicted in Fig. 6.24. Figure 6.25 shows an error prone situation with a three-dimensional cospherical point set. A valid tessellation cuts the domain in such pieces that the union of all pieces yields exactly the domain. Theoretically, minimizing λ suffices to produce a valid tessellation if no such point sets $P_{c,i}$ exist. Practically, this is not even the case for a *unique* Delaunay Triangulation due to finite precision arithmetics. The strict adherence to the λ criterion may *force* the generation of overlapping elements due to numerical errors [176]. The problem of finite precision arithmetics and of degenerate point sets are closely related.

If one wants to avoid the costly implementation of adaptive precision algorithms and floating point filters [11, 48, 54, 161], one cannot rely on a solution for exactly cospherical points $P_{c,i}$, but has to detect and treat *approximately* cospherical points $\tilde{P}_{c,i}$ with corresponding $\lambda_{c,i}$ from an algorithmic point of view.

At first must be a robust detection of $\tilde{P}_{c,i}$ which is a crucial task and cannot be achieved by a simple epsilon region where $\lambda_{c,i} \in [\lambda_{\min}, \lambda_{\min} + \varepsilon]$. Inconsistencies due to points that lie close to the border of the epsilon region would be inherent. Numerically they might appear at one time inside of the epsilon region and at another time outside of it. Therefore, an

quantity of ...			CPU time
points	tetrahedra	triangles	(in sec)
103	535	1098	0.2
503	3016	6112	1.4
703	4323	8739	2.1
1003	6268	12635	3.3
1503	9494	19107	5.1
2003	12713	25557	6.8
2503	16076	32300	8.8
3003	19394	38967	11.1
4003	25969	52140	15.6
5003	32691	65605	19.7
10003	65927	132160	46.5
20003	132854	266133	92.0
30003	199613	399756	145.0
40003	266899	534405	207.0

Table 6.1: Runtime on an HP 9000-735/100Mhz

adaptive epsilon region has been implemented which involves a sorting algorithm.

$$\begin{aligned} \lambda_{c,i} &\in \{\lambda_1, \lambda_2, \ldots, \lambda_i, \ldots, \lambda_n\} \quad \wedge \\ \lambda_1 &\equiv \lambda_{\min} \quad \wedge \\ \lambda_i &\leq \lambda_{i+1} \quad \wedge \\ \lambda_{i+1} - \lambda_i &\leq \varepsilon \quad \wedge \\ \lambda_{n+1} - \lambda_n &> \varepsilon \end{aligned}$$

Another critical situation can be observed in Fig. 6.26. The indicated point belongs to an epsilon region, but it somehow cannot be considered to belong to the set of cospherical points. An algorithm to deal with approximately cospherical points $\tilde{P}_{c,i}$ must be prepared for such cases as well.

The solution is to treat the points $\tilde{P}_{c,i}$ as an isolated tetrahedralization sub-problem forming a single step in the overall modified advancing front algorithm. Thereby the tetrahedralization technique for the sub-problem is performed by a local instance of the modified advancing front algorithm with a specific alteration described in the following paragraphs. A local instance of the algorithm means that a second queue is built to hold the local advancing front of the tetrahedralization of the sub-problem. An ortho-product point distribution consists of a large number of cospherical point subsets which are regularly stacked. The global advancing front for such an example is shown in Fig. 6.27. One can clearly observe how each subset is treated in a single step from a global point of view.

The local second queue can only provide the frame for the entire solution to the problem of cospherical point sets. Another specific alteration of the tetrahedralization algorithm is required. The enhancement is at first explained for the ideal case of exact cospherical points $P_{c,i}$. The local boundary of the point set is convex (Fig. 6.24-d, convex *LSPB*). The special tetrahedralization of this sphere is constructed in the following way. Consider a tetrahedralization with a convex boundary. A tetrahedron can be created by linking a point

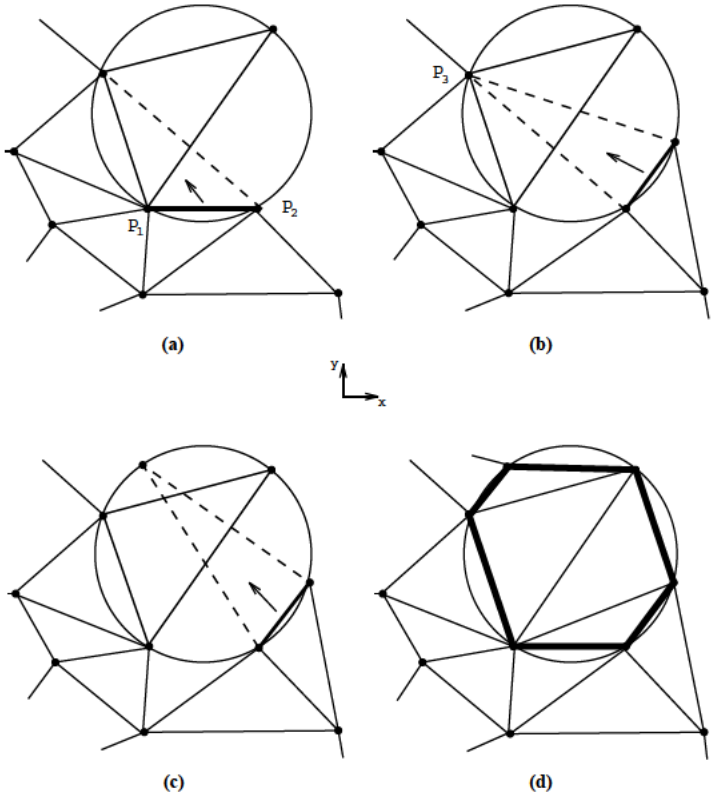


Figure 6.24: (a) Overlapping triangles which share two points (b) Only one common point (c) No points are shared (d) The convex local sphere boundary, *LSPB*

located anywhere outside the tetrahedralization to any triangle of the boundary that has a positive minimum distance to that point. Regardless which triangle of the boundary (with positive minimum distance) is taken the created tetrahedron cannot penetrate the existing tetrahedralization. If this single point is not only linked to one triangle, but to all triangles of the convex boundary that have positive minimum distance to the point, the result is again a tetrahedralization with a convex boundary. The tetrahedralization can be expanded point by point while keeping the temporary boundary convex. The gain from such a method can be immediately seen in Fig. 6.25. If the boundary would have been kept convex during the generation of the in the figure depicted tetrahedron, the error prone situation would have been avoided. The creation of a new tetrahedron could not lead to penetrations.

If the points are located on the perimeter of a sphere, the order in which the points are added to the tetrahedralization will not matter. Any point will be outside of the tetrahedralization of the other points. Also, any tetrahedralization of such points will be a Delaunay Triangulation. Thus, the convex *LSPB* can be tetrahedralized in this way with a slightly altered advancing front algorithm.

Starting with a base triangle a fourth point from the set of cospherical points is chosen to generate a tetrahedron. The first tetrahedron of the sphere forms a temporary convex boundary to which the other points of the sphere are added one after another. The temporary

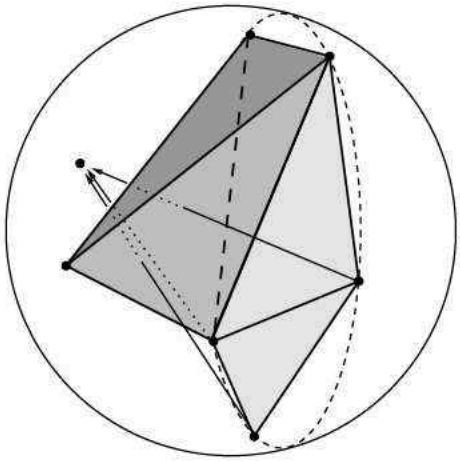


Figure 6.25: Possible error in a three-dimensional tetrahedralization of a cospherical point set.

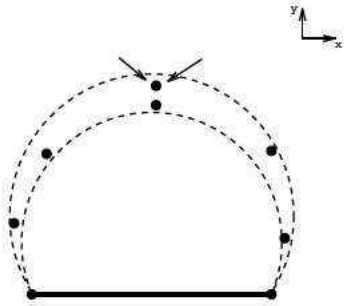


Figure 6.26: Approximately cospherical points can form unexpected constellations.

boundary is kept convex by "looking back" from each fourth point of the last generated tetrahedron to link it to all triangles with a positive minimum distance and thereby creating several tetrahedra in one step.

Once the *LSPB* is completely tetrahedralized, the "looking back" algorithm becomes unnecessary and it is returned to the normal tetrahedra growth process. Note that "looking back" and checking the minimum distance to the triangles is only required for triangles which belong to the tetrahedralization of the *LSPB*. As expected the implementation uses a temporary second higher priority queue for the advancing front within the sphere. Once the set $P_{c,i}$ is encountered and processed no later penetrations are possible.

The algorithm is generalized to tetrahedralize the points $\tilde{P}_{c,i}$ which possess general locations not exactly on the perimeter of a sphere. They do not necessarily form a convex Delaunay boundary. The λ criterion (Crit. 6.2) is applied for each generated tetrahedron. The point location must take all points in the search region into account (not only points from the subset $\tilde{P}_{c,i}$). If the criterion cannot be satisfied by a point of the subset $\tilde{P}_{c,i}$ with a small tolerance ε, the tetrahedron will not be created. The temporary boundary to which the points $\tilde{P}_{c,i}$ are added thus can contain non-convex regions which will be specially marked. If a point is being linked to all triangles with positive minimum distance such marked triangles are excluded.

Cocircular Point Sets

A cocircular point set (Def. 5.3) usually forms the intersection of two cospherical point sets as described in Section 5.4. Thereby, a facet is defined by the cocircular point set which forms the interface between the *LSPB*s of the two cospherical point sets. This facet does not possess a unique Delaunay Triangulation. The presence of such degenerate cases further complicates the tetrahedralization of the *LSPB*. When processing the second queue adjacent triangles must be taken into account to avoid facet connectivity inconsistencies (Fig. 6.28).

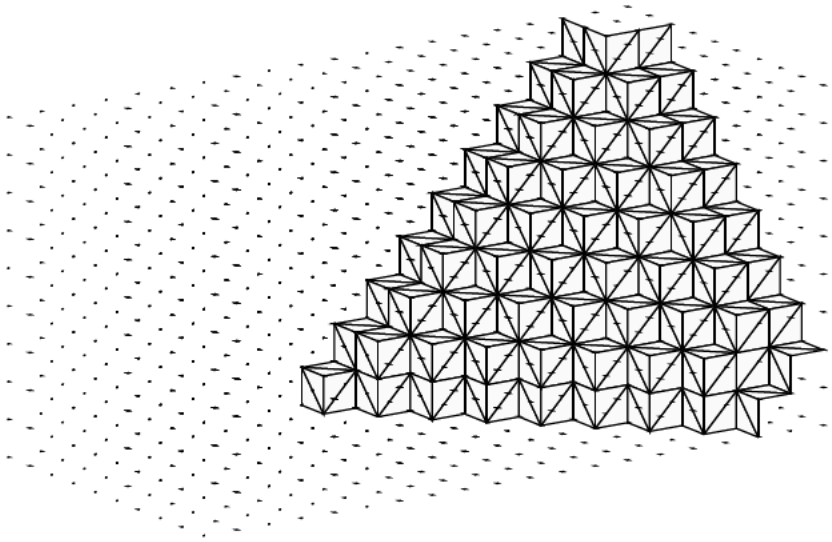


Figure 6.27: The advancing front of the global queue does not pass through a subset of cospherical points.

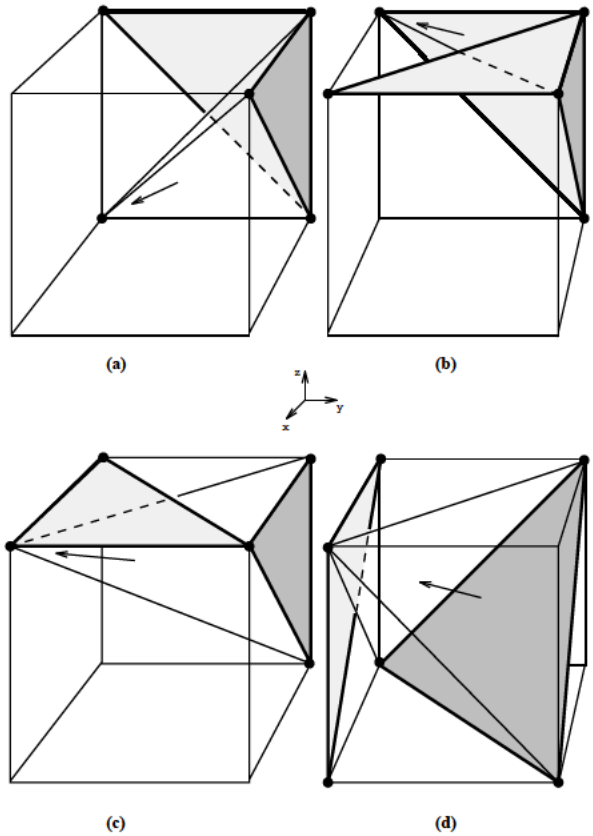


Figure 6.28: (a) One adjacent triangle to merge (b) Of two adjacent triangles only one can be merged correctly (c),(d) No adjacent triangles exist

The implementation generally aims to merge with adjacent triangles when choosing the fourth point for the construction of a tetrahedron. In theory such a degree of freedom exists due to the existence of several equal λ_i. In practice under finite precision arithmetics a small deviation ε from the minimal $\lambda_{\min}$ (Crit. 6.2) must be tolerated to allow a consistent merge with an adjacent triangle. The adjacent triangles in Fig. 6.28-b can be distinguished by calculating and maximizing normal distances of points to triangles. For each fourth point candidate from $\tilde{P}_{c,i}$ the normal distance to adjacent triangles is computed. The candidate which has maximal positive distance to all other adjacent triangles wins.

These techniques ensure a consistent tessellation in most cases. However, without further enhancements to the algorithm a consistent facet connectivity cannot be achieved in all cases due to the existence of untetrahedralizable polyhedra or twisted prisms (Section 5.5). For the advancing front style Delaunay algorithm it is useful to distinguish a general untetrahedralizable polyhedron and an untetrahedralizable cavity formed by Delaunay triangles. The modified advancing front algorithm will not create or encounter a general non-Delaunay twisted prism cavity. This follows from the fact that the advancing front consists of Delaunay triangles only. Thus, such a case may only occur if the facets from the input form a Schönhardt polyhedron. It will be transformed or refined by the surface preprocessor, because the triangles do not fulfill the Delaunay criterion. The question remains how to deal with the Delaunay twisted prism case, of which it can be guaranteed that it is formed by cospherical vertices.

Lemma 6.3 *If an untetrahedralizable pocket is formed by Delaunay triangles, its vertices must form a cospherical point set.*

Proof 6.3 *The Delaunay Triangulation of any point set as the dual of the Voronoi diagram must exist. Assuming a point set with a unique Delaunay Triangulation (no cospherical point set), a Delaunay triangle composed of any three points from the set must be contained in the tetrahedralization. Thus, it is not possible to form an untetrahedralizable pocket with a subset of the Delaunay triangles from the tetrahedralization.* ■

Figure 6.29 shows a twisted prism which is formed by a cospherical point set $P_{c,i}$. Hence, all its surface triangles are Delaunay triangles. No correct fourth point can be chosen for any of its surface triangles to perform a correct merge with the adjacent triangles. By rotation of the upper three vertices relative to the lower three vertices two variations of the prism can be constructed while the vertices remain cospherical (Fig. 6.29). Hence, all versions of the prism satisfy the Delaunay criterion and can occur during the formation of the advancing front. Without rotation three subsets of cocircular points are formed on each side of the prism. Recall the problem of facet connectivity which was the motivation for the examination of the twisted prism. While the first type in the figure, a convex polyhedron, poses no difficulties except that a bad sliver element will be created, the other two types seem a dramatic dead end for the advancing front Delaunay tetrahedralization engine. But in practice for very small rotation angles the three versions of the prism cannot even be distinguished numerically. This leads to the idea to find an algorithm which treats all cases in the same way as can be dealt with the harmless convex polyhedron type. Therefore one should observe the relationship between slivers and the twisted prism. As shown in Fig. 6.29 an *additive* or *subtractive* combination of slivers with one prism type always results in another type. A convex prism

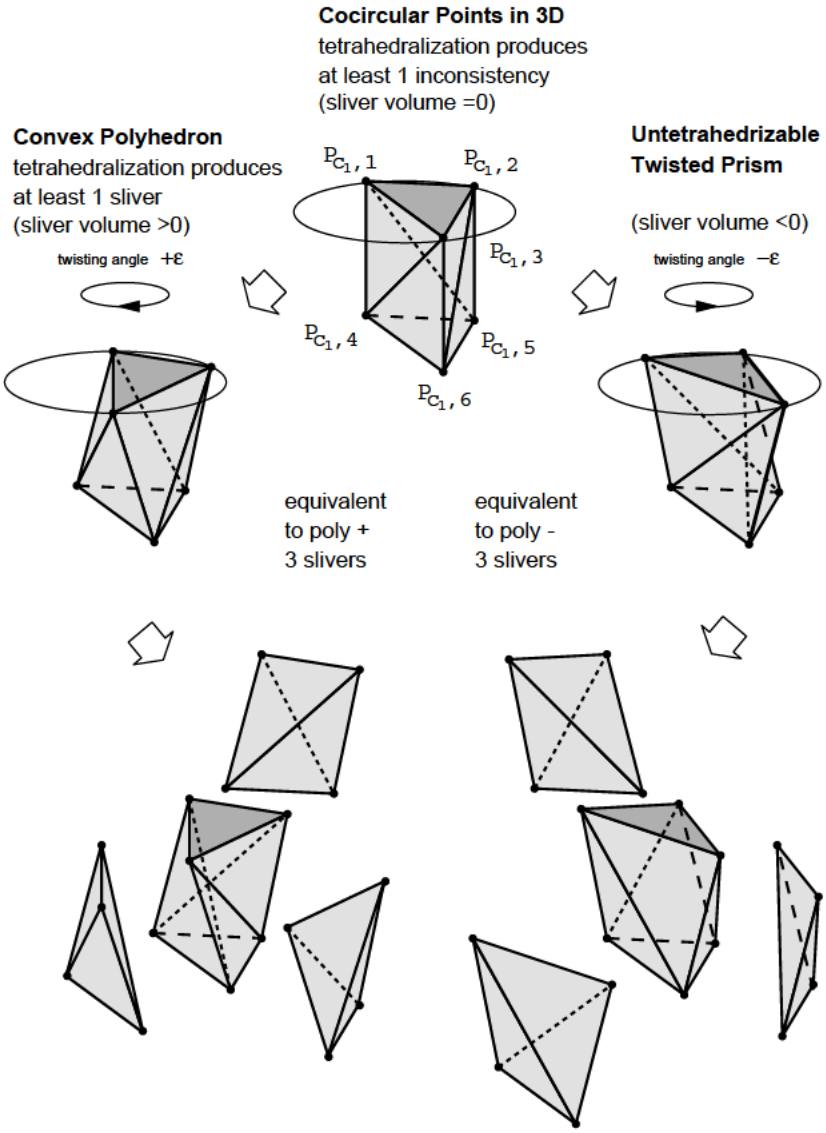


Figure 6.29: A twisted prism with cospherical vertices and three cocircular point subsets. It can be converted into a convex polyhedron or into an untetrahedralizable polyhedron while keeping the vertices cospherical. Thus, in all cases all triangles satisfy the Delaunay criterion.

can be formed by attaching slivers to the sides of the untetrahedralizable prism. Hence, the mesh topology and facet connectivity *can* be kept consistent elegantly for all cases if one allows the creation of slivers with *negative*, *zero*, or *positive* volume (orientation). For such a robust algorithm it is indifferent whether the volume of an extremely flat sliver tetrahedron appears to be exactly zero, positive, or negative.

