Freie Universität Berlin
Fachbereich für Mathematik und Informatik
Institut für Mathematik

Diplomarbeit

Generation of constrained high-quality multi-material tetrahedral meshes

Max Kahnt

Matrikel-Nr. 4105950

30. April 2012

Betreut durch

Prof. Dr. Konrad Polthier

Eidesstattliche Erklärung Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Arbeit wurde in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Berlin, den 30. April 2012

Contents

1 Motivation

1.1 Introduction

Real-world problems that can be described by partial differential equations (PDEs) are generally not solved analytically. Either they are too complex or no mathematical theory exists that explicitly constructs an appropriate solution for arbitrary PDE boundary conditions. The space for which the PDE is to be solved is called the *domain* as from now on. It typically is bounded and can be embedded in 3-dimensional Euclidean space. Finite element methods are a common tool to approximate the solution of such PDEs numerically.

The finite element method requires the discretisation of the space of interest into so-called finite elements in order to perform the necessary computations piecewise. The type of finite elements chosen depends on the application. Polyhedral cells allow to linearly approximate the boundary and offer a discretisation that is easy to handle. Along with decompositions into hexahedral elements, tetrahedral meshes form the majority of unstructured meshes (see Owen (1998)). Because any polyhedron can be subdivided into tetrahedra, such meshes are the most general. Although certain applications might favor other polyhedra or even mixed meshes, this study is focussing on tetrahedral meshes.

1.2 Challenges in tetrahedral mesh generation

When tetrahedral meshes are used to compute solutions for PDEs by finite element analysis, the realization of the mesh plays an important role in the performance of the finite element computation. Ideally the mesh would be directly optimized w.r.t. the respective problem and the computational outcome. Such general optimization strategies are hard to obtain because the parameter space is huge. Additionally the actual problem might not be known precisely by the time the mesh is generated. Instead properties of the finite element methods are transferred to simpler measures on the meshes, e.g. count, size and shape of the elements. From these quantities, the accuracy of the preceding finite element computation can be estimated and its error can be bounded. Unfortunately it is not always possible to produce meshes that are optimal w.r.t. all of these quantities because they are usually not mutually independent and contradict one another. Hence the challenge lies in trading off as little as possible.

A mesh should approximate the boundary of the domain well, e.g. the Hausdorff distance from domain boundary to the mesh boundary should be kept low. One way to achieve this is to use small elements. Note that boundaries do not always have to be exterior boundaries. For domains exhibiting several distinct regions, the interior boundaries have to be approximated as well.
Using smaller elements necessarily increases their count for a fixed domain. Because computations are performed element-wise in the finite element method, its complexity is usually highly related to the number of elements of the mesh.
Further the shape of the tetrahedra plays a crucial role in the outcome of finite element computations. A tetrahedron exhibits several dihedral angles, which is the angle between two of its triangular faces, and they are good indicator of the element's quality in terms of shape.

In summary, the mesh generation method does not necessarily have to know about the underlying problem. The demands of the discretisation can be formulated in sizing and shape criteria. A tetrahedral mesh generator should aim at creating a mesh that satisfies these needs while producing as few elements as possible.

This thesis presents a method to generate tetrahedral meshes with provable quality properties. It avoids generating unnecessary elements while providing a good geometric approximation due to a criteria-driven refinement scheme.

1.3 Structure of the thesis

We begin with an examination of existing approaches in tetrahedral grid generation. Special attention is given to their applicability and performance in specific multi-material settings. Recently, a method has been proposed that simplifies straightforward treatment of such setups but does not perform optimal w.r.t. the number of elements inserted to achieve high geometric accuracy. An investigation of the mechanisms used will lead to an approach to extend the method.

In chapter 3 our algorithmic extension is introduced conceptually. A proof that emphasises the validity of the approach is given. The strategy is gradually refined and extended to finally yield a coherent upgrading of the original method. The guaranteed properties of the extension are discussed as well as its restrictions.

In chapter 4 the algorithmic development framework is presented and the actual implementation is described. A current research problem in orthopedic surgery is addressed.

Finally the proposed approach is summarised. It is investigated to what extent it solves the original problem. A comparison with other existing approaches and implementations shows the relative advantages and disadvantages and highlights further topics of research.

2 Related work

This chapter explains how to evaluate the quality of tetrahedral meshes and introduces current methods to generate them. Particular attention is paid to the practical usability, which emphasises the advantages of a recent approach. An introduction to the concepts used in this approach leads to a detailed review and investigation of the method.

2.1 Tetrahedral mesh generation

Good linear finite element. Shewchuk (2002) extensively discusses how to evaluate the fitness of a tetrahedral mesh for finite element computations. A brief summary is given:
Creating smaller tetrahedra allows to decrease the interpolation error. Element size can be adjusted locally where the function changes quickly over the domain and in the vicinity of important geometric features. But generating a lot of unnecessarily small elements slows down the PDE approximation process and should be avoided. Finally, large dihedral angles compromise the interpolation error of the gradient function. This effect is only local. As Shewchuk (2002) shows, the minimal dihedral angle in the overall mesh highly relates to the condition of the finite element's stiffness matrix. A low minimal dihedral angle causes a high condition number which again slows down the convergence speed of the finite element computations significantly or even makes the results unreliable. In other words, the higher the minimal dihedral angle, the lower the discretisation error. By this result, shape has a global influence on the computational outcome.

A major difficulty for current mes generation methods results from the choice of representation of the input, i.e. the domain to be discretised. Its description might be complex or even inconsistent. For instance in case of multi-material domains, the regions are often defined separately, casually as different data types. Hence, below we will investigate tetrahedral mesh generation methods w.r.t. their demands on the input representation, especially for multi-material settings. The precomputations necessary in such setups will also be taken into account. To evaluate the resulting meshes, the quality properties of interest are the geometric approximation accuracy, the overall element count, and the guaranteed size and shape of the tetrahedra.

Explicit boundary representation. Many methods require the input domain to be given as a single consistent boundary representation. We begin with an investigation of the effort to derive such a representation in cases where it is not available directly, e.g. applications that entail investigating the interaction of objects that are not combined originally. Objects that may change their relative position significantly imply such a setting. Creating a consistent boundary representation of the fusion requires the explicit evaluation of the configuration of the separate domains. Difficulties arise where objects overlap and a decision has to be made in order to assign the overlap region to one domain or the other (or none). These configurations can be encoded in Boolean operations preserving the material assignments.

To perform Boolean operations on a set of boundaries, their intersections have to be computed. Further their intrinsic connectivity has to be adjusted eventually and the parts to discard have to be identified. While the boundary is maintained very accurately, this approach itself is subject to robustness issues for degenerate cases.
Performing Boolean operations on voxel data is easy and robust, because the Boolean prescriptions can simply be evaluated cell-wise. Then a surface has to be derived, e.g. using Hege et al. (1997). A conversion into voxel space might cause an essential loss of information. Curved 1-dimensional features cannot be represented smoothly with this approach.
Of course Boolean operations can also be performed with other geometric models, e.g. other B-reps and CSG. Still, deriving a single consistent surface mesh requires converting the separate domains to suitable representations and another conversion of the Boolean result.

Different approaches exists to generate tetrahedral meshes and a common classification outlines two important approaches: Advancing front methods and Delaunay refinement strategies, differing from each other in the way the mesh vertices are chosen. Variational schemes usually assume the presence of an initial mesh that is subject to optimization (see Alliez et al. (2005)). Such approaches are not considered as original mesh generators for that reason. Despite the classification of the methods, hybrid methods exits of course, where the advantages of both approaches (or those not mentioned here) are attempted to be exploited.