After the volume tetrahedralization an adaptation step which is responsible for geometrical quality and slivers must guarantee the removal of all zero and negative volume slivers. Various Delaunay sliver types were discussed in Section 5.7. The type of sliver which will be created here in the context of the twisted prism is one with vertices from a cospherical point set. Such a sliver can be removed by a local transformation without modifying the points. The Delaunay property is thereby maintained.

The implementation of the tetrahedralization had to be extended for such sliver elements of zero or negative volume. The tetrahedralization of the *LSPB* may produce negative sliver elements when it merges with adjacent triangles which are already attached to tetrahedra. If adjacent triangles are not yet attached to tetrahedra, e.g. boundary triangles, the creation of slivers is avoided. Instead the implementation of the tetrahedralization engine possesses the power to flip triangles on the fly.

An important advantage is that no intersection tests are required to check for such cases. The test if an identical triangle exists (Lemma 6.1) is extended for cocircular point sets. If at least one not yet merged triangle exists with vertices from a cocircular point set which has the opposite orientation of the base triangle, zero or negative sliver elements are required. This becomes clear if one considers that the tetrahedralization of the *LSPB* occurs in a single step. Hence the plane spanned by the cocircular point set as part of the cospherical point set is either entirely triangulated or possesses no triangles at all. Without going further into detail it can roughly be said that all triangles from the triangulation of the (coplanar) cocircular point set are identically oriented.

In practice, one realizes that zero or negative volume elements are hardly required and in most cases the tetrahedralization of the *LSPB* can be directly fitted to the adjacent triangle constraints.

6.5 Local Adaptation

The implemented fully unstructured volume decomposition algorithm is well suited for remeshing purposes. Mesh points can be erased as well as inserted, and deformed elements can be discarded. This full freedom point insertion and removal allows many different kinds of adaptation techniques. One possibility is the insertion of circumcenters of not well shaped tetrahedra. The necessary steps are similar to the described method for inserting Steiner points at the surface (Section 6.3).

- Find elements which need to be checked for deformations or which require a mesh update. For example locate the element which contains the circumcenter of a badly shaped tetrahedron.
- Delete or insert a mesh point by updating the mesh topology and thereby creating

unverified, possibly non-Delaunay tetrahedra. For example insert the circumcenter as a new mesh point into the found element.

- Verify the Delaunay criterion for all connected elements and remove non-Delaunay elements.
- Recurse by removing non-Delaunay elements which are connected to already removed elements. This is a reversal of the tetrahedra growth process by the modified advancing front algorithm.
- Reprocessing the combination of the resulting cavity and the mesh points. No surface preprocessing of the cavity surface is required. The modified advancing front algorithm performs the tetrahedralization of the local region by queuing the unconnected triangles. These triangles must already be Delaunay due to the previous recursive removal process.

All required functions to maintain a consistent data structure have been implemented for all simplices.

Create: These functions allocate point, triangle, or tetrahedron entities. The data structure which is composed of a set of consistent links and references is not yet updated. Only the *forward* references will be set. These are the coordinates of a point, the vertices of a triangle, and a triangle/vertex combination for a tetrahedron.

Insert: These functions actually insert the created entities by making the data structure consistent. A consistent data structure is thereby defined by correct *forward* and *backward* references. For example each point possesses a list of pointers to incident triangles. A triangle contains pointers to attached tetrahedra.

Remove: These functions are necessary to delete or modify elements. The element must be cut out leaving a consistent data structure behind. If a point is removed, all incident triangles and tetrahedra will be removed to maintain consistency. If a triangle is removed, the possibly attached tetrahedra will be removed as well. All *backward* references have to be updated by setting the corresponding pointers to removed higher-dimensional elements to null. This applies only to the maintained data structure and not to the cut out elements. Thus, through the cut out element all connected and removed elements are still accessible for e.g. modification purposes. For example a cut out point may hold references to removed triangles and tetrahedra.

Erase: These functions finally free the memory space consumed by cut out elements and their connected higher-dimensional simplices.

Search: Search functions only operate on a consistent data structure with correct *backward* references.

Refine: These functions allow the insertion of a point at an arbitrary position on a given element. Depending on the type of function edges of triangles, triangles, edges of tetrahedra, triangles of tetrahedra, and tetrahedra can be refined. All *backward* and *forward* references will be updated automatically.

Experiments showed that a tetrahedron is ideally represented by a pointer to a triangle and a pointer to the fourth vertex which closely reflects the modified advancing front algorithm. In this way the memory space consumed by a tetrahedron is reduced to two pointers instead of e.g. four pointers for each vertex. This type of data structure severely complicated the coding of the necessary manipulation functions. Especially the refinement functions were required to evaluate geometric constellations to find a correct combination of existing oriented triangles and existing points for the creation of new tetrahedra during the insertion of a refinement point. Fortunately, only the coding and not the performance during runtime was affected.

Local transformations were implemented to swap facets and their attached tetrahedra. With this technique the negative volume (orientation) sliver elements can be removed. For the example of such a sliver with a small negative volume the facet swapping technique modifies the local connectivity between the neighboring elements in such a way that the sliver element can be removed and the overlap between two tetrahedra due to the untetrahedralizable *LSPB* is resolved. In this case the resulting topology still fulfills the Delaunay criterion, because the elements are formed by point sets $P_{c,i}$.

Chapter 7

Examples

7.1 CAD

A few mesh examples are shown which are not related to semiconductor process or device simulation. A wide field of applications exists ranging from computational fluid dynamics or stress mechanics to computer tomography, anthropology, or computer graphics. Models were obtained from some of these areas to show the versatility of the implemented mesh generator.

The modified advancing front algorithm allows to pause the meshing process to take a snapshot of the incomplete mesh. The advantage is that such a snapshot reveals those parts of the mesh which are finished and those parts which were not yet touched at all. The finished parts will stay unchanged in the final mesh. In such a way it can be quickly determined *where* faults in the surface description exist. If the program runs into problems or shows unexpected behavior, the developer can halt the process and view the current state of the front and the problem region.

Scientific visualization is a challenging task and is important to grasp the results from a three-dimensional simulation [51]. To provide a meaningful feedback to the user different techniques must be developed to appropriately prepare scalar or vector data and to expose the interior of a mesh which becomes inherently difficult in three dimensions [57]. A powerful visualization toolkit [150] was used to render some of the more complex images shown in this chapter.

The first example is a model of a hand. Snapshots of the advancing front are shown in Fig. 7.1. The second example is a model of Beethoven's bust. The surface mesh is depicted in Fig. 7.2. Extracting all structural edges results in a contour plot (Fig. 7.3). Figure 7.4 shows the model of a cow. The meshes for all three examples are depicted in Fig. 7.5, Fig. 7.6, and Fig. 7.7. The edges and points of the mesh of the hand rendered as tubes and spheres can be seen in Fig. 7.6. Computation time is minimal (less than a minute) and the mesh contains 4344 tetrahedra. For the example of the cow and Beethoven's bust the tetrahedral elements are itself visualized by shrinking them (Fig. 7.5, Fig. 7.7). The mesh of Beethoven's bust contains 17665 tetrahedra and the mesh of the cow 11608 tetrahedra.

The tetrahedralization domain for a quite different example, a human skull (Fig. 7.8), is extremely non-convex. In fact, only the outer shell has been tetrahedralized and the region of

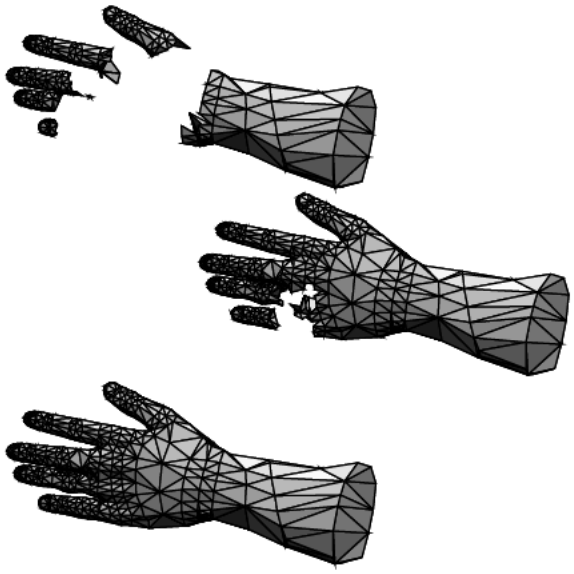


Figure 7.1: The advancing front at several moments during the meshing process. It separates the meshed region from the empty parts of the volume.

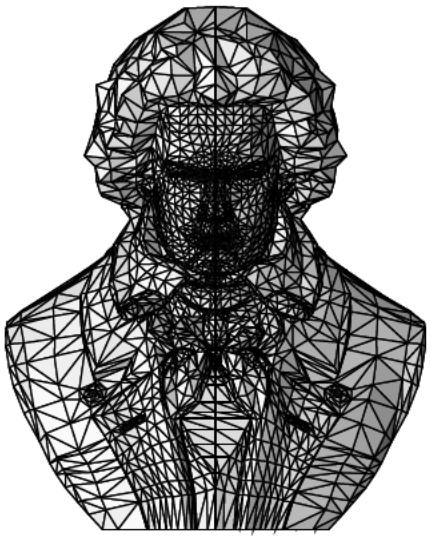


Figure 7.2: Surface mesh of Beethoven's bust.

Figure 7.3: Structural edges form a contour plot.

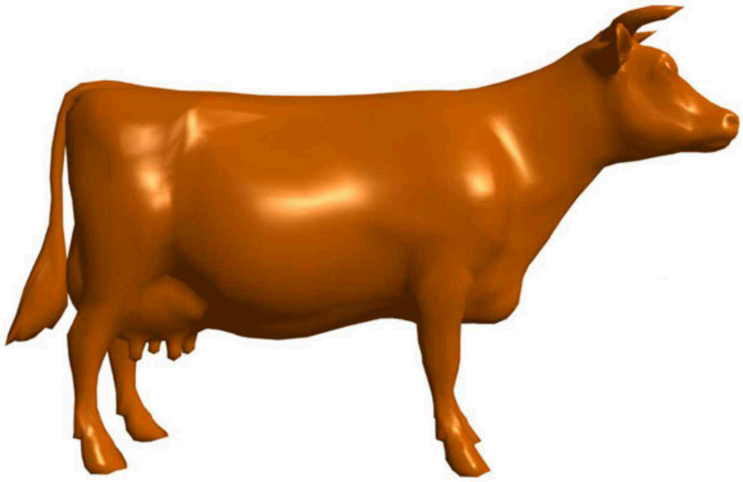


Figure 7.4: The model of a cow.

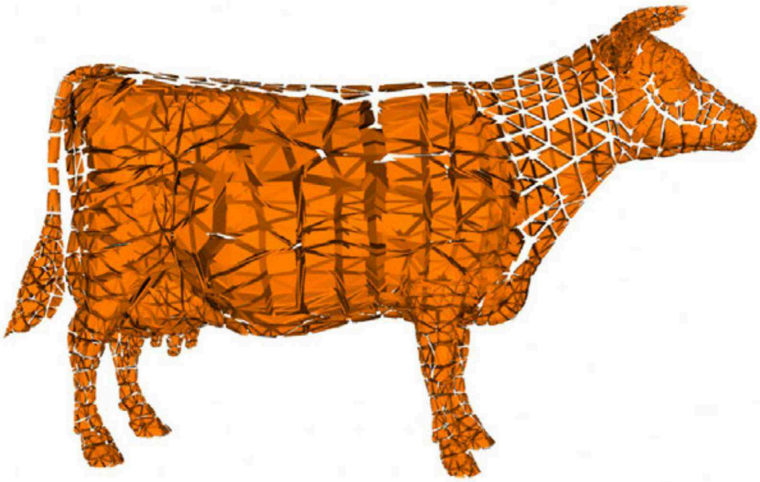


Figure 7.5: The mesh of a cow with 11608 elements.

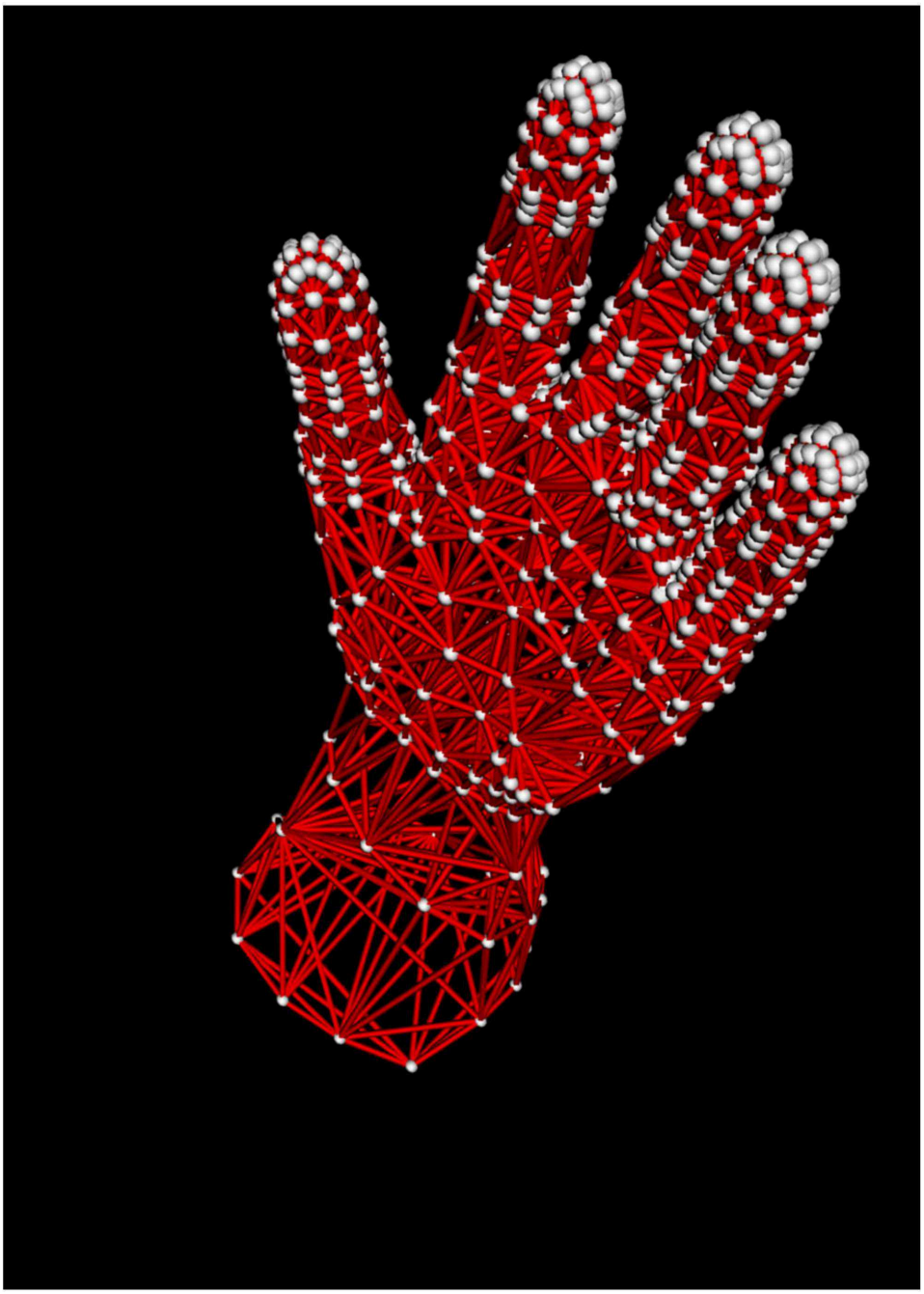


Figure 7.6: Edges and points of the final mesh.