Advancing front methods. The advancing front technique starts with a given surface mesh. The surface is promoted into the interior, generating layers of tetrahedra. The result conforms to the boundary. This can be advantageous or disadvantageous, depending on whether the boundary triangulation needs to be preserved exactly for some reason or has to be derived for mesh generation specifically. Advancing front methods have difficulties in successfully closing up their fronts with provably quality for arbitrary surfaces. A basic method is described by Löhner and Parikh (1988), advanced choice of locations to insert points employ elaborate strategies, e.g. Radovitzky and Ortiz (2000), Yang et al. (2005).
Schöberl (1997) present an advancing front approach that derives its initial front (the boundary) from a given CSG representation. Intersection tests are performed recursively in order to determine points at possible singularities and assure a good approximation of the bounding surface.

Delaunay refinement strategies. Based on the mathematical concept of a Delaunay triangulation, these methods aim at inserting vertices at the right locations in order to refine an initial coarse mesh such that it meets the constraints on the elements' shape and size. Based on a two-dimensional method by Chew (1989) many strategies have been developed and properties of the Delaunay triangulation have been exploited to obtain provable mesh quality (Ruppert (1995)). The approach has been adapted to three-dimensional mesh generation.
Shewchuk (2002) denotes that the radius-edge ratio is the measure naturally improved with Ruppert-like Delaunay refinement strategies. Refining tetrahedra with small radius-edge ratio removes most types of tetrahedra that bear small angles, but those that are called *slivers*. Several approaches aim at removing them from meshes where the radius-edge ratio has been optimized beforehand (see Edelsbrunner et al. (2000), Cheng et al. (2000)).

Because most of the following methods require them, we introduce two types of boundary representations: A *piecewise linear complex* (PLC), which was introduced by Miller et al. (1996), is a general boundary description consisting of vertices, together with a collection of segments and facets. A *piecwise smooth complex* (PSC) allows the surface patches to be smooth, as well as the curves where surface patches are stiched together.

Si and Gärtner (2005) has its focus on generating tetrahedra for a given PLC with the least number of points added. The concept of a *constrained Delaunay triangulation* is used to determine the location where to insert points in order to recover the boundary of the PLC accurately. An implementation is publicly available in the realisation in Tetgen (Si (2006)). Quality issues are not dealt with in the original algorithm, but an extension based on Shewchuk (1998) is implemented that guarantees termination if no angle less than 90°is present in the input PLC. Si (2006) claims that their algorithm terminates if no input angle less than 60°and a radius-edge ratio below 2 can be guaranteed.

Cheng et al. (2004) address the issue of handling small input angles. Again the input is a PLC. Their approach terminates with arbitrary angles and they show that resulting tetrahedra not respecting the radius-edge length criterion are caused by (and lie close to) small angles not allowing for an improvement with their method.

Rineau and Yvinec (2008) assume a more general input, namely a PSC. Their algorithm suffers from a severe angle restriction though, presuming that no point exhibits smooth patches meeting at an angle less than $\frac{\pi}{2}$.
Cheng et al. (2010) propose a method to mesh PSCs that has no constraints on the input angles. While they focus on the theoretical guarantees and put implementational aspects aside, Dey and Levine (2009) trades off theoretical guarantees for implementable quality. Both approaches require the explicit representation of the PSC.

Most of the above approaches, i.e. advancing front and Delaunay methods, can deal with multi-material setups (although they are rarely mentioned in the articles cited). The methods that leave the boundaries unchanged allow for a material-wise application. The other methods may change the boundary but do not restrict the boundaries to separate interior from exterior. Hence internal boundaries can also be treated and an assignment of materials according to the tetrahedra barycenters can be accomplished on all resulting meshes.

An abstract domain: The oracle. While the above methods rely on a certain explicit input representation, Oudot et al. (2005) propose a Delaunay refinement algorithm, that handles the input more abstractly through a so-called *oracle*. They require the oracle to reliably respond to only two types of simple queries:

1. for a point p determine the material it belongs to and
2. for a line segment e compute one intersection of e with the boundary, if any.

The demands in item 2 can be reduced to an inquiry of the existence of an intersection, because the computation of the point can always be performed by segment subdivision and the use of item 1 up to arbitrary decimal precision.
An algorithm relying on this black-boxing idea, can access a consistent representation of the domain. The point queries of item 1 have to be handled such that they respect the Boolean configuration of the composed domain and can be evaluated separately. None of the problems arise that are involved in computing a single consistent boundary representation as long as the point queries can be handled robustly for each subdomain.

For setups that require the fusion of several different domains, the oracle approach is advantageous in comparison to those that need rely on a boundary representation. The oracle can well be understood as an inifitesimal version of a voxel data approach, but still differs in the fact that evaluation at a point is only triggered on concrete query. However, it also shares a disadvantage with the voxel data representation: The boundaries are not known explicitly. Point insertion is guided by the refinement rules only. Oudot et al. (2005) employ the second query type of the oracle (item 2) in order to approximate the boundary well. However, they exclude boundaries exhibiting non-smoothness from the algorithm and their proofs consequently. 1-dimensional features are unlikely to be accurately recovered in the resulting grid correspondingly, a desirable property still missing and motivating this thesis.

2.2 General definitions

This section introduces the mathematical concepts, the Delaunay mesh generation process is based on. As a starting point, a specific structure to separate decompositions of space with nice intersection properties from the arbitrary ones will be introduced. Further there is a natural way of assigning regions to a set of points in space, which yields an appropriate decomposition of space and, if the points are chosen appropriately, of the domain of interest. By the concept of duality, an associated decomposition is derived which can be proven to yield a tetrahedral mesh for negligible assumptions. The statements in this chapter are given without proofs.

2.2.1 Decomposition of space

The geometric structures that will be observed in this chapter can be generalized with a simple mathematical concept. We stick to the notion of a *complex*, i.e. a system of hierarchically ordered elements with certain intersection properties, decomposing a topological space. The hierarchy is understood w.r.t. the elements dimension. As we will deal with triangulations in $\mathbb{R}^3$, it suffices in this context to introduce the complexes mostly according to Edelsbrunner and Harer (2010).

Definition 2.2.1.1 (Abstract simplicial complex)**.** *An* abstract simplicial complex *is a finite collection of sets A such that $\alpha \in A$, $\beta \subset \alpha$ implies $\beta \in A$ and the dimension of the complex is the maximum dimension of any of its simplices.*

Remark 2.2.1.1. The sets in A are its *simplices*, the *dimension* of a simplex is $\dim \alpha = \operatorname{card} \alpha - 1$.

Theorem 2.2.1.1 (Geometric realization of an abstract simplicial complex)**.** *Every abstract simplicial complex of dimension d has a geometric realization in $\mathbb{R}^{2d+1}$.*

Remark 2.2.1.2. As our final meshes will consist of simplices of dimension 3, we can assure there is a geometric realization in $\mathbb{R}^7$. This is not very helpful w.r.t. the discretisation of a $3D$-object. Fortunately we don't have to go this way round (define the complex first, then embed it into Euclidean space) but derive a 3-complex embedded in $\mathbb{R}^3$ by construction.

Definition 2.2.1.2 (Simplex)**.** *A k-*simplex *is the convex hull of $k+1$ affinely independent points in $\mathbb{R}^d$. Its* dimension *is k. A* face *of a simplex σ is a subset τ and we write $\tau \leq \sigma$. We call τ a* proper *subface of σ if $\tau \subsetneq \sigma$.*

We use special names for the first few dimensions, *vertex* for 0-simplex, *edge* for 1-simplex, *triangle* for 2-simplex, and *tetrahedron* for 3-simplex.