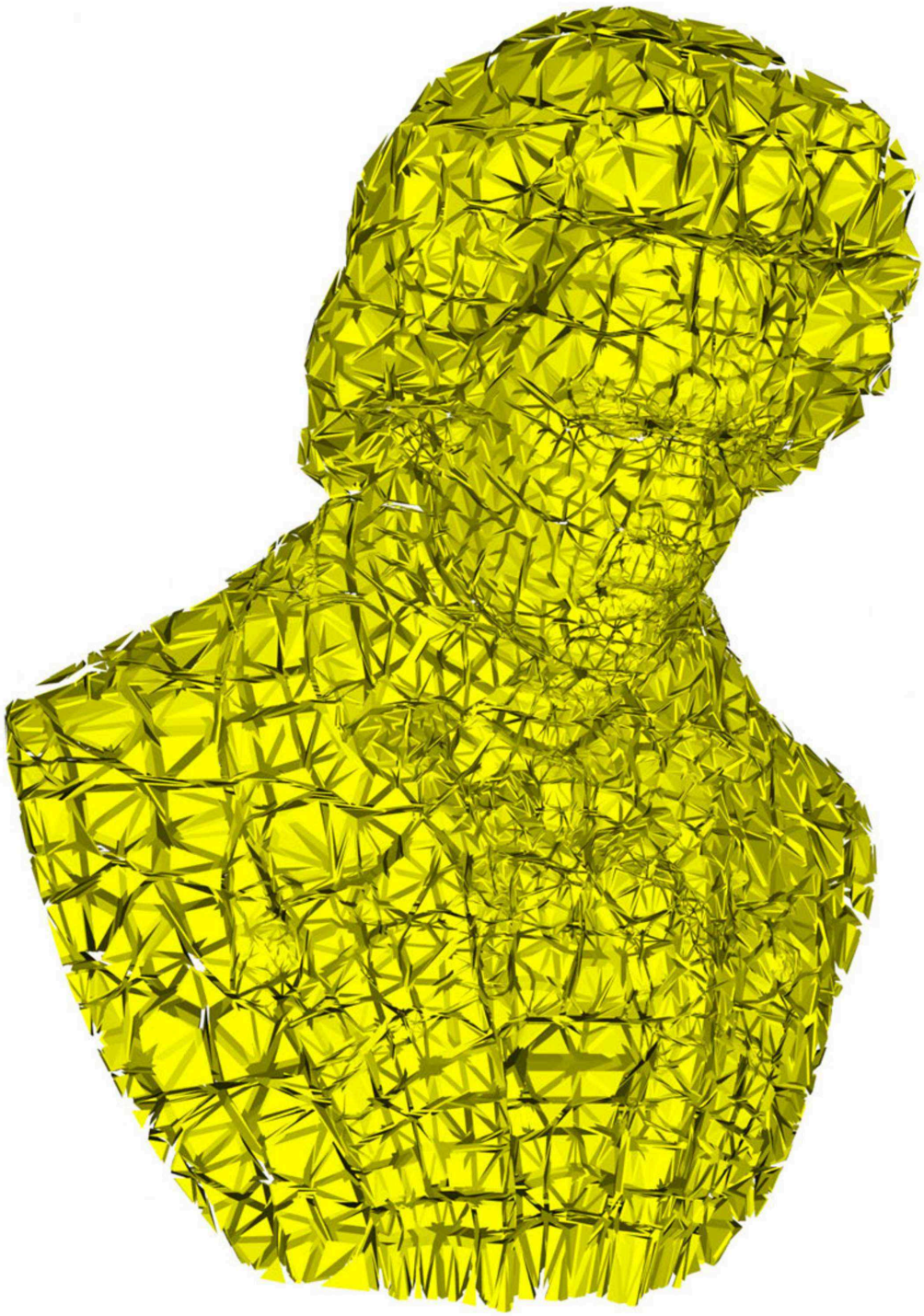


Figure 7.7: Final mesh of Beethoven's bust with 17665 elements.

the brain itself was never processed. Figure 7.9 shows cross-sections of the mesh at interesting parts of the skull. It can be seen that the meshed volume is much smaller than the volume covered by the convex hull. The top left picture in Fig. 7.9 shows the cross-section of the jaw area. In the following pictures one can see the nasal cavity, nasal bone, and nasal septum. The bottom right picture contains the cross-section of the skull above the nose and eye area.

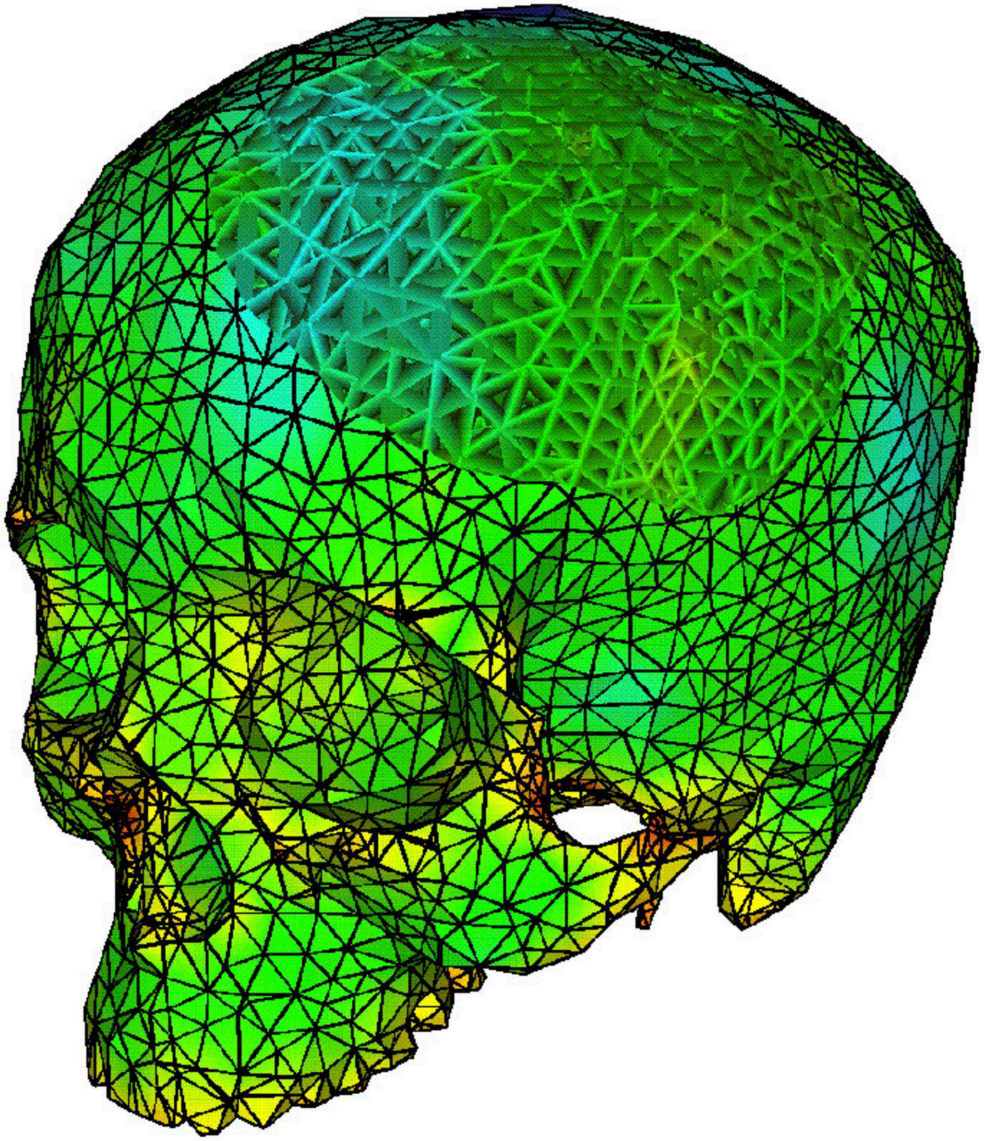


Figure 7.8: Human skull, mesh with 28512 elements.

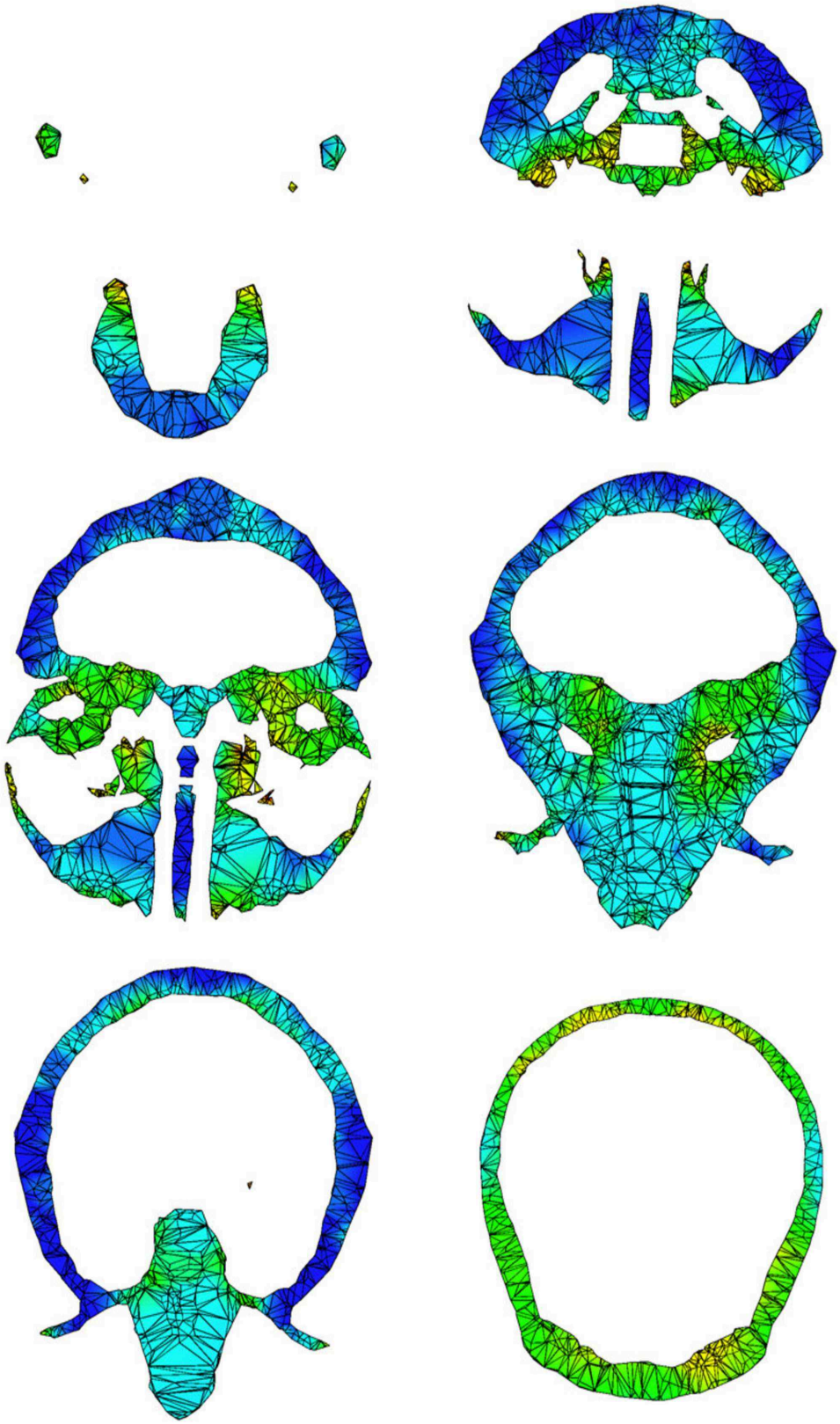


Figure 7.9: Crossections of the human skull mesh.

7.2 Interconnects

An important field of application is the capacitance extraction and thermal analysis of interconnects in semiconductor devices. The finite element package *SAP* [140] can be used for electro-thermal interconnect simulation. It contains a layer-based preprocessor to generate three-dimensional meshes from cross-sections. This limited approach forces a uniform element size along one coordinate axis as was described in Section 4.2.1. The lateral mesh density must be identical for all cross-sections. Figure 7.10 shows the interconnects for a region of a DRAM. With some restrictions the layer-based preprocessor of *SAP* allows to apply a linear transformation function to a layer. The conductors depicted in Fig. 7.10 were modeled in such a way. The mesh generated with *SAP* for such a layered solid model is shown in Fig. 7.11. The same interconnect structure meshed with the fully unstructured modified advancing front method is presented in Fig. 7.12. The variation of the element size in Fig. 7.12 compared to the mesh in Fig. 7.11 shows the increased flexibility of the approach.

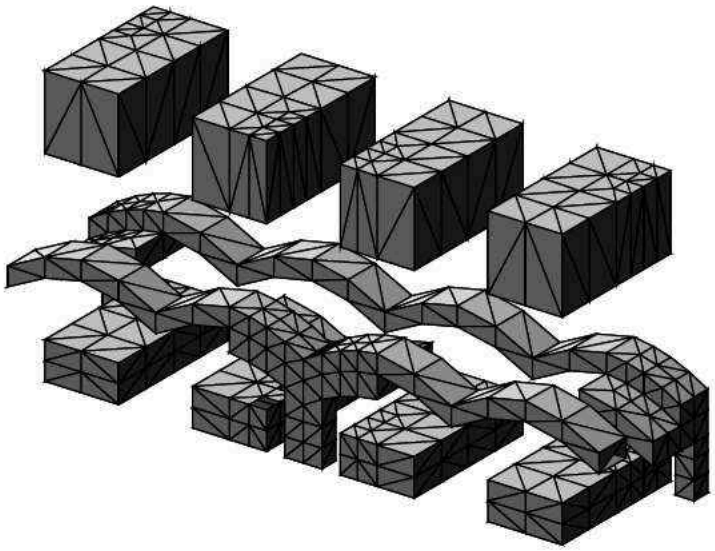


Figure 7.10: Structure of the discretized conductors in a DRAM ($0.8\mu m$ x $3.2\mu m$).

A much bigger challenge arises if the solid model is not edited but rather derived from process simulation. The structures evolving after etching and deposition steps are more complex than can be handled with methods such as the layer-based method. The output of a topography simulator is often highly refined to match the required resolution for the manipulation of the structure. Such a more complex two layer interconnect structure typically

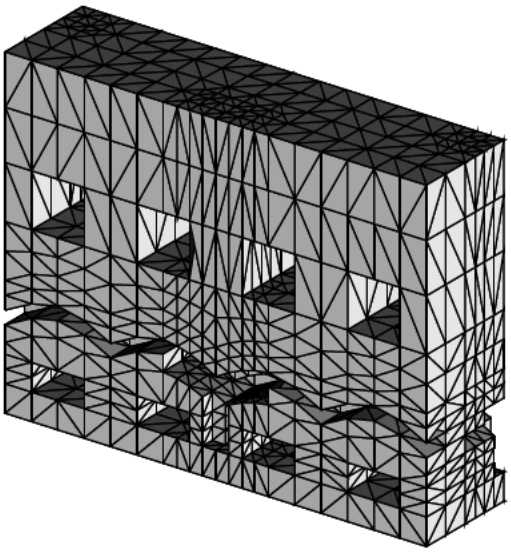


Figure 7.11: Mesh generated with the layer-based method, 6480 tetrahedra. The vertical propagation of refinement through all layers can be observed.

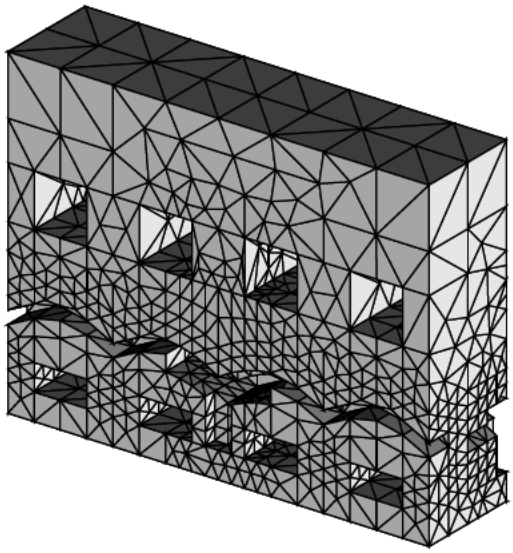


Figure 7.12: Fully unstructured Delaunay mesh of the DRAM, 5290 tetrahedra.

evolving from the simulation of semiconductor processes is shown in Fig. 7.13 and Fig. 7.14. In this example the layout consists of two masks which define the metal lines for each of the two layers. A three-dimensional etch simulation with *ETCH3D* [170] is used to derive the structure from this layout data. Each layer results from directional etching of a metal film which is masked with the resist pattern. After stripping the resist for the first layer, an isolating oxide layer is deposited. The second layer metal film is deposited on top of this oxide layer. The structure depicted in Fig. 7.13 shows the state after stripping the resist for the second layer. The isolating oxide layer between the two metal films and before further oxidation is shown separately in Fig. 7.14.