Definition 2.2.1.3 (Simplicial complex)**.** *A* simplicial complex *is a finite collection of simplices K such that $\sigma \in K$ and $\tau \leq \sigma$ implies $\tau \in K$, and $\sigma, \sigma_0 \in K$ implies $\sigma \cap \sigma_0$ is either empty or a face of both.*

In order to discretise the domain using tetrahedra, we aim at finding or generating a simplicial complex. However, there is a generalization of these complexes that allows for a little more generic decomposition and which we will start off from.

Definition 2.2.1.4 (Polytope)**.** *A* polytope *is an intersection of a finite number of half-spaces. The dimension of the polytope is the smallest dimension of a proper affine subspace one can restrict the original space to without affecting the polytope.*

Remark 2.2.1.3. There might exist infinitely many such descriptions. However the minimal set of half-spaces is unique.
We are restricting to convex polytopes. Up to dimension three, we call a bounded polytope *vertex*, *edge*, *polygon*, or *polyhedron* respectively. A face of a polytope is an intersection of the polytope with a halfspace such that none of the interior points of the polytope lie on the boundary of the halfspace (actually it suffices to look at the intersections with hyperplanes to obtain the proper faces).

Definition 2.2.1.5 (Polytopal complex)**.** *A* polytopal *complex K is a set of polytopes such that every face of a polytope in K is also in K and the intersection of any two polytopes in K is a face of both. The polytopes in K are called its cells.*

Evidently simplicial complexes are polytopal complexes, and apparently, the converse is not true for all polytopal complexes.

2.2.2 Voronoi diagrams

Generating complexes is quite costly if all cells have to be constructed manually, especially because one has to take care of not contradicting its properties. But there is a useful concept in computational geometry that allows to access the advantages of complexes without generating the cells explicitly. Instead we start with a simple set of points (the vertices) in some metric space (V, d) and have a look at the Voronoi decomposition.

Definition 2.2.2.1 (Voronoi region)**.** *Let P be a finite set of points in (V, d). For $p \in P$, we define the* Voronoi region *of p as*

$$\mathrm{V}(p) := \{x \mid \forall q \in P : d(x, p) \leq d(x, q)\}$$

Remark 2.2.2.1 (Voronoi faces). For $Q \subset P$ any non-empty

$$\mathrm{V}(Q) := \bigcap_{q \in Q} \mathrm{V}(q)$$

is called a face and for two faces $\alpha = \mathrm{V}(Q_1), \beta = \mathrm{V}(Q_2)$

$$\alpha \subset \beta \Longleftrightarrow Q_1 \supset Q_2.$$

Definition 2.2.2.2 (Voronoi diagram)**.** *The Voronoi diagram is defined as*

$$\mathrm{Vor}\,(P) := \bigcup_{Q \subset P} \mathrm{V}(Q)$$

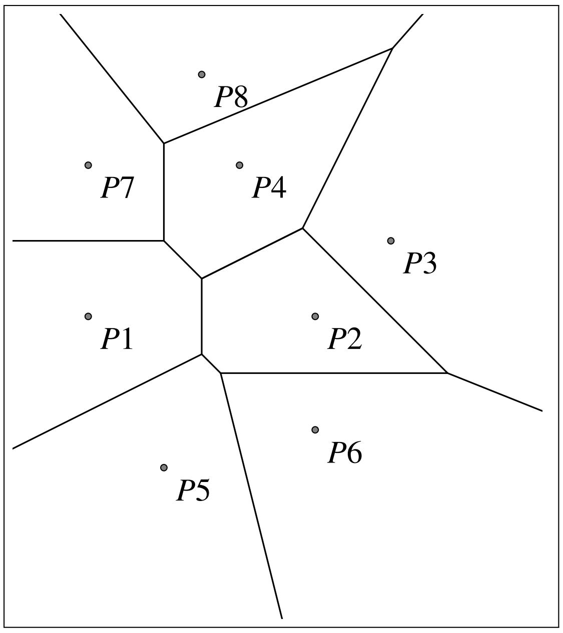


Figure 2.1: Voronoi diagram in the plane. Voronoi edges and Voronoi facets (and Voronoi cells in 3-dimensional space) might be bounded or unbounded. All are the intersection of finite half spaces. The cell a point P_i is contained in is the set of points in space closest to it.

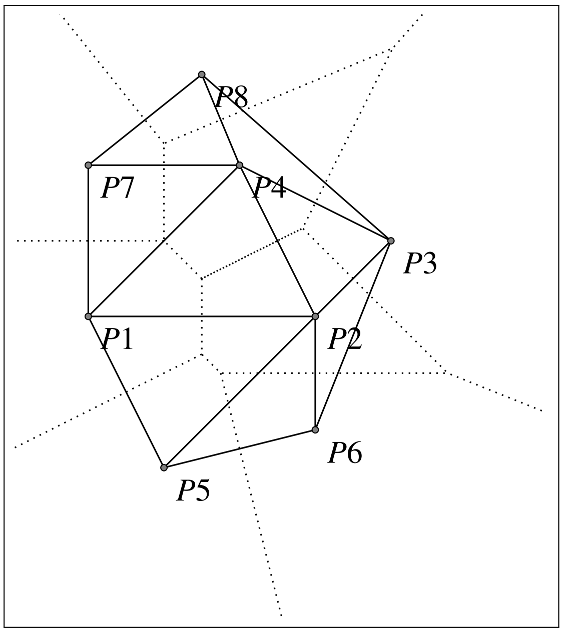


Figure 2.2: The dual Delaunay triangulation. Triangulates the convex hull of the point set. In 2-dimensional space each Voronoi facet has a dual Delaunay edge. The do not necessarily intersect. The same holds in higher dimensions.

Lemma 2.2.2.1 (Face convexity in $\mathbb{R}^d$)**.** *All faces of* Vor (P) *are convex.*

Lemma 2.2.2.2 (Voronoi diagram is polytopal complex)**.**

Definition 2.2.2.3 (General position)**.** *Let P be a finite point set in $\mathbb{R}^d$. We say P is in* general position*, iff no $d+2$ points lie on a common $(d-1)$-sphere.*

Note: We assume P is in $\mathbb{R}^3$ and in general position as from now.

Remark 2.2.2.2. Let $f \in \operatorname{Vor}(P)$ be an i-dimensional face of the Voronoi complex of P. Then for f holds:

$$\exists! Q \subset P, |Q| = i : \bigcap_{p \in Q} \mathrm{V}(p) = \mathrm{V}(Q) = f$$

For $i = 1, \dots, 4$ we call f resp. a *Voronoi cell*, *Voronoi facet*, *Voronoi edge* and *Voronoi vertex*. $i \geq 5$ is disqualified by the imposed general position condition on the vertices of P.

A face $f \in \operatorname{Vor}(P)$ might be *bounded* or *unbounded* in case it is a Voronoi cell, facet or edge; see fig. 2.1.

Voronoi diagrams are partitions of the space that are used in many different kinds of application. However we are mainly interested in its geometric dual.

2.2.3 Delaunay triangulations

Definition 2.2.3.1 (Delaunay complex)**.** *Let P as before. The vertices of* Del(P) *are precisely the points of P and $\{p_i\}_{1\leq i\leq 4}$ points are connected by a hyperedge iff there is a sphere centered at point x which passes through the points $p_1, \dots, p_i$ and does not contain any point $p \in P \setminus \{p_i\}_{1\leq i\leq 4}$.*

Lemma 2.2.3.1. *The Delaunay complex of P is a simplicial complex.*

Remark 2.2.3.1 (Delaunay complex is **Delaunay triangulation**). In $\mathbb{R}^3$ the Delaunay faces are called *vertices*, *edges*, *triangles* and *tetrahedra* respectively. Due to the general position of the points those can be guaranteed to be geometrically non-degenerate:

- there can be no 4-hyperedge with all 4 vertices on the boundary of a circle without simultaneously having a sphere passing through these points and any other point of the point set (assumed $|P| > 4$ and not all points are contained in some affine 2-dimensional subspace)
- neither can there be a 3-hyperedge with all 3 vertices on a straight line.

Vor (P) and Del(P) are unique by definition. For P in general position Del(P) is a triangulation of P as described above, also being unique and by that means from now on also called the Delaunay triangulation.

Lemma 2.2.3.2. *The* Delaunay complex Del(P) *is the dual of the corresponding Voronoi complex* Vor (P) *and vice versa.*

Hence, for points in general position there is a one-to-one-relation of elements in the Delaunay complex and those in the Voronoi complex (see fig. 2.2). By this means we introduce another convention. Delaunay elements are simply identified by the points of P they are incident to, i.e. pqr is a Delaunay triangle. Correspondingly Voronoi elements are identified by their dual, i.e. V_{pqr} is the Voronoi edge that is dual to the Delaunay triangle pqr.

Definition 2.2.3.2 (Restricted Delaunay triangulation, adapted but according to Dey (2007), Definition 1.8)**.** *Let P be a set of points and $S \subset \mathbb{R}^3$. The* restricted Delaunay triangulation *of P with respect to S is*

$$\mathrm{Del}_{|S}(P) = \bigcup_{Q \subset P} \mathrm{V}(Q) \cap S \neq \emptyset$$

Remark 2.2.3.2. We slightly depart from this general definition as proposed in Boissonnat and Oudot (2005). Let ∂O be a surface. Then

$$\mathrm{Del}_{|\partial O}(P) = \{p_1 p_2 p_3 \in \mathrm{Del}(P) \mid V_{p_1 p_2 p_3} \cap S \neq \emptyset\}$$

contains only those triangles whose dual intersects the surface and their subfaces. We neglect other lower-dimensional faces and assume that no higher-dimensional element is in the restricted triangulation.
We proceed analogously for volumes $O \subset \mathbb{R}^3$. Then

$$\mathrm{Del}_{|O}(P) = \{p_1 p_2 p_3 p_4 \in \mathrm{Del}(P) \mid V_{p_1 p_2 p_3 p_4} \cap O \neq \emptyset\}$$

contains only those tetrahedra (and their subfaces) whose circumcenter is within O. The restricted Delaunay triangulations still are subcomplexes of the Delaunay triangulation.

Definition 2.2.3.3 (Conforming Delaunay triangulation)**.** *Let C be a simplicial complex.*

P is said to conform *to C, iff*

$$\forall Q \in C \, \exists R \subset \mathrm{Del}(P) : Q = \bigcup_{r \in R} r$$

in words that each simplex of C occurs (possibly subdivided) also in the Delaunay triangulation of P. $\mathrm{Del}(P)$ *is then said to be a* conforming Delaunay triangulation *of (or respecting) C.*

2.2.4 Delaunay triangulation properties

Empty-sphere criterion. By definition of the Delaunay triangulation, for each face f there is at least one sphere passing through the vertices of f not containing any other point of the point set. Furthermore the inverse also holds, i.e. if there is sphere passing through $p_1, \ldots, p_i$ not containing any other point of the point set, then there is a Delaunay face $p_1 \ldots p_i$. Such a ball will be called a *Delaunay ball* from now on. This property can be excessively exploited, e.g. when incrementally building a Delaunay triangulation by successively inserting points.

Definition 2.2.4.1 (Circumsphere, -center, -radius)**.** *For a point set Q a circumsphere S is a sphere passing through all points of Q. It might not be unique in case* $|Q| < 4$ *or not existent if* $|Q| > 4$ *in* $\mathbb{R}^3$*. The circumcenter is the center and the circumradius is the radius of such a sphere.*
For $|Q|| = 2$ *the smallest circumsphere is called the* diametral sphere *or* diametral ball.

Encroached tetrahedra. As the Delaunay ball of a tetrahedron is unique (it is exactly the circumsphere), we can directly state whether the insertion of an additional point $q \notin P$ affects a tetrahedron of $\mathrm{Del}(P)$. A tetrahedron t in $\mathrm{Del}(P)$ with circumcenter c and circumradius r is said to be *encroached* by a point q iff $d(c, q) < r$. If t is encroached by q then $t \notin \mathrm{Del}(P \cup \{q\})$. With this notion the work pioneered by Bowyer (1981) and Watson (1981) can be used to
Incrementally build the Delaunay triangulation of a finite point set $P, |P| = n$. Let $P_i = \{p_1, \ldots, p_i\}$. Start with P_4. Assuming these points are not all contained in a hyperplane, $\mathrm{Del}(P_4)$ contains the corresponding tetrahedron and all its faces. For algorithmic reasons one also inserts *virtual, infinite* Delaunay elements, i.e. a point at infinity and edges, triangles and tetrahedra connecting the faces forming the respective convex hull

of P to this vertex. Now for a new point p_5 we identify all encroached tetrahedra, remove them from the triangulation, leaving a star-shaped cavity at p_5. It can be shown that the new faces to be formed can be created by simply connecting all cavity-enclosing facets to p_5, yielding $\mathrm{Del}(P_5)$. Repeat analogously with the other points up to p_n and obviously $\mathrm{Del}(P_n) = \mathrm{Del}(P)$.

The existence and type of cavity also shows that such point insertions are local only, making it a valuable tool for our mesh generation algorithm later on.

Encroached triangle. A triangle $t = nop$ may have several Delaunay balls. Assume each triangle has an associated Delaunay ball $B(c, r)$. A triangle is said to be encroached by $q \notin P$ iff $q \in B(c, r)$. It is not true that an encroached triangle will not occur in the extended point set $P \cup \{q\}$, but iff both tetrahedra t_1, t_2 incident to t are encroached by q.

2.3 Oracle-based volume meshing

This section will introduce further definitions that are relevant in the understanding of the work of Oudot et al. (2005). The generic design of similar algorithms is explained as well as the realization of the authors. Oudot et al. (2005) formulate their strategy as an abstract set of rules. Based on these, the validity of the approach is proved and some guarantees on the output are provided. Finally, the restrictions of the algorithm are discussed to emphasise the subsequent efforts.

2.3.1 Domain description and approximation

Definition 2.3.1.1 (Labeling function, material)**.** *Let $z : \mathbb{R}^3 \to \{0, \ldots, n\}, n \in \mathbb{N}$. Then we define*

$$O_i := \overline{\{x \in \mathbb{R}^3 \mid z(x) = i\}}$$

If O_i bounded for all $i > 0$ then we call z a labeling function*, O_i the $i-th$ material and ∂O_i its boundary.*

Remark 2.3.1.1. The above definition resembles the use of a *labeling function* in Pons et al. (2007). When the term ∂O occurs in the remainder of this thesis, then we are either in the context of having a labeling function defining only a single material or $\partial O = \bigcup_{i>0} \partial O_i$.