The geometrical data available to the mesh generator consisted in this case only of the polygonal description of the material interfaces. The closed boundary representations of the material segments were generated during the meshing process. The seed triangles for the enclosed segments of each material were extracted from the interface data given by the topography simulator.

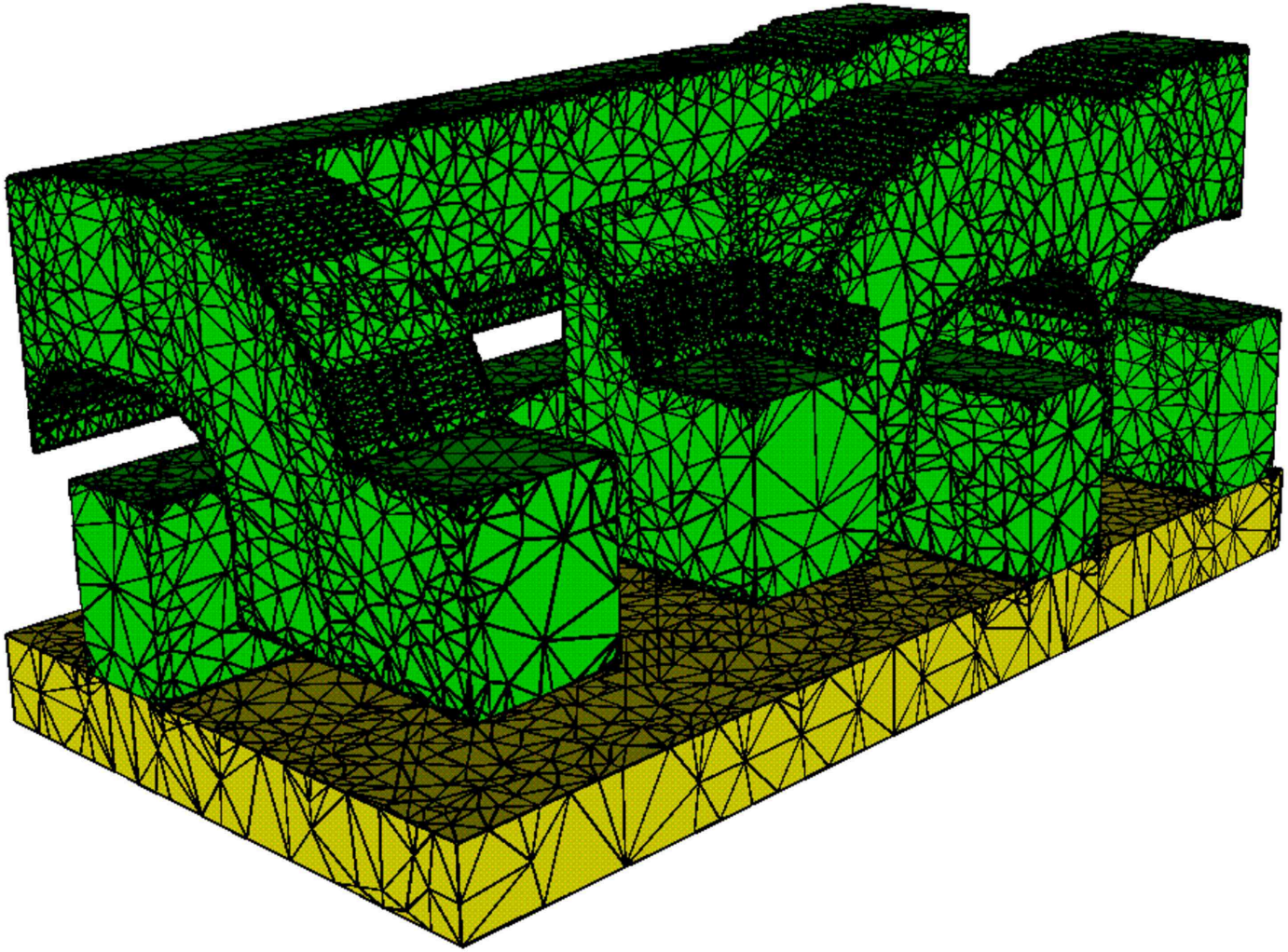


Figure 7.13: Silicon bulk (5139 elements) and four metal lines (51682 elements).

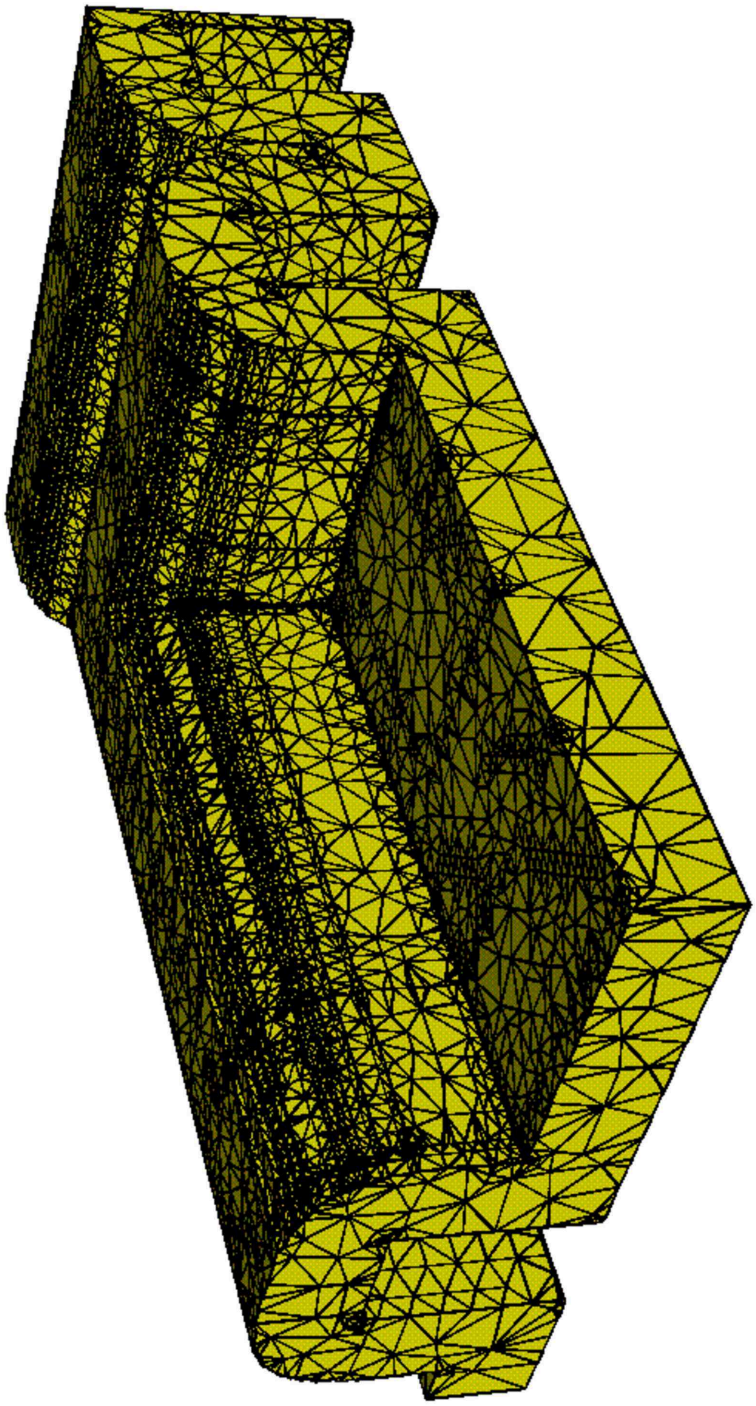


Figure 7.14: Oxide layer, 47203 elements.

7.3 Chemical Vapor Deposition and Reaction Kinetics

High pressure chemical vapor deposition *CVD* of Tungsten is used for a Ti/TiN/W plug fill process. The geometry results from an initial low pressure deposition of a TiN barrier layer into the via. This physical vapor deposition *PVD* process is determined by ballistic transport of the sputtered Ti particles. For the subsequent high pressure *CVD* process it is assumed that W is reduced from WF_6 using H_2 and forming HF as by-product. The three gas species diffuse in the via and the reduction takes place at its surface. Depending on the diffusion coefficients and the reaction rates a steady state of the gas distribution is reached leading to a depletion of WF_6 at the bottom of the via and to non-uniform deposition rates. This results in a characteristic overhang in the layer profile. The simulated structure is located at an off center position of the wafer. Thus the TiN layer, formed by sputter deposition prior to the Tungsten *CVD* is strongly asymmetric requiring the rigorous three-dimensional simulation of the *CVD* film formation.

The chemistry model is calculated on the mesh with *AMIGOS* [128] which provides an analytic interface for discretizing and solving differential equations. Figure 7.16 shows the mesh of the cylindrical via and the WF_6 concentration. A cross-section of the mesh is depicted in Fig. 7.17. A different mesh with a highly refined region near the boundary and in the interior of the via was generated by constructing a non-uniform mesh point distribution derived from the boundary vertices (Fig. 7.18).

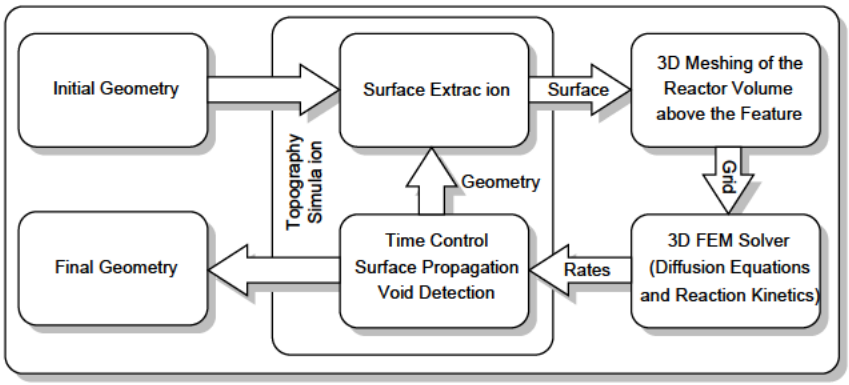


Figure 7.15: Flow diagram for the high pressure *CVD* model.

The calculated deposition rates are further used to advance the structure surface through topography simulation with *ETCH3D*. As shown in Fig. 7.15 the complete setup consists of several tools which are directly linked to allow a fully automatic simulation sequence for as many iterations as desired [126]. After extracting the surface of the initial geometry, a three-dimensional mesh of the gas domain above the considered structure is generated. The differential equations describing the mass transfer and the reaction kinetics are set up and evaluated with *AMIGOS* on this unstructured mesh. The resulting deposition rates are transfered to *ETCH3D*. The topography simulator controls the time step for the surface propagation which is especially important during the formation of voids. Underestimating the size of such a void is avoided by reducing a too large time step so that the first closure of the void can be observed. The surface of the resulting cellular geometry is extracted and

the procedure is repeated for each time step. The parameters for the meshing tool and the description of the rate model are set up in control files and remain unchanged during all time steps. In this way the process runs fully automatic without any user interaction.

The required CPU time on an Alpha-station 600/333 for the example shown in Fig. 7.19 is approximately 60 minutes for the complete, automatically controlled simulation sequence, including surface extraction, meshing, calculation of the deposition rates, time step control, void detection, and surface propagation. Depending on the size of the structure between 10.000 and 30.000 tetrahedra were used for the continuum transport model. The same model was calculated on a damascene structure and the resulting WF_6 concentration is shown in Fig. 7.20.

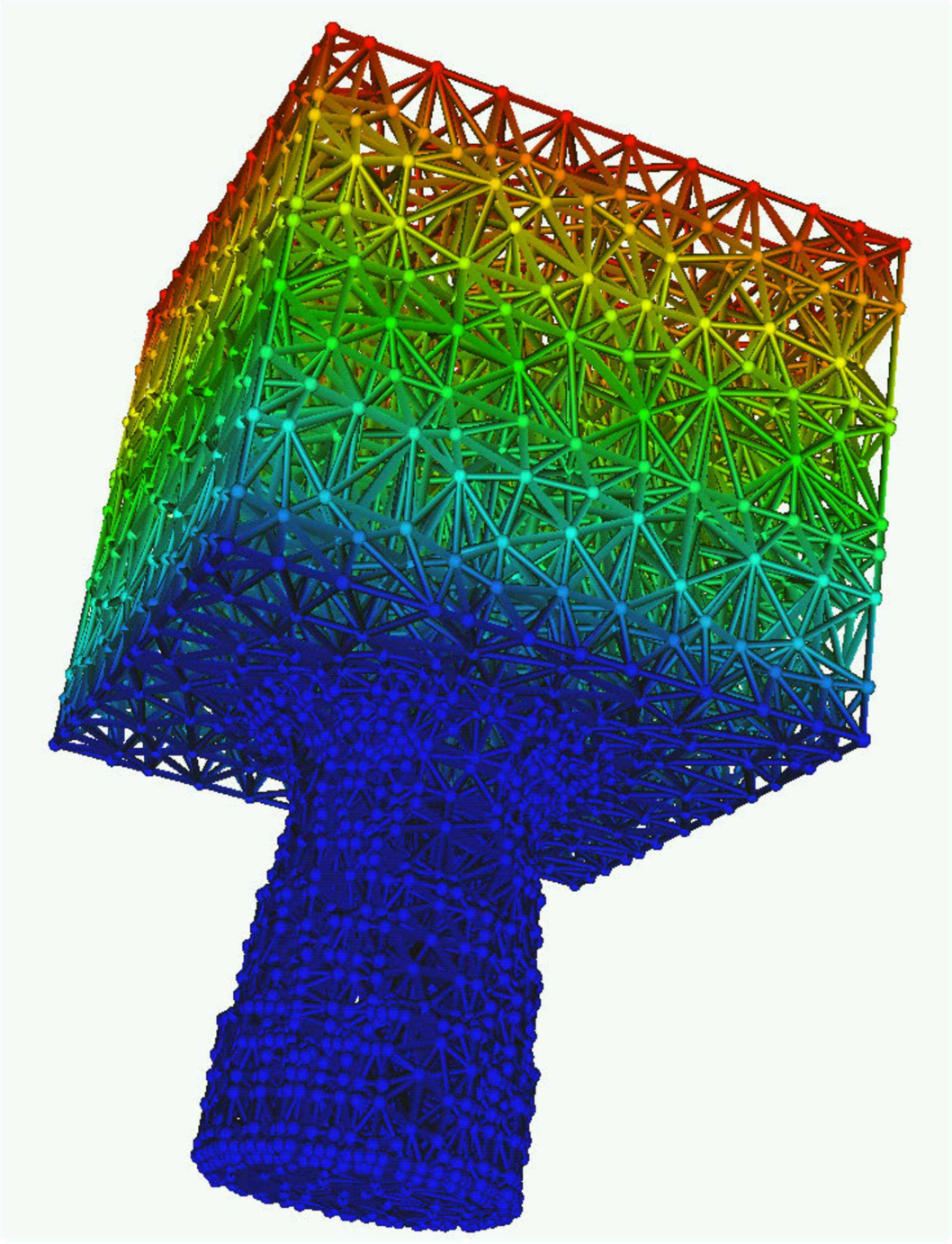


Figure 7.16: A cylindrical via, mesh with 7324 elements

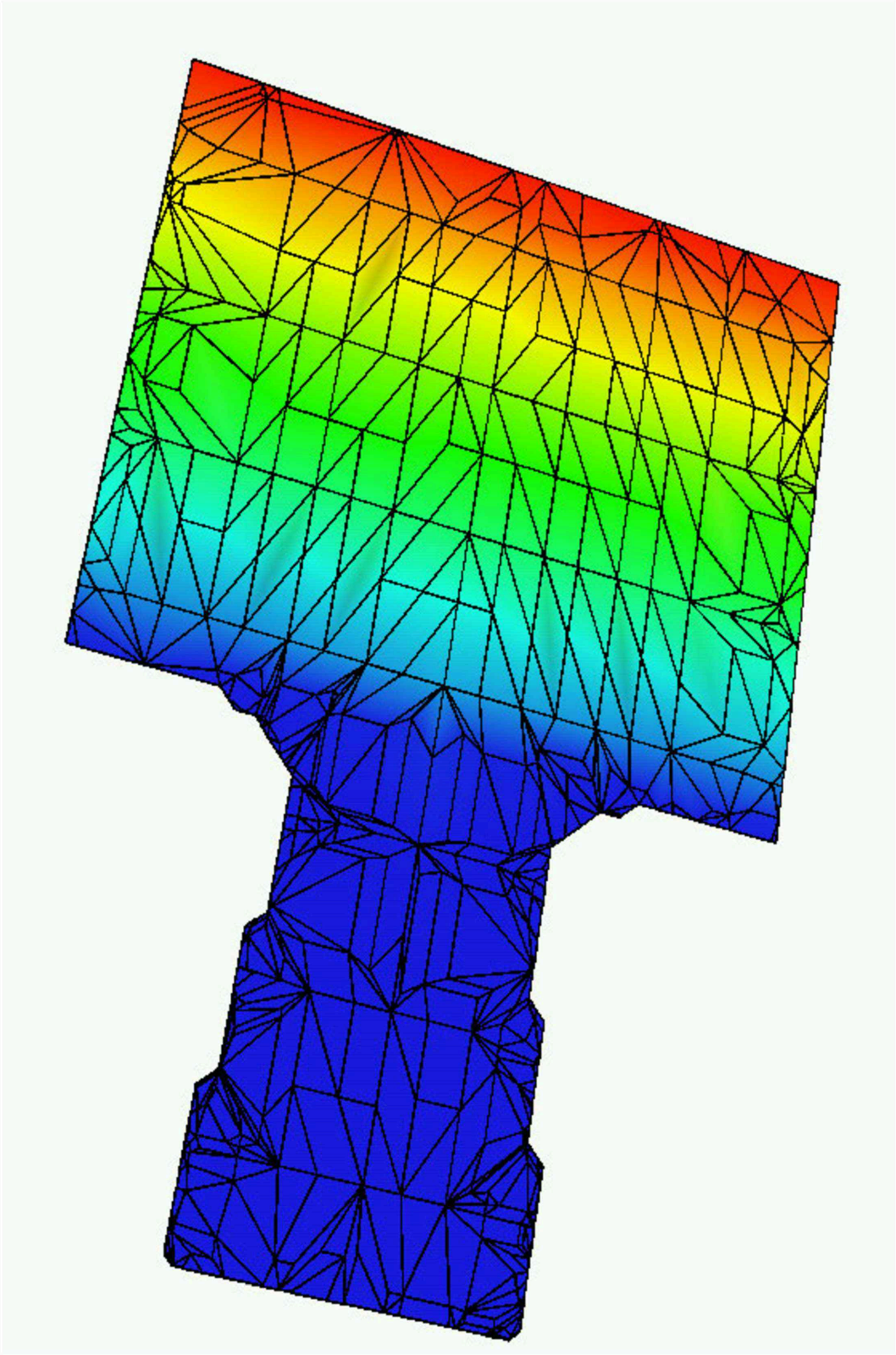


Figure 7.17: A cross-section of a cylindrical via with a uniform mesh point distribution, 7324 elements.

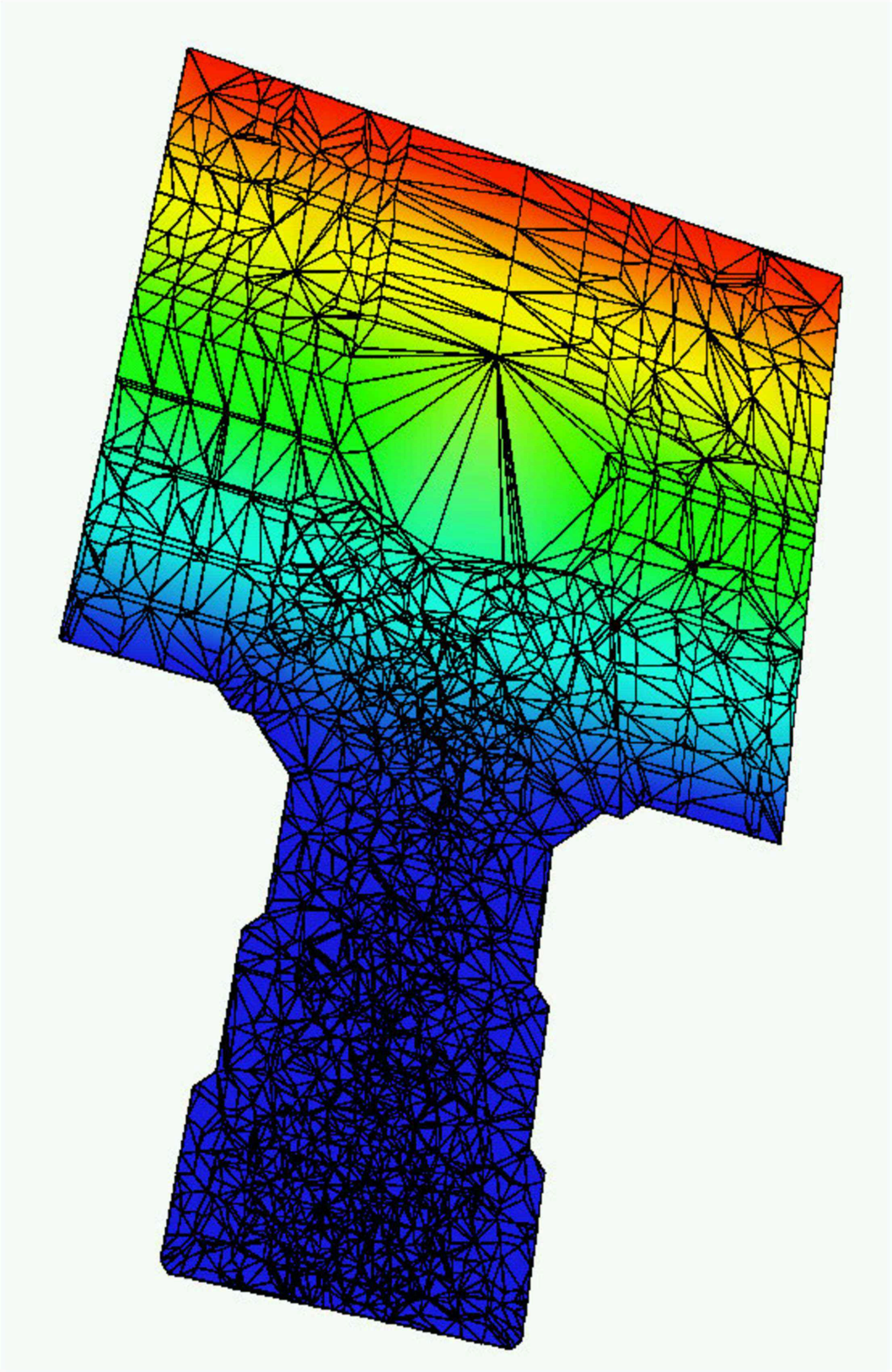


Figure 7.18: The cross-section of the same cylindrical via meshed differently shows the non-uniform distribution of mesh points, 75720 elements.

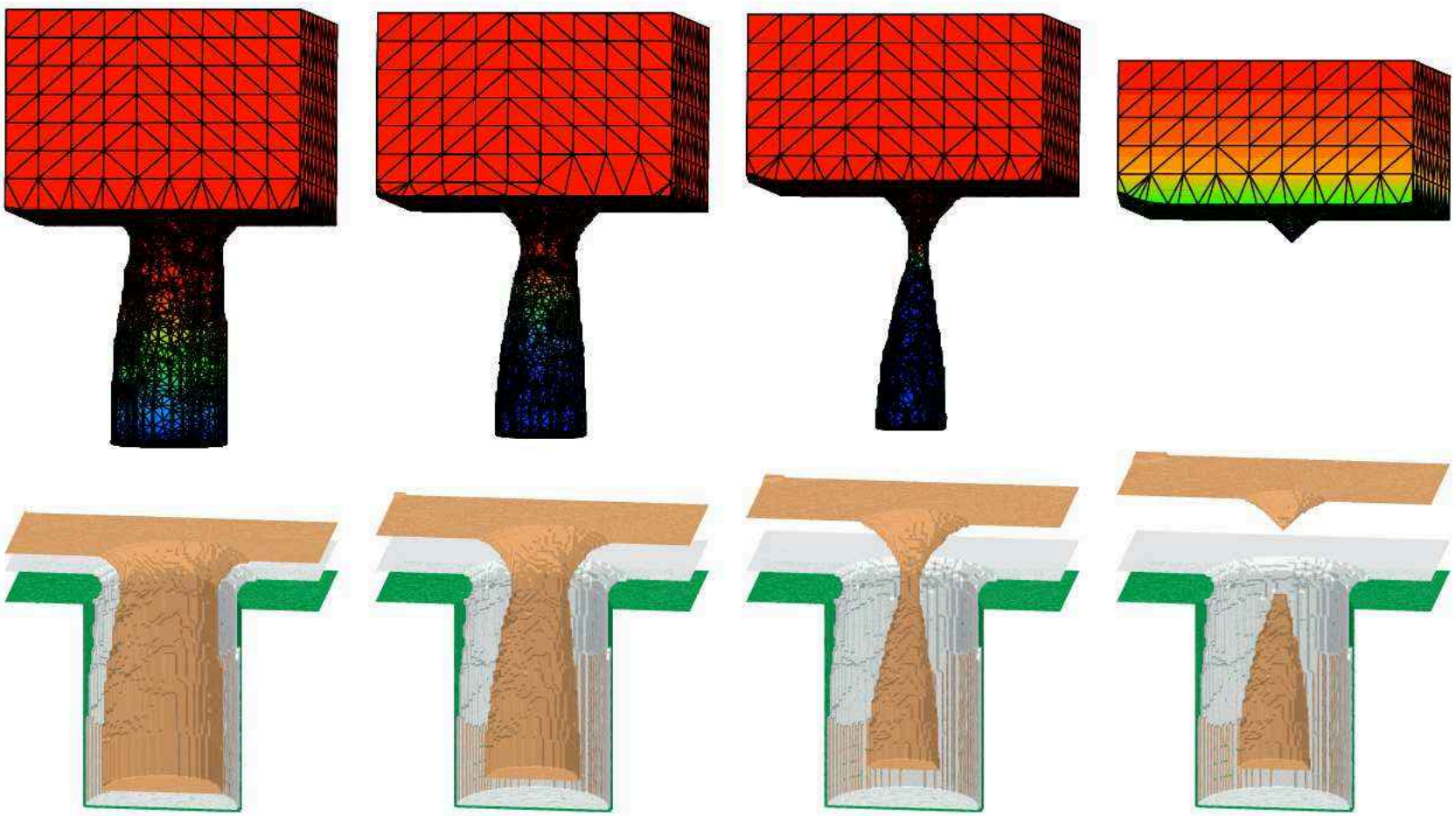


Figure 7.19: The volume meshes used for the continuum transport model and the corresponding three-dimensional film profiles for a sequence of time steps of a Ti/TiN/W plug fill process. The profiles show from bottom to top the initial circular via, the *PVD* TiN layer formed by sputter deposition and the *CVD* tungsten layer.

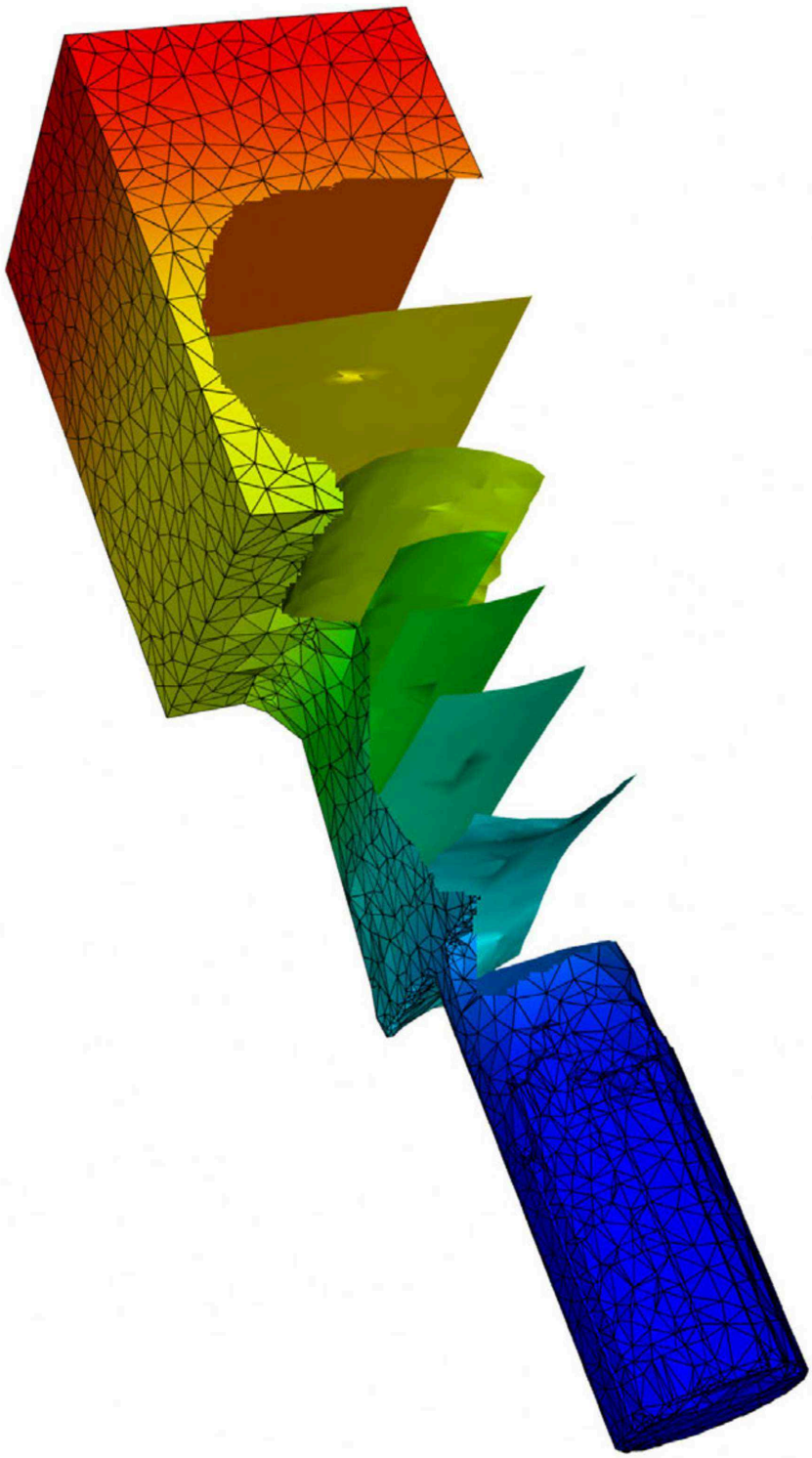


Figure 7.20: Iso surfaces of the WF_6 concentration in a damascene structure, mesh with 18148 elements.

7.4 NMOS Transistor

The example in Fig. 7.22 shows a NMOS transistor in $0.18\mu m$ technology. Two regions of field oxide isolate the bottom silicon bulk from the poly-silicon. The gate is formed in the center of the depicted structure between the field oxides under the crossing poly-silicon. The gate-oxide which isolates the poly-silicon from the silicon at the gate area is $5nm$ thick and not visible in the figure. The size of the gate is given by the width of the poly-silicon and the distance between the field oxides ($0.18\mu m \times 0.18\mu m$). A $30nm$ thin oxide layer covers the entire structure. The structural edges of the device can be seen in Fig. 7.21.

Devices with high ratios between local feature sizes pose a challenge to most existing meshing methods. The difficulty lies in the anisotropic grading of the mesh density. Isotropic grading results in a mesh with a too large number of elements in three dimensions. The above described device exhibits such a geometrical anisotropy which was managed through a specific technique for the generation of the internal mesh points. As can be observed in Fig. 7.23 the mesh possesses long and thin elements to resolve the thin layer. The mesh points were generated at cross-sections of the structure. The tetrahedralization performed by the modified advancing front algorithm produced the desired anisotropic elements. As can be seen from the left side wall in Fig. 7.23 the mesh is fully three-dimensional and could not have been generated by extruding a two-dimensional mesh as for example through a layer-based product method which was described in Section 4.2.1). However, the resulting mesh elements are not completely ideal, because their anisotropy is one-dimensional. Flat prismatic elements may be preferred to resolve the thin layer.

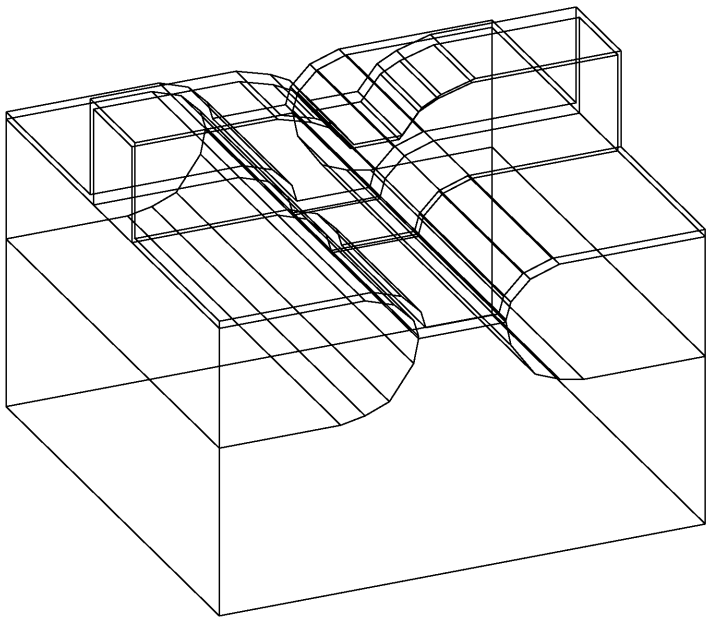


Figure 7.21: Structural edges.

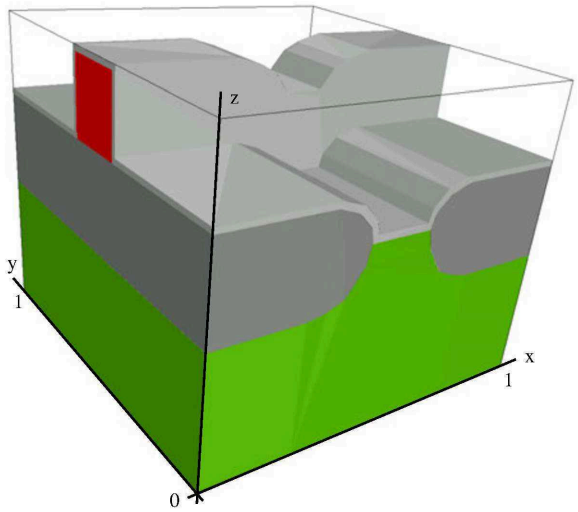


Figure 7.22: NMOS Transistor with a thin oxide layer.