Definition 2.3.1.2 (Medial axis)**.** *The* medial axis *of ∂O, denoted by M, is the topological closure of the set of points of $\mathbb{R}^3$ that have more than one nearest neighbour in ∂O.*

Definition 2.3.1.3 (Distance to the medial axis)**.** *For a point $x \in \mathbb{R}^3$, we call* distance to the medial axis $d_M(x)$ *at x the Euclidean distance from x to the medial axis of ∂O.*

Remark 2.3.1.2. What we call *distance to the medial axis* is sometimes also called the *local feature size*. We avoid this notion whatsoever, because in Boissonnat and Oudot (2005) it is used in a slightly different meaning.

Definition 2.3.1.4 (Reach)**.** *We define the reach of ∂O the minimal distance to the medial axis, i.e.*

$$reach(\partial O) = \inf\{d_M(x) \mid x \in \partial O\}$$

Definition 2.3.1.5 (ε-sample)**.** *A finite point set P is an* ε-sample *of ∂O if $\forall x \in \partial O$: $P \cap B(x, \varepsilon d_M(x)) \neq \emptyset$.*

Definition 2.3.1.6 (Loose ε-sample)**.** *A finite point set P is a* loose ε-sample *of ∂O if the two following conditions are verified:*

1. $\forall x \in \partial O \cap \mathrm{Vor}_E(P) : P \cap B(x, \varepsilon d_M(x)) \neq \emptyset$
2. $\mathrm{Del}_{|\partial O}(P)$ *has vertices on all the connected components of ∂O*

where $\mathrm{Vor}_E(P)$ denotes the set of all edges of $\mathrm{Vor}(P)$.

Definition 2.3.1.7 (Sparse sample)**.** *Given a positive constant κ and a positive 1-Lipschitz function ϕ, a point sample P of ∂O is said to be (κ, ϕ)-sparse if $\forall x \in P : d(x, P \setminus \{x\}) \geq \kappa\,\phi$.*

Remark 2.3.1.3. An ε-sample is said to be κ-sparse if $\forall x \in P : d(x, P \setminus \{x\}) \geq \kappa\,\varepsilon\, d_M(x)$.

2.3.2 Generic Delaunay refinement scheme

Domain. Let

$$z : \mathbb{R}^3 \to \{0, \ldots, n\}, n \in \mathbb{N}$$

be a labeling function where $z^{-1}(0)$ is the exterior and $z^{-1}(i)$ represents a distinct material for each $i > 0$. z describes the multi-material domain that is to be discretised.

Quality criteria. Let further

$$a_1, \dots, a_k : \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}, \quad b_1, \dots, b_l : \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$$

be size and shape criteria for triangles (a_j) resp. tetrahedra (b_j).
For logical reasons the shape rating should generally be invariant under similarity transformations whilst the sizing certainly depends on the scale but should still be invariant to rigid transformations. Please note that nevertheless one might want to apply different criteria depending on the location in space.

Initial point set. Let $P_0 \subset \partial O$ be a set of points, such that the restricted Delaunay triangulation has vertices on all connected components of ∂O.

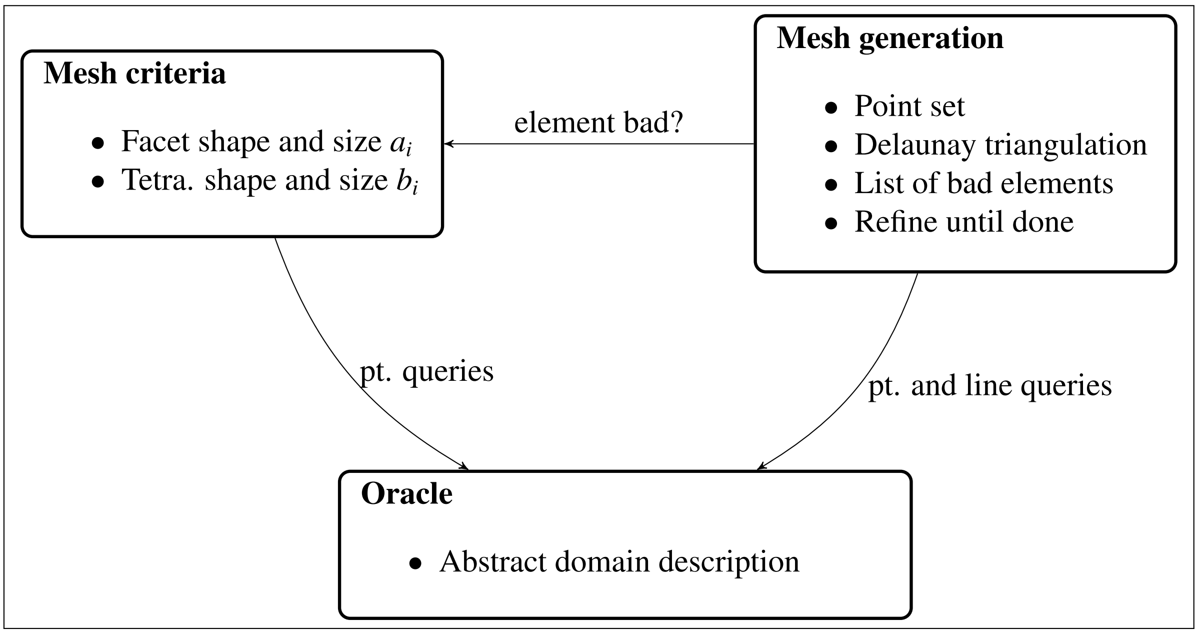


Figure 2.3: Interaction of mesh generation method, quality criteria and the abstract domain representation. Point queries are used to obtain the corresponding material. Line queries are used to determine the location of points to insert in order to approximate the boundary fo the domain.

Algorithm Starting off with an initial point set, the key idea is to adjust its Delaunay triangulation to fit the user-given set of criteria by successive insertion of points. The criteria and the strategy how to achieve their compliance is put into a set of rules operating on elements of the Delaunay triangulation or restricted Delaunay subsets.

Rineau and Yvinec (2007) propose a software design to implement such strategies and claim that their approach fits all known Delaunay refinement algorithms. The whole meshing procedure is tied up into so-called meshing levels, each dealing with the shape and quality criteria of Delaunay cells of a single dimension only. It is assured that point insertions triggered by higher dimensional cells do not contradict existing lower dimensional Delaunay elements by the notion of *encroachment*.

We present a Generic Delaunay refinement algorithm (p. 19). A cell (triangle t or tetrahedron u) of the Delaunay triangulation is called *bad*, if $a_j(t) > 0$ resp. $b_j(u) > 0$ for some j. Please note that not all computations have to be done repeatedly from scratch, e.g. the changes of $\mathrm{Del}(P)$ can be computed incrementally and also lists of *bad* triangles and tetrahedra which get updated progressively allow to reduce retrieval computations.

Generic Delaunay refinement algorithm

```
1. Initialize $P = P_0$.
2. Compute $\mathrm{Del}(P)$ and if necessary restricted versions.
3. If there is a bad surface triangle $t$:
    a) Choose $p$, e.g. $p \in \{V_t \cap \mathrm{Del}(P)_{|\partial O}\}$
    b) Update $P = P \cup \{p\}$
    c) Return to item 2
4. If there is bad tetrahedron $u$:
    a) Let $q$ be the circumcenter of $u$.
    b) If $q$ encroaches on some surface triangle $t$ goto item 3a
    c) Update $P = P \cup \{q\}$
    d) Return to item 2
5. Done.
```

Result From the structure of the algorithm it is clear that it cares first for the satisfaction of the triangle criteria (a_j) followed by those for the tetrahedra (b_j). The algorithm terminates when all criteria are fulfilled, therefore the result can be guaranteed to satisfy all of them. Hence, the crucial point is to show the termination of the algorithm.