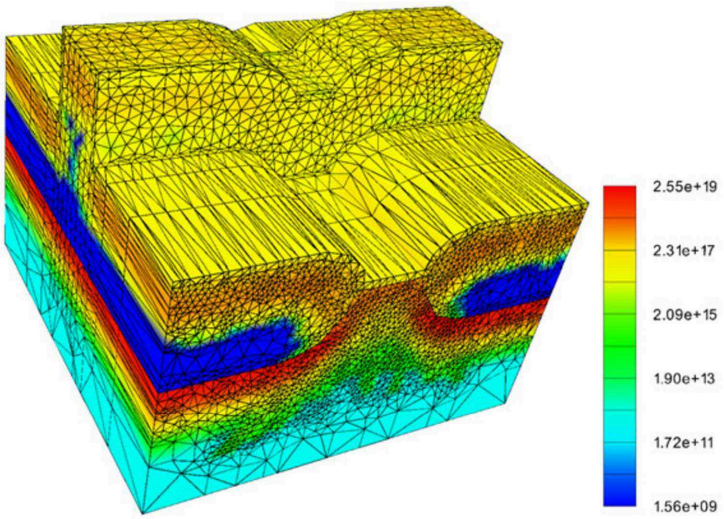


Figure 7.23: Boron implantation profile, mesh with 134374 elements.

7.5 CMOS Inverter

A more complicated structure which consists of two transistors is shown in Fig. 7.24. The NMOS transistor and the PMOS transistor form a typical complementary MOS (CMOS) device. While the geometrical structures of the two transistors cannot be distinguished from each other (Fig. 7.24), it is the doping profile which differs. The blue material segments denote the two poly-silicon gate regions. The green segment of the structure is the silicon which is separated from the gate by a thin oxide layer. This oxide layer covers a shallow trench around all four sides of each transistor to ensure a proper isolation (shallow trench isolation, STI). The mesh density in the oxide layer is of no relevance. However in the silicon region the mesh density is crucial to resolve the doping profile and the electrical current density. The source and drain contacts penetrate the oxide layer to reach the silicon.

The red segments are silicon-nitride spacers which help to isolate the gate from the source and drain regions and which prevent adversary effects due to a source-gate or drain-gate capacitor. From a process technology point of view the spacers allow a further separation between the deep drain/source implant regions and the gate. The source/drain extensions with a shallow doping profile reach under the spacer.

The metallization as shown in the figure consists of the Aluminum conductors which connect the two gates and the two drain contacts. The passive oxide layers which cover the entire device and which form the isolation between the metal layers are stripped to make the interconnects visible in the figure. A quite coarse mesh is depicted in Fig. 7.25. It consists of 1016 and 1005 tetrahedra for the two gate regions, 3199 tetrahedra for the four spacers, 13620 tetrahedra in the oxide layer, 15511 tetrahedra for the silicon, and 1283 tetrahedra for the metallization.

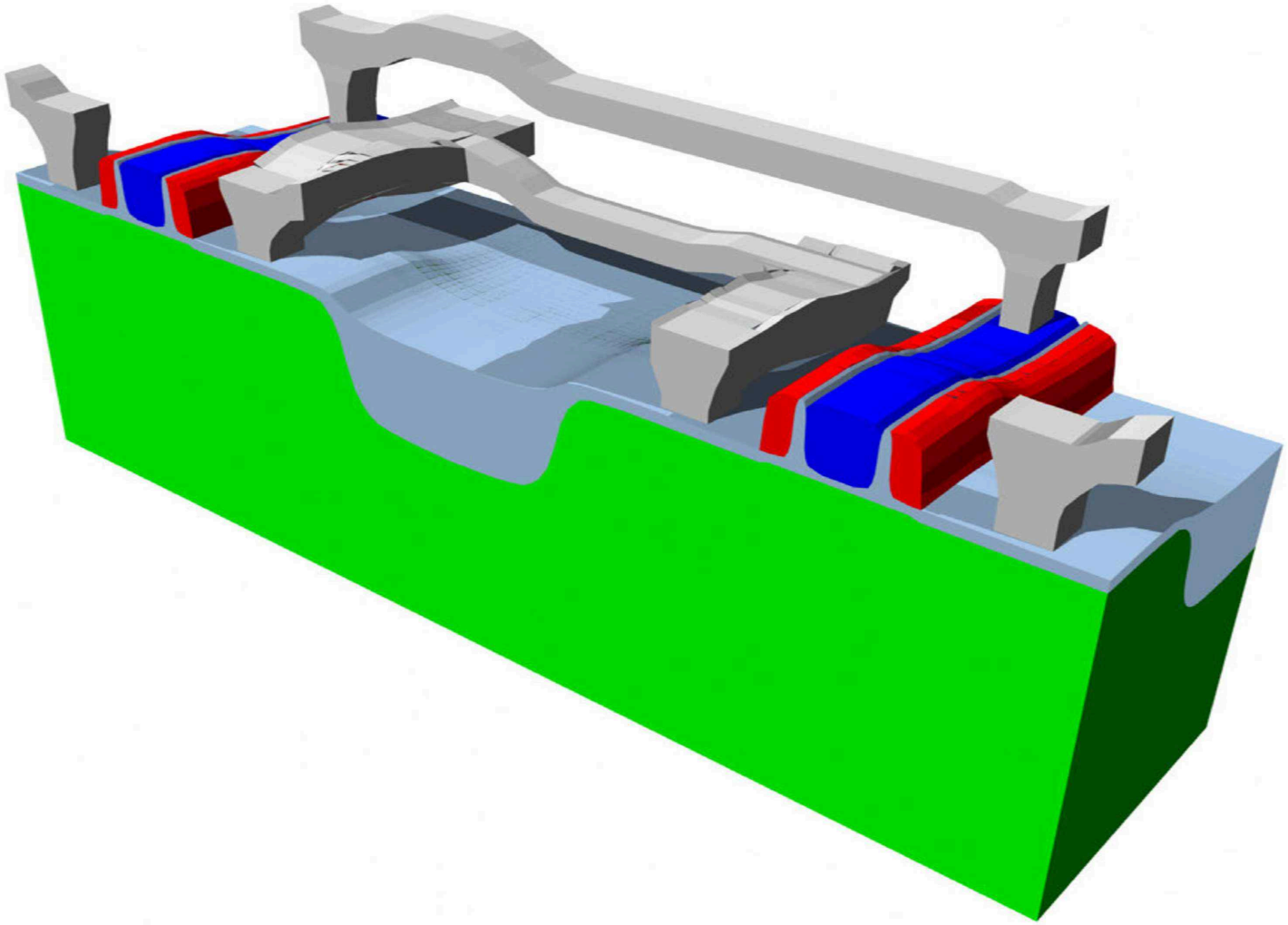


Figure 7.24: Typical CMOS inverter structure with two transistors.

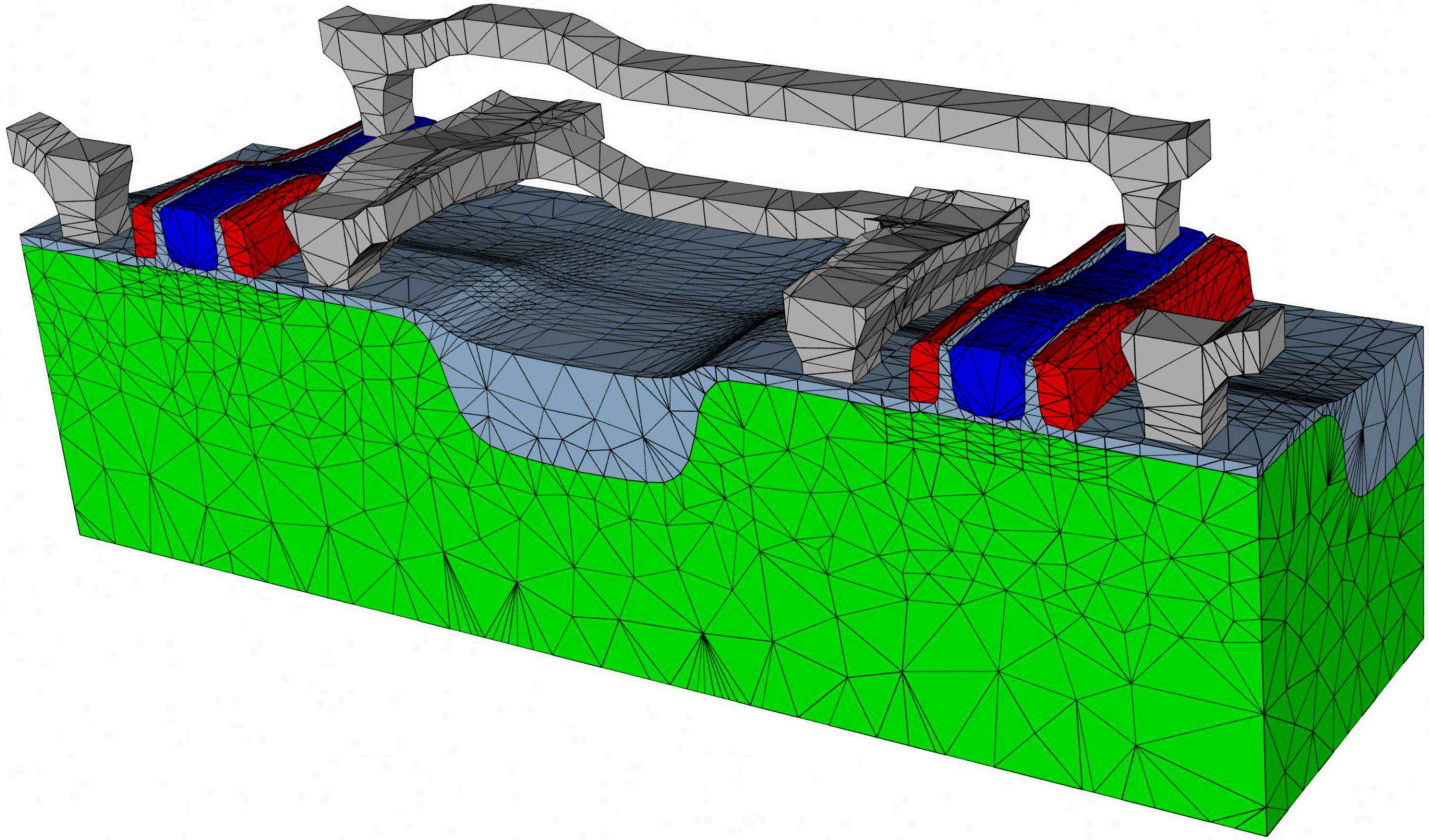


Figure 7.25: Initial coarse mesh of the CMOS device.

Chapter 8

Outlook

The present state of the meshing tool forms a foundation for the future to cope with coming issues and already important aspects of mesh generation for semiconductor process and device simulation. Further development is required from a scientific as well as software engineering point of view. The chosen approach — the type of Delaunay kernel algorithm and the a priori boundary integration — seem promising to meet the demands. The ultimate goal is a robust tetrahedralization engine which satisfies the special boundary triangle requirements of a Voronoi type box integration and which handles extremely thin oxide layers without compromising the mesh in the silicon bulk region. The engine must be versatile to allow a combination with special techniques for semiconductor device simulation, as for example elliptical grid generation techniques. Ideally, the engine is to some extent itself capable to produce boundary-fitted meshes at e.g. the channel of a semiconductor device. For none-Voronoi type box integration the strict boundary triangle requirements can be disabled to not become a limitation of the mesh generator and to utilize the efficiency of a Delaunay kernel for the generation of hybrid, prismatic meshes or generally meshes where not all elements possess the Delaunay property. This is especially important for the optimization of meshes with regard to slivers for finite element computations. In summary the following are important further steps:

- Continuing the research and the improvements of the Delaunay surface preprocessor especially with regard to thin layers.
- Investigation of various point generation methods such as advancing front style methods for boundary-fitted meshes and a better control of the local mesh density. A generation of hybrid tetrahedral/prismatic meshes might be an important step.
- Utilizing the already implemented tetrahedral refinement functions to test various types of Steiner Triangulations.
- Implementation of additional element quality measures like aspect ratio and minimum dihedral angle of a tetrahedron also with regard to anisotropy.
- Other software engineering aspects as for example the enhancement of the existing library interface to improve the link between the application and the mesh generator. An implementation of the control space is required for mesh adaptation.

Bibliography

[1] D. Adalsteinsson and J.A. Sethian. A Fast Level Set Method for Propagating Interfaces. *J.Comput.Phys.*, 118(1):269–277, 1995.

[2] M.J. Aftosmis, J.E. Melton, and M.J. Berger. Adaptation and Surface Modeling for Cartesian Mesh Methods. In *12th AIAA Computational Fluid Dynamics Conference*, number 95-1725-CP, San Diego, USA, 1995.

[3] C.G. Armstrong, D.J. Robinson, R.M. McKeag, T.S. Li, S.J. Bridgett, and R.J. Donaghy. Applications of the Medial Axis Transform in Analysis Modelling. In *Proc. of the 5th International Conference on Reliability of Finite Element Methods for Engineering Applications*, pages 415–426, 1995.

[4] V. Axelrad. Grid Quality and Its Influence on Accuracy and Convergence in Device Simulation. *IEEE Trans.Computer-Aided Design of Integrated Circuits and Systems*, 17(2):149–157, 1998.

[5] I. Babuska and A.K. Aziz. On the Angle Condition in the Finite Element Method. *SIAM J.Numer.Anal.*, 13(2):214–226, 1976.

[6] M. Bächtold, M. Emmenegger, J.G. Korvink, and H. Baltes. An Error Indicator and Automatic Adaptive Meshing for Electrostatic Boundary Element Simulations. *IEEE Trans.Computer-Aided Design of Integrated Circuits and Systems*, 16(12):1439–1446, 1997.

[7] B.S. Baker, E. Grosse, and C.S. Rafferty. Nonobtuse Triangulation of Polygons. *Discrete & Computational Geometry*, 3(2):147–168, 1988.

[8] R.E. Bank. *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations*, volume 7 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 1990. Users' Guide 6.0.

[9] R.E. Bank and R.K. Smith. Mesh Smoothing Using a Posteriori Error Estimates. *SIAM J.Numer.Anal.*, 34(3):979–997, 1997.

[10] R.E. Bank and A. Weiser. Some A Posteriori Error Estimators for Elliptic Partial Differential Equations. *Math.Comp.*, 44(170):283–301, 1985.

[11] B. Barber. *Computational Geometry with Imprecise Data and Arithmetic*. PhD thesis, Computer Science Department, Princeton University, 1992. Available as Techn.Rep. CS-TR-377-92.

[12] B. Barber, D.P. Dobkin, and H.T. Huhdanpaa. The Quickhull Algorithm for Convex Hull. Technical Report GCG53, The Geometry Center, University of Minnesota, Minneapolis, Minnesota, USA, 1993. `http://www.geom.umn.edu/ bradb/qhull-news.html`.

[13] R. Bauer. *Numerische Berechnung von Kapazitäten in dreidimensionalen Verdrahtungsstrukturen.* Dissertation, Technische Universität Wien, 1994.

[14] M. Bern. Compatible Tetrahedralizations. In *Proc. 9th Annual Symposium on Computational Geometry*, pages 281–288, San Diego, USA, 1993. ACM.

[15] M. Bern and D. Eppstein. Mesh Generation and Optimal Triangulation. In F.K. Hwang and D.-Z. Du, editors, *Computing in Euclidean Geometry.* World Scientific, 1992.

[16] M. Bern, D. Eppstein, and J.R. Gilbert. Provably Good Mesh Generation. In *Proc. 31th Annual Symposium on Foundations of Computer Science*, pages 231–241. IEEE, 1990.

[17] M. Berzins. A Solution-Based Triangular and Tetrahedral Mesh Quality Indicator. *SIAM J.Sci.Comput.*, 19(6):2051–2060, 1998.

[18] T. Binder, K. Dragosits, T. Grasser, R. Klima, M. Knaipp, H. Kosina, R. Mlekus, V. Palankovski, M. Rottinger, G. Schrom, S. Selberherr, and M. Stockinger. *MINIMOS-NT User's Guide.* Institut für Mikroelektronik, 1998.

[19] T.D. Blacker and M.B. Stephenson. Paving: A New Approach to Automated Quadrilateral Mesh Generation. *Int.J.Numer.Meth.Eng.*, 32(4):811–847, 1991.

[20] F.J. Bossen and P.S. Heckbert. A Pliant Method for Anisotropic Mesh Generation. In IMRT'96 [144], pages 63–74.

[21] A. Bowyer. Computing Dirichlet Tessellations. *The Computer Journal*, 24(2):162–166, 1981.

[22] E.K. Buratynski. A Fully Automatic Three-Dimensional Mesh Generator for Complex Geometries. *Int.J.Numer.Meth.Eng.*, 30:931–952, 1990.

[23] G. Butlin and C. Stops. CAD Data Repair. In IMRT'96 [144], pages 7–12.

[24] J.E. Castillo. *Mathematical Aspects of Numerical Grid Generation.* SIAM, Philadelphia, 1991.

[25] J.C. Cavendish, D.A. Field, and W.H. Frey. An Approach To Automatic Three-Dimensional Finite Element Mesh Generation. *Int.J.Numer.Meth.Eng.*, 21:329–347, 1985.

[26] Johann Cervenka. CGG: Ein Gittergenerator für die Bauelementesimulation. Diplomarbeit, Technische Universität Wien, 1999.

[27] T. Chen, D.W. Yergeau, and R.W. Dutton. A Common Mesh Implementation for Both Static and Moving Boundary Process Simulations. In Meyer and Biesemans [107], pages 101–104.

[28] L.P. Chew. Guaranteed-Quality Triangular Meshes. Technical Report TR-89-983, Cornell University, 1989.

[29] L.P. Chew. Guaranteed-Quality Mesh Generation for Curved Surfaces. In *Proc. 9th Annual Symposium on Computational Geometry*, pages 274–280, San Diego, USA, 1993. ACM.

[30] L.P. Chew. Guaranteed-Quality Delaunay Meshing in 3D. In *Proc. 13th Annual Symposium on Computational Geometry*, pages 391–393. ACM, 1997.

[31] P. Cignoni, D. Laforenza, C. Montani, R. Perego, and R. Scopigno. Evaluation of Parallelization Strategies for an Incremental Delaunay Triangulator in E^3. *Concurrency: Practice and Experience*, 7(1):61–80, 1995.

[32] P. Cignoni, C. Montani, R. Perego, and R. Scopigno. Parallel 3D Delaunay Triangulation. In *EUROGRAPHICS*, volume 12, pages C–129. Blackwell Publishers, 1993.

[33] P. Cignoni, C. Montani, and R. Scopigno. DeWall: a Fast Divide & Conquer Delaunay Triangulation Algorithm in E^d. *Computer-Aided Design*, 30(5):333–341, 1998.

[34] B.A. Cipra. A Rapid-Development Force for CFD: Cartesian Grids. *SIAM News*, 28(10):1–2, 1995.

[35] P. Conti. *Grid Generation for Three-Dimensional Semiconductor Device Simulation.* Hartung-Gorre, 1991.

[36] Boris N. Delaunay. Sur la Sphère Vide. *Izvestia Akademia Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7:793–800, 1934.

[37] T.K. Dey, C.L. Bajaj, and K. Sugihara. On Good Triangulations in Three Dimensions. In *Proc. of the ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications.* ACM, 1991.

[38] R.J. Donaghy, W. McCune, S.J. Bridgett, C.G. Armstrong, D.J. Robinson, and R.M. McKeag. Dimensional Reduction of Analysis Models. In IMRT'96 [144], pages 307–320.

[39] R.A. Dwyer. Higher-Dimensional Voronoi Diagrams in Linear Expected Time. *Discrete & Computational Geometry*, 6:343–367, 1991.

[40] H. Edelsbrunner, F.P. Preparata, and D.B. West. Tetrahedrizing Point Sets in Three Dimensions. *J. Symbolic Computation*, 10:335–347, 1990.

[41] H. Edelsbrunner, T.S. Tan, and R. Waupotitsch. An $O(n^2 \log n)$ Time Algorithm for the Minmax Angle Triangulation. *SIAM J.Sci.Stat.Comput.*, 13(4):994–1008, 1992.

[42] T. Fang and L.A. Piegl. Delaunay Triangulation in Three Dimensions. *IEEE Computer Graphics and Applications*, 15(3):62–69, 1995.

[43] T.P. Fang and L.A. Piegl. Delaunay Triangulation Using a Uniform Grid. *IEEE Computer Graphics and Applications*, 13(3):36–47, 1993.

[44] C. Fischer, P. Habaš, O. Heinreichsberger, H. Kosina, Ph. Lindorfer, P. Pichler, H. Pötzl, C. Sala, A. Schütz, S. Selberherr, M. Stiftinger, and M. Thurner. *MINIMOS 6 User's Guide*. Institut für Mikroelektronik, Technische Universität Wien, Austria, 1994.

[45] P. Fleischmann, R. Sabelka, A. Stach, R. Strasser, and S. Selberherr. Grid Generation for Three Dimensional Process and Device Simulation. In SISPAD'96 [166], pages 161–166.

[46] S. Fortune. A Sweepline Algorithm for Voronoi Diagrams. *Algorithmica*, 2(2):153–174, 1987.

[47] S. Fortune. Voronoi Diagrams and Delaunay Triangulations. In F.K. Hwang and D.-Z. Du, editors, *Computing in Euclidean Geometry*, pages 193–233. World Scientific, Singapore, 1992.

[48] S. Fortune and C.J. Van Wyk. Efficient Exact Arithmetic for Computational Geometry. In *Proc. 9th Annual Symposium on Computational Geometry*, pages 163–172, San Diego, USA, 1993. ACM.

[49] L.A. Freitag and C. Ollivier-Gooch. A Comparison of Tetrahedral Mesh Improvement Techniques. In IMRT'96 [144], pages 87–100.

[50] P.J. Frey, H. Borouchaki, and P.L. George. Delaunay Tetrahedralization Using an Advancing-Front Approach. In IMRT'96 [144], pages 31–43.

[51] R.S. Gallagher, editor. *Computer Visualization*. CRC Press, 1995.

[52] G. Garretón. *A Hybrid Approach to 2D and 3D Mesh Generation for Semiconductor Device Simulation*. Dissertation, ETH Zürich, 1999. Hartung-Gorre.

[53] G. Garretón, L. Villablanca, N. Strecker, and W. Fichtner. Unified Grid Generation and Adaptation for Device Simulation. In Ryssel and Pichler [138], pages 468–471.

[54] G. Garretón, L. Villablanca, N. Strecker, and W. Fichtner. A Hybrid Approach for Building 2D and 3D Conforming Delaunay Meshes Suitable for Process and Device Simulation. In Meyer and Biesemans [107], pages 185–188.

[55] P.L. George. Automatic Mesh Generation and Finite Element Computation. In P.G. Ciarlet and J.L. Lions, editors, *Handbook of Numerical Analysis*, volume IV. Elsevier, 1996.

[56] P.L. George, F. Hecht, and E. Saltel. Automatic Mesh Generator with Specified Boundary. *Computer Methods in Applied Mechanics and Engineering*, 92:269–288, 1991.

[57] C.S. Gitlin and C.R. Johnson. A Tool for Exploring 3D Unstructured Tetrahedral Meshes. In IMRT'96 [144], pages 333–345.

[58] H.K. Gummel. A Self-Consistent Iterative Scheme for One-Dimensional Steady State Transistor Calculations. *IEEE Trans.Electron Devices*, ED-11:455–465, 1964.

[59] S. Halama. *The Viennese Integrated System for Technology CAD Applications–Architecture and Critical Software Components.* Dissertation, Technische Universität Wien, 1994.

[60] J. Häuser and C. Taylor. *Numerical Grid Generation in Computational Fluid Dynamics.* Pineridge Press, 1986.

[61] W.D. Henshaw. Automatic Grid Generation. In A. Iserles, editor, *Acta Numerica*, volume 5, pages 121–148. Cambridge University Press, 1996.

[62] L.R. Hermann. Laplacian-Isoparametric Grid Generation Scheme. *J. of the Engineering Mechanics Division of the American Society of Civil Engineers*, 102:749–756, 1976.

[63] F. Hermeline. Triangulation Automatique d'un Polyèdre en Dimension N. *RAIRO Analyse Numérique*, 16(3):211–242, 1982.

[64] N. Hitschfeld. *Grid Generation for Three-Dimensional Non-Rectangular Semiconductor Devices.* Hartung-Gorre, 1993.

[65] N. Hitschfeld, P. Conti, and W. Fichtner. Mixed Element Trees: A Generalization of Modified Octrees for the Generation of Meshes for the Simulation of Complex 3-D Semiconductor Device Structures. *IEEE Trans.Computer-Aided Design*, 12(11):1714–1725, 1993.

[66] N. Hitschfeld and W. Fichtner. 3D Grid Generation for Semiconductor Devices Using a Fully Flexible Refinement Approach. In Selberherr et al. [157], pages 413–416.

[67] N. Hitschfeld, M.C. Rivara, and M. Palma. Improving the Quality of Delaunay Triangulations for the Control Volume Discretization Method. In Meyer and Biesemans [107], pages 189–192.

[68] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh Optimization. In *Computer Graphics, SIGGRAPH'93 Proceedings*, pages 19–26, 1993.

[69] T. Ikeda. *Maximum Principle in Finite Element Models for Convection-Diffusion Phenomena.* North Holland, Amsterdam, 1983.

[70] B. Joe. Delaunay Triangular Meshes in Convex Polygons. *SIAM J.Sci.Stat.Comput.*, 7(2):514–539, 1986.

[71] B. Joe. Three-Dimensional Triangulations from Local Transformations. *SIAM J.Sci.Stat.Comput.*, 10(4):718–741, 1989.

[72] B. Joe. Construction of Three-Dimensional Delaunay Triangulations Using Local Transformations. *Computer Aided Geometric Design*, 8:123–142, 1991.

[73] B. Joe. Delaunay versus Max-Min Solid Angle Triangulations for Three-Dimensional Mesh Generation. *Int.J.Numer.Meth.Eng.*, 31:987–997, 1991.

[74] B. Joe. GEOMPACK - A Software Package for the Generation of Meshes using Geometric Algorithms. *Advanced Engineering Software*, 13(5/6):325–331, 1991.

[75] B. Joe. Construction of Three-Dimensional Improved-Quality Triangulations Using Local Transformations. *SIAM J.Sci.Comput.*, 16(6):1292–1307, 1995.

[76] A.A. Johnson and T.E. Tezduyar. Mesh Generation and Update Strategies for Parallel Computation of 3D Flow Problems. In S.N. Atluri, G. Yagawa, and T.A. Cruse, editors, *Proc. of the International Conference on Computational Engineering Science*, Hawaii, USA, 1995.

[77] Y. Kallinderis, A. Khawaja, and H. McMorris. Hybrid Prismatic/Tetrahedral Grids for Turbomaschinery Applications. In *6th International Meshing Roundtable*, pages 21–31, Park City, Utah, 1997. Sandia National Labs.

[78] P. Knupp and S. Steinberg. *Fundamentals of Grid Generation.* CRC Press, 1993.

[79] N. Kotani. TCAD in Selete. In Meyer and Biesemans [107], pages 3–7.

[80] M. Krizek. On the Maximum Angle Condition for Linear Tetrahedral Elements. *SIAM J.Numer.Anal.*, 29:513–520, 1992.

[81] P. Krysl and M. Ortiz. Generation of Tetrahedral Finite Element Meshes: Variational Delaunay Approach. In *7th International Meshing Roundtable*, pages 272–284, Dearborn, Michigan, 1998. Sandia National Labs.

[82] S. Kumashiro and I. Yokota. A Triangular Mesh Generation Method Suitable for the Analysis of Complex MOS Device Structures. In *Int. Workshop on Numerical Modeling of Processes and Devices for Integrated Circuits NUPAD V*, pages 167–170, Honolulu, 1994.

[83] M.E. Law. The Virtual IC Factory ... can it be achieved? *IEEE Circuits & Devices*, pages 25–31, 1995.

[84] C.L. Lawson. Software for C^1 Surface Interpolation. In J.R. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, New York, 1977.

[85] C. Ledl. Konvertierung rasterorienter Geometrien in polygonal begrenzte. Diplomarbeit, Technische Universität Wien, 1994.

[86] D.T. Lee and B.J. Schachter. Two Algorithms for Constructing a Delaunay Triangulation. *International Journal of Computer and Information Sciences*, 9(3):219–242, 1980.

[87] E. Leitner, W. Bohmayr, P. Fleischmann, E. Strasser, and S. Selberherr. 3D TCAD at TU Vienna. In J. Lorenz, editor, *3-Dimensional Process Simulation*, pages 136–161, Wien, 1995. Springer.

[88] E. Leitner and S. Selberherr. Three-Dimensional Grid Adaptation Using a Mixed-Element Decomposition Method. In Ryssel and Pichler [138], pages 464–467.

[89] F.W. Letniowski. Three-Dimensional Delaunay Triangulations for Finite Element Approximations to a Second-Order Diffusion Operator. *SIAM J.Sci.Stat.Comput.*, 13(3):765–770, 1992.

[90] K. Lilja, V. Moroz, and D. Wake. A 3D Mesh Generation Method for the Simulation of Semiconductor Processes and Devices. In MSM'98 [113], pages 334–338.

[91] J. Litsios, B. Schmithüsen, U. Krumbein, A. Schenk, E. Lyumkis, B. Polsky, and W. Fichtner. *DESSIS 3.0: Manual.* ISE Integrated Systems Engineering, Zürich, Switzerland, 1996. release 3.0 edition.

[92] A. Liu and B. Joe. On the Shape of Tetrahedra from Bisection. *Mathematics of Computation*, 63(207):141–154, 1994.

[93] A. Liu and B. Joe. Quality Local Refinement of Tetrahedral Meshes Based on Bisection. *SIAM J.Sci.Comput.*, 16(6):1269–1291, 1995.

[94] R. Löhner and P. Parikh. Three-Dimensional Grid Generation by the Advancing Front Method. *Int.J.Numer.Meths.Fluids.*, 8:1135–1149, 1988.

[95] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[96] R.H. Macneal. An Asymmetrical Finite Difference Network. *Quarterly Applied Mathematics*, 11(3):295–310, 1952.

[97] M. Mäntylä. *An Introduction to Solid Modeling.* Computer Science Press, Rockville, 1988.

[98] MARC Analysis Research Corp., Palo Alto, USA. *Automatic Mesh Generation in Mentat II*, 1995.

[99] P.V. Marcal. Constructive Solid Geometry, the CAD-FEM Connection. In IMRT'96 [144], pages 157–168.

[100] D.L. Marcum and N.P. Weatherill. Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection. In *12th AIAA Applied Aerodynamics Conference*, number 94-1926, Colorado Springs, USA, 1994.

[101] R. Martins, R. Sabelka, W. Pyka, and S. Selberherr. Rigorous Capacitance Simulation of DRAM Cells. In Meyer and Biesemans [107], pages 69–72.

[102] M. Masquelier, D. George, and A. Kuprat. Unstructured 3D Grid Toolbox for Modeling and Simulation. In MSM'98 [113], page M3.4.3. Book of abstracts, MSM'98.

[103] C.W. Mastin and J.F. Thompson. Quasiconformal Mappings and Grid Generation. *SIAM J.Sci.Stat.Comput.*, 5(2):305–316, 1984.

[104] D.J. Mavriplis. An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness. *J.Comput.Phys.*, 117:90–101, 1995.

[105] J.E. Melton, F.Y. Enomoto, and M.J. Berger. 3D Automatic Cartesian Grid Generation for Euler Flows. *AIAA*, (93-3386-CP), 1993.

[106] M.L. Merriam. An Efficient Advancing Front Algorithm for Delaunay Triangulation. In *AIAA 29th Aerospace Sciences Meeting*, number 91-0792, Reno, USA, 1993.

[107] K. De Meyer and S. Biesemans, editors. *Simulation of Semiconductor Processes and Devices.* Springer, Wien, New York, 1998.

[108] G.L. Miller, D. Talmor, S. Teng, N. Walkington, and H. Wang. Control Volume Meshes Using Sphere Packing. In IMRT'96 [144], pages 47–62.

[109] S.A. Mitchell and S.A. Vavasis. Quality Mesh Generation in Three Dimensions. Technical Report TR-92-1267, Cornell University, 1992. `http://www.cs.cornell.edu/home/vavasis/qmg1.1`.

[110] R. Mlekus and S. Selberherr. Polygonal Geometry Reconstruction after Cellular Etching or Deposition Simulation. In Ryssel and Pichler [138], pages 50–53.

[111] R.H. Möhring and M. Müller-Hannemann. Mesh Refinement via Bidirected Flows: Modeling, Complexity, and Computational Results. *J.ACM*, 44(3):395–426, 1997.

[112] V. Moroz, S. Motzny, and K. Lilja. A Boundary Conforming Mesh Generation Algorithm for Simulation of Devices with Complex Geometry. In SISPAD'97 [167], pages 293–295.

[113] *International Conference on Modeling and Simulation of Microsystems Semiconductors, Sensors and Actuators*, Santa Clara, CA, USA, 1998.

[114] E.P. Mücke. *Shapes and Implementations in Three-Dimensional Geometry.* PhD thesis, Computer Science Department, University of Illinois at Urbana-Champaign, 1993.

[115] E.P. Mücke. A Robust Implementation for Three-Dimensional Delaunay Triangulations. *Int. Journal of Computational Geometry & Applications*, 8(2):255–276, 1998.

[116] S. Müller, K. Kells, and W. Fichtner. Automatic Rectangle-Based Adaptive Mesh Generation Without Obtuse Angles. *IEEE Trans.Computer-Aided Design*, 11(7):855–863, 1992.

[117] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations – Concepts and Applications of Voronoi Diagrams.* Wiley, 1992.

[118] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-Independent Modeling with Simplicial Complexes. *ACM Trans.Graphics*, 12(1):56–102, 1993.

[119] A.L. Pardhanani and G.F. Carey. A Mapped Scharfetter-Gummel Formulation for the Efficient Simulation of Semiconductor Device Models. *IEEE Trans.Computer-Aided Design of Integrated Circuits and Systems*, 16(10):1227–1233, 1997.

[120] J.W. Peterson. Tessellation of NURB Surfaces. In P.S. Heckbert, editor, *Graphics Gems IV*, pages 286–320. Academic Press, 1994.

[121] Ch. Pichler, R. Plasun, R. Strasser, and S. Selberherr. High-Level TCAD Task Representation and Automation. *IEEE J.Technology Computer Aided Design*, 1997. `http://www.ieee.org/journal/tcad/accepted/pichler-may97/`.

[122] L.A. Piegl and A.M. Richard. Algorithm and Data Structure for Triangulating Multiply Connected Polygonal Domains. *Comput. & Graphics*, 17(5):563–574, 1993. Pergamon Press.

[123] F.P. Preparata and M.I. Shamos. *Computational Geometry.* Springer, 1985.