During the course of the algorithm that is to be presented, an iterative and greedy strategy leads to successive point insertions in P. After each insertion, the overall Delaunay triangulation $\mathrm{Del}(P)$ as well as its restricted associate $\mathrm{Del}_{|\partial O}(P)$ is updated accordingly. Elements of $\mathrm{Del}_{|\partial O}(P)$ are called *facets*. Each facet f has by definition at least one intersection of its (Voronoi) dual with ∂O and by property of this Voronoi element, all these intersection points are equally far from the vertices of f (the distance is different for distinct intersection points though). For each facet f an arbitrary intersection point is chosen that serves as the center of a so-called *surface Delaunay ball* passing through the vertices of f.

2.3.3 Oracle-based algorithm

Oudot et al. (2005) present a three-dimensional meshing algorithm that derives from the surface meshing method of Boissonnat and Oudot (2005) and extends the results therein to the volumetric mesh generation. Rule Set R (p. 20) is a complete description of the strategy as a set of rules. The rules are repeatedly considered according to their order until no more element coincides with a rule premise.

Rule Set R

R1 if a facet f of $D_{|O}(P)$ does not have its three vertices on ∂O, then refine f;
R2 if a facet f has a surface Delaunay ball $B(c, r)$ with ratio $\frac{r}{\sigma(c)} > \alpha$, then refine f;
R3 if a tetrahedron t of $D_{|\partial O}(P)$ has a has a circumradius r greater than $\sigma(c)$, where c is the circumcenter of t, or if t has a radius-edge ratio greater than B, then consider the circumcenter c of t:

R3.1 if c is not included in any surface Delaunay ball, then insert c in P;
R3.2 else, insert in P the center of one surface Delaunay ball containing c.

For a facet f let $B_f(r_f, c_f)$ be the associated surface Delaunay ball; for a tetrahedron t

let $B_t(u_t, c_t)$ be the circumsphere. The criteria implied by the rules are the following:

$$(a_1) \qquad a_1(f) = \sigma(c_f) - r_f$$

$$(a_2) \qquad a_2(f) = \begin{cases} 0 & \text{all vertices of } f \text{ are in } \partial O \\ 1 & \text{else} \end{cases}$$

$$(a_3) \qquad a_3(f) = \alpha\sigma(c_f) - r_f$$

$$(b_1) \qquad b_1(t) = \sigma(c_t) - r_t$$

$$(b_2) \qquad b_2(t) = B - r/e(t)$$

Actually a_1 is not present in the rule set of Oudot et al. (2005). It is mentioned here for completeness because the volume mesh generation process presumes the results of Boissonnat and Oudot (2005) who use a_1 only. Now it is claimed that the algorithm implicitly guarantees the same results as in Boissonnat and Oudot (2005) (for suitable input). Omitting a_1 and looking at what happens when a_2, a_3 are satisfied both for the first time in the meshing process, shows that this is actually true: a_2 should not have been violated at all up to this point because the initial point set as well as all additionally inserted points are chosen to be in ∂O; and if all points satisfy a_3 for $\alpha < 1$ then this obviously implies the satisfaction of a_1.

2.3.4 Output guarantees and termination

Relying on the fact that a_1 is implied by a_2, a_3, Oudot et al. (2005) claim that the algorithm initially constructs a point set P_i that is a $\frac{1}{3}$-sparse 0.09-sample of ∂O, or equivalently

$$\forall x \in \partial O : \qquad d(x, P_i) \leq 0.09 d_M(x)$$

$$\forall p \in P_i : \qquad d(p, P_i \setminus \{p\}) \geq 0.03 d_M(p)$$

Remark 2.3.4.1. We state that this is not entirely true although the implications remain valid, because a_1 has been replaced by a_2, a_3. There is no rule imposing that triangles t with higher value $a_3(t)$ are dealt with primarily. Such a rule would imply that indeed at some time when a_3 is triggered, the assumptions are valid, and it would be a natural choice in an implementation. But as this is not the case, there might be point insertions occurring in the algorithm of Oudot et al. (2005) due to a_3 that would not have been triggered in the algorithm of Boissonnat and Oudot (2005). Thus P_i can be only said to be κ-sparse 0.09-sample for some $\kappa \leq 1$. Still this is a constant and only this fact is required for the termination proof (we try to consider this observation in remark 3.1.2.1). Also in Boissonnat and Oudot (2005) the size of the facets is upper bounded by a user-given σ and most of the termination and approximation proofs assume $\sigma \approx \epsilon d_M$ for some ϵ. Now, choosing σ substantially smaller than these bounds will cause refinements that take the sample way below the defined sparseness. Again it still holds that during the course of the algorithm there will be some κ replacing the chosen $\frac{1}{3}$ in order to yield the desired bound.

The labeling function in this approach is binary, i.e. it only separates *exterior* from the *interior* that we want to generate a mesh for. The crucial theorem to approximate O appropriately relies on a sufficiently dense sampling P of its boundary ∂O which is supposed to be a 2-manifold that is (1) compact, (2) orientable, (3) and twice-differentiable.

Remark 2.3.4.2. Note that with our definition we are not dealing with an abstract 2-manifold any more. It suffices to have O bounded to assure compactness, because we are in $\mathbb{R}^3$ and closeness is imposed by definition of O. Orientability is also ensured. So the only relevant condition is the smoothness of the surface.

Remark 2.3.4.3. As stated in Oudot et al. (2005), ∂O being $C^{1,1}$ implies that the *reach* is positive, i.e. $\inf_{x \in \partial O} d_M(x) > 0$.

The choice of σ essentially determines the approximation quality of the result. This is formulated in the following statements.

Lemma 2.3.4.1 (Lemma 6.3 in Boissonnat and Oudot (2005)). *If $\sigma \leq \varepsilon d_M$, then, upon termination of the algorithm, P is a loose ε-sample of ∂O.*

Theorem 2.3.4.1 (Theorem 1 in Oudot et al. (2005)). *If P is a loose ε-sample of ∂O with $\varepsilon \leq 0.09$, then $D_{|\partial O}(P)$ is a closed 2-manifold ambient isotopic to ∂O at Hausdorff distance $O(\varepsilon^2)$ from ∂O and its normals approximate the normals of ∂O within an error of $O(\varepsilon)$. Moreover, ∂O is covered by the surface Delaunay balls of P and P is a $\varepsilon(1 + 8.5\varepsilon)$-sample of ∂O.*

At this point we have to make sure that $\sigma \leq \varepsilon d_M, \varepsilon \leq 0.09$ in order to access the results of the theorem, whenever a_1 is satisfied. As this criterion has been replaced with a_2, a_3, the theorem can be extended to

Theorem 2.3.4.2 (Theorem 2 in Oudot et al. (2005)). *$D_{|O}(P)$ is a 3-manifold ambient isotopic to O at Hausdorff distance $O(\varepsilon^2)$ from O where $\varepsilon = \min\left\{0.09, \sup_{x\in\partial O} \frac{\alpha\sigma(x)}{d_M(x)}\right\}$. Moreover, the surface Delaunay balls of P cover ∂O.*

Remark 2.3.4.4. In Oudot et al. (2005), the theorem involves the closure $\overline{O}$. Note that with our definition of O it is a premise that all O_i are closed.

As mentioned in the paper, the theorem's premises hold for P whenever rule R3 is triggered (i.e. no criterion a_j has been provoked), especially ∂O is covered by the surface Delaunay balls of P. This is essential in the proof of termination of the algorithm. However rules b_1, b_2 are not superfluous as we are interested in generating a *good* tetrahedral mesh. Proving termination of the algorithm guarantees fulfillment of the size and shape criteria for tetrahedra in whatever range B and α are allowed to be chosen.

In order to prove the **termination** of the algorithm, we need

Definition 2.3.4.1 (Insertion radius, Definition 6 from Oudot et al. (2005)). *Given a point p inserted in P by the algorithm, the* insertion radius *of p, or $r(p)$ for short, is the Euclidean distance from p to P right before its insertion (which is the length of the smallest Delaunay edge created when p is inserted). The insertion radius of a point p of the initial point set P_i is the Euclidean distance from p to $P_i \setminus \{p\}$.*

Defining for $x \in O$ the sizing function generalizations

$$\sigma_0(x) := \inf\{d(x, x') + d_M(x') | x' \in \partial O\}$$
$$\sigma'(x) := \min\{\alpha\sigma(x), 0.03\sigma_0(x)\}$$

σ_0 has been proven to be 1-Lipschitz and if σ is 1-Lipschitz, σ' is γ-Lipschitz, $\gamma = \max\{\alpha, 0.03\}$. This yields access to the main lemma needed for the termination proof.

Lemma 2.3.4.2 (Lemma 6 of Oudot et al. (2005)). *If $\alpha < \frac{1}{5}$ and $B \geq \frac{4}{1-5\gamma}$ then the following conditions are verified*

$$\text{(C1)} \qquad \forall p \in P: \qquad r(p) \geq \sigma'(p)$$
$$\text{(C2)} \qquad \forall p \in P \setminus P_{|\partial O}: \qquad \delta(p) \geq \frac{1}{1-\gamma}\sigma'(p)$$

where $\delta(p)$ is the Euclidean distance from p to ∂O.

Proving that the balls $(B(p))_{p\in P}$ with radius $\frac{1}{2(1+\gamma)}\sigma'(p)$ are pairwise disjoint (Lemma 7 in Oudot et al. (2005)) and that for any $p \in P$, $B(p) \cap O$ contains a ball of radius $\frac{1}{4(1+\gamma)}\sigma'(p)$ (Lemma 8 in Oudot et al. (2005)) prepares proving

Theorem 2.3.4.3 (Oudot et al. (2005))**.** *If $\alpha < \frac{1}{5}$ and $B \geq \frac{4}{1-5\gamma}$, then the output point sample P verifies*

$$|P| = O\left(\int_O \frac{dx}{\sigma_0^3(x)} + \int_O \frac{dx}{\sigma^3(x)}\right)$$

where σ_0 depends only on O (not on σ).

Arguing that σ_0, σ are both positive and continuous over O, which is compact, the bound is finite. Because the algorithm inserts one point in P per iteration (w.r.t. the rule set) and never removes points from P, the algorithm terminates (Corollary 2 in Oudot et al. (2005)).

2.3.5 Limitations of the original oracle-based approach

The quality of the resulting mesh of the method of Oudot et al. (2005) is not exposed explicitly. The shape criterion optimized in their approach is the radius-edge ratio, bounded by a user-given value B. This has only slight influence on the resulting minimal dihedral angle. Post-processing methods that assume Delaunay meshes with good radius-edge ratio aim at removing the remaining elements with low minimal dihedral angles, which are so-called *slivers*, by slightly perturbing vertices (see Edelsbrunner et al. (2000)) or modifying the metric used in the Voronoi diagram and the dual Delaunay triangulation (see Cheng et al. (2000)).

Settings involving multi-material domains put further demands on the mesh generation algorithm. There must be a strategy to assign materials to the tetrahedra. Pons et al. (2007) employ the oracle to determine restricted Delaunay triangulations $\mathrm{Del}_{|O_i}$ for each material separately. By the consistency of the oracle, this yields a consistent decomposition of the tetrahedra into the single material parts. It is assured that the boundaries conform with this decomposition, mainly because the lemmata of Oudot et al. (2005) can be applied for each material O_i separately.

Lemma 2.3.5.1 (Lemma 5 of Oudot et al. (2005))**.** *The surface Delaunay balls of P and those of $P_{|\partial O}$ are the same.*

When dealing with multi-material domains we will employ the material assignment proposed by Pons et al. (2007) but refer to the method by Oudot et al. (2005), because all theoretical results are retained from them.

Non-smooth boundaries. Not all multi-material domains can be handled as a result of the extension by Pons et al. (2007). Such domains induce boundaries that are non-manifold or disconnected or both. The case where ∂O is manifold but disconnected is handled perfectly by the algorithm as-is. But multi-material junctions rarely can be assumed to be smooth in non-manifold cases, i.e. there is a ∂O_i not smooth. Of course, this is a weakness of the approach by Oudot et al. (2005) already, because they exclude non-smooth boundaries.

In practice, the boundary is not explicitly known by the mesh generation algorithm, because it is hidden within the oracle. By that reason the approach can be applied to domains exhibiting non-smoothnesses, because for each finite sample point set of the boundary of the domain ∂O there is a domain O' with smooth boundary $\partial O'$ containing the point set as well. The termination and approximation proofs do not apply in that case because some properties of O have to be bounded a-priori, e.g. the reach. When applying the algorithm to such boundaries, the sizing field satisfying $\sigma \leq 0.09 d_M$ is replaced by one with positive lower bound, trading off the ϵ-sampling properties in the region of trimming. Without this deviation, it would not be able to handle any piecewise smooth input properly, especially surface triangulations enclosing the domain.
We call features of the domain non-smoothnesses due to multi-material junctions, sharp edges of domains or simply any arbitrary curve if it has some meaning for the domain. Albeit many elements might be introduced in the features vicinity by adjusting the sizing field, there is no guarantee that smooth feature curves will be adequately represented in the resulting mesh.

Apart from facet and tetrahedron sizing criteria that can be applied with different values per material, Pons et al. (2007) propose a sizing criterion that enables to tune the facet sizing adaptively during the mesh generation process. A facet distance bound g indicates a facet (Delaunay element pqr) to be too large if the distance $d(i_1, i_2) > g$ exceeds this bound where V_{pqr} the dual Delaunay segment (or ray) and i_1, i_2 its intersections with pqr and ∂O. This criterion is highly related to the curvature, i.e. it tends to enforce the generation of smaller elements in regions of high curvature of the boundary.

We propose an extension to the algorithm by Oudot et al. (2005) that aims at preserving an a-priori discretised representation of such features. Particularly we expect to provide a method that generates less elements in areas where the approximation of a feature is driven by a surface distance criterion. At the same time we expect providing higher geometric accuracy in this area due to the explicit representation of the feature.

3 Preserving segments in a Delaunay triangulation

We have seen that the oracle approach yields a very general concept to deal with arbitrary input domains. The Delaunay refinement strategy by Oudot et al. (2005) with the multi-label extension of Pons et al. (2007) allows to generate tetrahedra for multi-material domains. However, an accurate representation of multi-material junctions and sharp features is excluded from the theoretical foundation.
This chapter introduces an extension to the approach by Oudot et al. (2005) which aims to fill this gap. In section 3.1 the Rule Set R (p. 20) is extended such that it preserves given segments. Several assumptions are made on the segments' configuration, particularly that they shall not intersect. In the subsequent section these configurations are examined more closely and a preparatory algorithm is developed that guarantees to meet some of the requirements from section 3.1. Section 3.3 explains how to integrate the preparatory algorithm into the rule set of the algorithm. It is discussed to what extent the resulting method can handle the initial problem.