[124] M. Price, C. Stops, and G. Butlin. A Medial Object Toolkit For Meshing and Other Applications. In IMRT'95 [143], pages 219–229.

[125] M. Putti and Ch. Cordes. Finite Element Approximation of the Diffusion Operator on Tetrahedra. *SIAM J.Sci.Comput.*, 19(4):1154–1168, 1998.

[126] W. Pyka, P. Fleischmann, B. Haindl, and S. Selberherr. Linking Three-Dimensional Topography Simulation with High Pressure CVD Reaction Kinetics. In *International Conference on Simulation of Semiconductor Processes and Devices*, pages 199–202, Kyoto, Japan, 1999. Business Center for Academic Societies Japan.

[127] M. Radi. *Three-Dimensional Simulation of Thermal Oxidation.* Dissertation, Technische Universität Wien, 1998.

[128] M. Radi, E. Leitner, E. Hollensteiner, and S. Selberherr. AMIGOS: Analytical Model Interface & General Object-Oriented Solver. In SISPAD'97 [167], pages 331–334.

[129] M. Radi and S. Selberherr. Three-Dimensional Adaptive Mesh Relaxation. In Meyer and Biesemans [107], pages 193–196.

[130] V.T. Rajan. Optimality of the Delaunay Triangulation in R^d. In *Proc. 7th Annual Symposium on Computational Geometry*, pages 357–363. ACM, 1991.

[131] M.C. Rivara. Mesh Refinement Processes Based on the Generalized Bisection of Simplices. *SIAM J.Numer.Anal.*, 21(3):604–613, 1984.

[132] M.C. Rivara and P. Inostroza. A Discussion on Mixed (Longest Side Midpoint Insertion) Delaunay Techniques for the Triangulation Refinement Problem. In IMRT'95 [143], pages 335–346.

[133] M. Rottinger. Generation of Triangular Grids in MINIMOS-NT. Annual review, Institute for Microelectronics, Vienna, Austria, 1997. `http://www.iue.tuwien.ac.at/reviews/1997/rottinger_text.html`.

[134] J. Ruppert. *Results on Triangulation and High Quality Mesh Generation.* PhD thesis, University of California at Berkeley, 1992.

[135] J. Ruppert. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms*, 18:548–585, 1995.

[136] J. Ruppert and R. Seidel. On the Difficulty of Tetrahedralizing 3-Dimensional Non-Convex Polyhedra. In *Proc. 5th Annual Symposium on Computational Geometry*, pages 380–393. ACM, 1989.

[137] J. Ruppert and R. Seidel. On the Difficulty of Triangulating Three-Dimensional Non-Convex Polyhedra. *Discrete & Computational Geometry*, 7:227–253, 1992.

[138] H. Ryssel and P. Pichler, editors. *Simulation of Semiconductor Devices and Processes*, volume 6, Wien, 1995. Springer.

[139] A. Saalfeld. Delaunay Edge Refinements. In *Proc. 3rd Canadian Conf. Comp. Geometry*, pages 33–36, 1991.

[140] R. Sabelka, R. Martins, and S. Selberherr. Accurate Layout-Based Interconnect Analysis. In Meyer and Biesemans [107], pages 336–339.

[141] Z.H. Sahul, R.W. Dutton, and M. Noell. Grid and Geometry Techniques for Multi-Layer Process Simulation. In Selberherr et al. [157], pages 417–420.

[142] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[143] Sandia National Labs. *4th International Meshing Roundtable*, Albuquerque, New Mexico, 1995.

[144] Sandia National Labs. *5th International Meshing Roundtable*, Pittsburgh, Pennsylvania, 1996.

[145] N. Sapidis and R. Perucchio. Delaunay Triangulation of Arbitrarily Shaped Planar Domains. *Computer Aided Geometric Design*, 8:421–437, 1991.

[146] D.L. Scharfetter and H.K. Gummel. Large-Signal Analysis of a Silicon Read Diode Oscillator. *IEEE Trans.Electron Devices*, ED-16(1):64–77, 1969.

[147] R. Schneiders. Information on Finite Element Mesh Generation. publ. on the WWW. `http://www-users.informatik.rwth-aachen.de/~roberts/meshgeneration.html`.

[148] R. Schneiders, R. Schindler, and F. Weiler. Octree-based Generation of Hexahedral Element Meshes. In IMRT'96 [144], pages 205–215.

[149] E. Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Mathematische Annalen*, 98:309–312, 1928.

[150] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics.* Prentice-Hall, 1996.

[151] W.J. Schroeder and M.S. Shephard. Geometry Based Fully Automatic Mesh Generation and the Delaunay Triangulation. *Int.J.Numer.Meth.Eng.*, 26:2503–2515, 1988.

[152] W.J. Schroeder and M.S. Shephard. A Combined Octree/Delaunay Method for Fully Automatic 3-D Mesh Generation. *Int.J.Numer.Meth.Eng.*, 29:37–55, 1990.

[153] W.J. Schroeder, J.A. Zarge, and W.E. Lorenson. Decimation of Triangle Meshes. *Computer Graphics*, 26(2):65–70, 1992.

[154] H.R. Schwarz. *Methode der finiten Elemente.* Teubner, 1980.

[155] R. Seidel. Constrained Delaunay Triangulations and Voronoi Diagrams with Obstacles. Technical Report 260, Inst. for Information Processing, Graz, Austria, 1988.

[156] S. Selberherr. *Analysis and Simulation of Semiconductor Devices.* Springer, Wien, 1984.

[157] S. Selberherr, H. Stippel, and E. Strasser, editors. *Simulation of Semiconductor Devices and Processes*, volume 5. Springer, 1993.

[158] M. Sever. Delaunay Partitioning in Three Dimensions and Semiconductor Models. *COMPEL*, 5(2):75–93, 1986.

[159] M.S. Shephard and M.K. Georges. Automatic Three-Dimensional Mesh Generation by the Finite Octree Technique. *Int.J.Numer.Meth.Eng.*, 32:709–749, 1991.

[160] J.R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *First Workshop on Applied Computational Geometry*, pages 124–133, Philadelphia, 1996. ACM.

[161] J.R. Shewchuk. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, 18(3):305–363, 1997.

[162] J.R. Shewchuk. *Delaunay Refinement Mesh Generation.* PhD thesis, Computer Science Department, Carnegie Mellon University, 1997. Available as Techn.Rep. CMU-CS-97-137.

[163] J.R. Shewchuk. A Condition Guaranteeing the Existence of Higher-Dimensional Constrained Delaunay Triangulations. In *Proc. 14th Annual Symposium on Computational Geometry*, pages 76–85. ACM, 1998.

[164] J.R. Shewchuk. Tetrahedral Mesh Generation by Delaunay Refinement. In *Proc. 14th Annual Symposium on Computational Geometry*, pages 86–95. ACM, 1998.

[165] N. Shigyo, T. Wada, and S. Yasuda. Discretization Problem for Multidimensional Current Flow. *IEEE Trans.Computer-Aided Design*, 8(10):1046–1050, 1989.

[166] *International Conference on Simulation of Semiconductor Processes and Devices*, Tokyo, Japan, 1996. Business Center for Academic Societies Japan.

[167] *International Conference on Simulation of Semiconductor Processes and Devices*, Cambridge, Massachusetts, 1997.

[168] S.P. Spekreijse. Elliptic Grid Generation Based on Laplace Equations and Algebraic Transformations. *J.Comput.Phys.*, 118:38–61, 1995.

[169] V. Srinivasan, L. Nackman, J. Tang, and S. Meshkat. Automatic Mesh Generation Using the Symmetric Axis Transformation of Polygonal Domains. *Proc.IEEE*, 80(9):1485–1501, 1992.

[170] E. Strasser and S. Selberherr. Algorithms and Models for Cellular Based Topography Simulation. *IEEE Trans.Computer-Aided Design*, 14(9):1104–1114, 1995.

[171] R. Strasser. Multigrid Methods in 2D-Process Simulation. Diplomarbeit, Technische Universität Wien, 1995.

[172] R. Strasser, Ch. Pichler, and S. Selberherr. VISTA - A Framework for Technology CAD Purposes. In W. Hahn and A. Lehmann, editors, *9th European Simulation Symposium*, pages 450–454, Budapest, Hungary, 1997. Society for Computer Simulation International.

[173] R. Strasser and S. Selberherr. Parallel and Distributed TCAD Simulations using Dynamic Load Balancing. In Meyer and Biesemans [107], pages 89–92.

[174] P. Su and L.S. Drysdale. A Comparison of Sequential Delaunay Triangulation Algorithms. In *Proc. 11th Annual Symposium on Computational Geometry*, pages 61–70, Vancouver, CANADA, 1995. ACM.

[175] K. Sugihara. An Intersection Algorithm Based on Delaunay Triangulation. *IEEE Computer Graphics and Applications*, 12(2):59–67, 1992.

[176] K. Sugihara and M. Iri. Construction of the Voronoi Diagram for One Million Generators in Single-Precision Arithmetic. *Proc.IEEE*, 80(9):1471–1484, 1992.

[177] T. Syo, Y. Akiyama, S. Kumashiro, I. Yokota, and S. Asada. A Triangular Mesh with the Interface Protection Layer Suitable for the Diffusion Simulation. In SISPAD'96 [166], pages 173–174.

[178] K. Tanaka, A. Notsu, and H. Matsumoto. A New Approach to Mesh Generation for Complex 3D Semiconductor Device Structures. In SISPAD'96 [166], pages 167–168.

[179] M. Tannemura, T. Ogawa, and N. Ogita. A New Algorithm for Three-Dimensional Voronoi Tessellation. *J.Comput.Phys.*, 51(2):191–207, 1983.

[180] T.J. Tautges and S.A. Mitchell. Whisker Weaving: Invalid Connectivity Resolution and Primal Construction Algorithm. In IMRT'95 [143], pages 115–127.

[181] Technology Modeling Associates, Inc., Sunnyvale, California. *TMA Medici, Two-Dimensional Device Simulation Program, Version 4.0 User's Manual*, 1997.

[182] Technology Modeling Associates, Inc., Sunnyvale, California. *TMA WorkBench Version 2.2 User's Manual*, 1997.

[183] J.F. Thompson. A General Three-Dimensional Elliptic Grid Generation System on a Composite Block Structure. *Computer Methods in Applied Mechanics and Engineering*, 64:377–411, 1987.

[184] J.F. Thompson, Z.U.A. Warsi, and C.W. Mastin. *Numerical Grid Generation.* North Holland, 1985.

[185] K. Tietzel, A. Bourenkov, and J. Lorenz. Coupled 3D Process and Device Simulation of Advanced MOSFETs. In Meyer and Biesemans [107], pages 255–258.

[186] D.F. Watson. Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes. *The Computer Journal*, 24(2):167–172, 1981.

[187] N.P. Weatherill and O. Hassan. Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints. *Int.J.Numer.Meth.Eng.*, 37:2005–2039, 1994.

[188] M. Westermann, N. Strecker, P. Regli, and W. Fichtner. Reliable Solid Modeling for Three-Dimensional Semiconductor Process and Device Simulation. In *Int. Workshop on Numerical Modeling of Processes and Devices for Integrated Circuits NUPAD V*, pages 49–52, Honolulu, 1994.

[189] J. Xu and L. Zikatanov. A Monotone Finite Element Scheme for Convection-Diffusion Equations. *Mathematics of Computation*, 68(228):1429–1446, 1999.

[190] A. Yamada, K. Shimada, and T. Itoh. Energy-Minimizing Approach to Meshing Curved Wire-Frame Models. In IMRT'96 [144], pages 179–191.

[191] M.A. Yerry and M.S. Shephard. Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique. *Int.J.Numer.Meth.Eng.*, 20:1965–1990, 1984.

[192] Gernot Zankl. Über die Erzeugung von Gitterstützpunkten in dreidimensionalen Geometrien. Diplomarbeit, Technische Universität Wien, 1997.

[193] O.C. Zienkiewicz. *The Finite Element Method.* McGraw-Hill, 1977.

[194] O.C. Zienkiewicz. Origins, Milestones and Directions of the Finite Element Method — A Personal View. In P.G. Ciarlet and J.L. Lions, editors, *Handbook of Numerical Analysis*, volume IV. Elsevier, 1996.

List of Figures

Publications

[P13] P. Fleischmann, R. Kosik, B. Haindl, and S. Selberherr. Simple Mesh Examples to Illustrate Specific Finite Element Mesh Requirements. In *8th International Meshing Roundtable*, pages 241–246, South Lake Tahoe, California, 1999. Sandia National Labs.

[P12] P. Fleischmann, W. Pyka, and S. Selberherr. Mesh Generation for Application in Technology CAD. *IEICE Trans.Electron.*, E82-C(6):937–947, 1999.

[P11] W. Pyka, P. Fleischmann, B. Haindl, and S. Selberherr. Three-Dimensional Simulation of HPCVD – Linking Continuum Transport and Reaction Kinetics with Topography Simulation. *IEEE Trans.Computer-Aided Design*, 1999. accepted for publication.

[P10] P. Fleischmann, B. Haindl, R. Kosik, and S. Selberherr. Investigation of a Mesh Criterion for Three-Dimensional Finite Element Diffusion Simulation. In *International Conference on Simulation of Semiconductor Processes and Devices*, pages 71–74, Kyoto, Japan, 1999. Business Center for Academic Societies Japan.

[P9] W. Pyka, P. Fleischmann, B. Haindl, and S. Selberherr. Linking Three-Dimensional Topography Simulation with High Pressure CVD Reaction Kinetics. In *International Conference on Simulation of Semiconductor Processes and Devices*, pages 199–202, Kyoto, Japan, 1999. Business Center for Academic Societies Japan.

[P8] B. Haindl, R. Kosik, P. Fleischmann, and S. Selberherr. Comparison of Finite Element and Finite Box Discretization for Three-Dimensional Diffusion Modeling Using AMIGOS. In *International Conference on Simulation of Semiconductor Processes and Devices*, pages 131–134, Kyoto, Japan, 1999. Business Center for Academic Societies Japan.

[P7] P. Fleischmann and S. Selberherr. Three-Dimensional Delaunay Triangulation Constrained to Boundaries by a Modified Advancing Front Algorithm. *IEEE Trans.Computer-Aided Design.* submitted December 1998.

[P6] P. Fleischmann, E. Leitner, and S. Selberherr. Optimized Geometry Preprocessing for Three-Dimensional Semiconductor Process Simulation. In *IASTED Int. Conf. on Applied Modelling and Simulation*, pages 317–321, Honolulu, Hawaii, USA, August 1998.

[P5] P. Fleischmann and S. Selberherr. Three-Dimensional Delaunay Mesh Generation Using a Modified Advancing Front Approach. In *6th International Meshing Roundtable*, pages 267–276, Park City, Utah, 1997. Sandia National Labs.

[P4] P. Fleischmann and S. Selberherr. Fully Unstructured Delaunay Mesh Generation Using a Modified Advancing Front Approach for Applications in Technology CAD. *IEEE J.Technology Computer Aided Design*, August 1997. `http://www.ieee.org/journal/tcad/accepted/fleischmann-aug97/`.

[P3] P. Fleischmann, R. Sabelka, A. Stach, R. Strasser, and S. Selberherr. Grid Generation for Three Dimensional Process and Device Simulation. In *International Conference on Simulation of Semiconductor Processes and Devices*, pages 161–166, Tokyo, Japan, 1996. Business Center for Academic Societies Japan.

[P2] P. Fleischmann and S. Selberherr. A New Approach to Fully Unstructured Three-Dimensional Delaunay Mesh Generation with Improved Element Quality. In *International Conference on Simulation of Semiconductor Processes and Devices*, pages 129–130, Tokyo, Japan, 1996. Business Center for Academic Societies Japan.

[P1] E. Leitner, W. Bohmayr, P. Fleischmann, E. Strasser, and S. Selberherr. 3D TCAD at TU Vienna. In J. Lorenz, editor, *3-Dimensional Process Simulation*, pages 136–161, Wien, 1995. Springer.

Curriculum Vitae

PETER A. FLEISCHMANN

GOLDSCHLAGSTRASSE 78/22
A-1150 VIENNA, AUSTRIA/EUROPE
TEL: +43-1-9855344, +43-664-3262390, FAX: +43-1-8765207
fleischmann@iue.tuwien.ac.at

• December 1994 to date	Ph.D. studies under the supervision of Prof. Siegfried Selberherr at the Institute for Microelectronics, Technical University Vienna, Austria.
• December 1997	Internship at NEC Corp., Silicon Systems Research Labs., Sagamihara - Kanagawa, Japan. Setup of a parallel simulation environment including porting of a TCAD framework to NEC proprietary workstations (EWS-UX).
• July 1992 - October 1994	M.S. studies with specialization in Communications and Radio-Frequency Engineering at the Technical University Vienna, Austria. Diplom-Ingenieur in Electrical Engineering conferred on October 14, 1994 with honors. (Thesis deals with Voronoi diagrams)
• September 1993	Work for the ECANSE project, a C++ based graphical user environment for computer-aided neural software engineering, Program and System development - PSE, Siemens AG, Austria.
• October 1987 - June 1992	B.S. in Electrical Engineering, Technical University Vienna, Austria (with honors).
• July 1988	Installation and maintenance of electrical power supply systems, Elsont CO.KG., Vienna, Austria
• 1979 - 1987	High school/college: Bundesrealgymnasium XIII, Wenzgasse - 1130 Vienna, scientific branch. Matura June 1987 (A-Levels, with honors)
• 1975 - 1979	Elementary/primary school: Volksschule Am Platz - 1130 Vienna
• June 1, 1969	born to Austrian parents in Kabul, Afghanistan