3.1 Preserving isolated segments

Let C be a simplicial complex. Then let E denote the set of its 1-dimensional simplices and P_E the set of proper subfaces of E, i.e. the endpoints of the segments. In accordance with the design of the mesh generation algorithm and the mathematical concept of a conforming Delaunay triangulation (definition 2.2.3.3), we add another criteria layer to the algorithm with highest priority. This layer will assure that the resulting triangulation conforms to the segments prescribed by E, which is why elements of E are called *constrained segments* as from now. The penalization of constrained segments not occuring in the Delaunay triangulation corresponds to the quality criteria for facets and tetrahedra (see fig. 3.1).

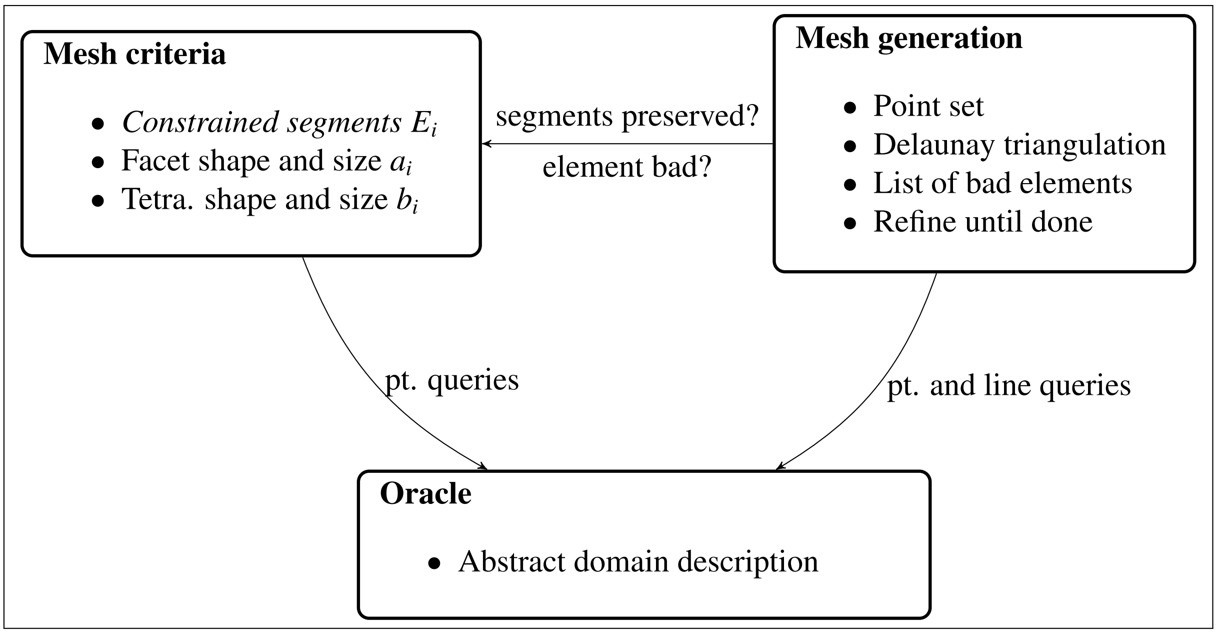


Figure 3.1: Our algorithm extension (compare fig. 2.3). We additionally employ the segments we want to preserve as a set of constrained segments. The mesh generation algorithm shall not insert points such that the preservation of the constrained segments is compromised.

For a segment to appear in a Delaunay triangulation, a necessary condition is that its endpoints are part of the point set. Hence, we need to add all points P_E to the initial point set P_0 of the algorithm. We will assume that all segments are present before starting the refinement algorithm. This and more assumptions will be further investigated and discussed in section 3.2 and section 3.3. During the course of the algorithm, a point insertion shall not extinct a constrained segment. Hence by the use of the notion of a point encroaching a segment, we assure that the relevant segments are maintained throughout the algorithm.

A segment e with endpoints a, b and midpoint m is said to be *encroached* by a point p if $d(p, m) < d(m, a)$, i.e. the point is in the open diameter ball of the segment.

Remark 3.1.0.1. Note that this encroachment definition differs from the former encroachment notions for tetrahedra and Delaunay facets, because it is triggered also on point insertions not necessarily extinguishing the segment from the Delaunay triangulation. The converse is true though: If there is no point within the diametral sphere, the corresponding segment is a simplex of the Delaunay triangulation.

3.1.1 Algorithm

The Algorithm preserving isolated segments (p. 29) departs slightly from the concept of the generic algorithm. So far the elements to be refined have been determined by evaluating the criteria functions a_j, b_j on Delaunay faces at the current state of P. We are not applying analogous functions for constrained segments. At this point, we consider a segment to be *bad* if it is supposed to be in the Delaunay triangulation of P, i.e. it is a constrained segment, but does not occur. As we assume that all segments are present in the mesh on initialization, none will be bad from the start.

Algorithm preserving isolated segments

```
1. Initialize $P = P_0$.
2. Compute $\mathrm{Del}(P)$ and if necessary restricted versions.
3. If there is a bad segment $s$:
    a) Choose $p$ midpoint of $s$
    b) Update $P = P \cup \{p\}$
    c) Return to item 2
4. If there is a bad surface triangle $t$:
    a) Choose $p \in \{V_t \cap \mathrm{Del}(P)_{|\partial O}\}$
    b) If $p$ encroaches on some segment $s$ to be preserved goto item 3a
    c) Update $P = P \cup \{p\}$
    d) Return to item 2
5. If there is bad tetrahedron $u$:
    a) Let $q$ be the circumcenter of $u$.
    b) If $q$ encroaches on some segment $s$ to be preserved goto item 3a
    c) If $q$ encroaches on some surface triangle $t$ goto item 4a
    d) Update $P = P \cup \{q\}$
    e) Return to item 2
6. Done.
```

Rule Set S (p. 30) denotes the set of refinement rules that corresponds to Rule Set R (p. 20). We expect it to be obvious that it is equivalent to the Algorithm preserving isolated segments (p. 29). Rule S1 is somewhat dummy, as it will never be triggered as shown in proof of theorem 3.1.2.1. We integrate it here nevertheless for two reasons: (1) Once we proved that it is not triggered, this implies that all segments are in the Delaunay triangulation throughout the whole run of the algorithm. (2) It serves as a placeholder for a refined strategy that follows in a succeeding step.

Rule Set S

S1 If a segment e is not in $\mathrm{Del}(P)$, refine e.

S2 If a facet f does not have its three vertices on ∂O or has a surface Delaunay ball $B(c, r)$ with ratio $\frac{r}{\sigma(c)} > \alpha$, then:

- **S2.1** if c is included in a segment diameter ball $B(c', r')$, insert c'
- **S2.2** else insert c

S3 If a tetrahedron t with circumcenter c has a circumradius r greater than $\sigma(c)$ or radius-edge ratio $\frac{r}{l_{\min}}$ greater than B, then

- **S3.1** if c is included in a segment diameter ball $B(c'', r'')$, insert its center c''
- **S3.2** else if c is included in a surface Delaunay ball $B(c', r')$ and c' is included in a segment diameter ball $B(c'', r'')$, insert c''
- **S3.3** else if c is included in a surface Delaunay ball $B(c', r')$ insert c'
- **S3.4** else insert c

Note that the set E has to be modified during the course of Algorithm preserving isolated segments (p. 29). Point insertions due to item 3a split a constrained segment. Precisely, for a segment s with endpoints a, b whose midpoint p is inserted, we will not be able to recover s in the course of the algorithm anymore. Instead we will aim at recovering its subsegments (a, p) and (p, b) resulting from the split. This splitting process might be iteratively repeated. If we can prove termination, then the initial segment $s \in E$ will be represented in the resulting Delaunay triangulation by this very argument: e is represented as the union of its subsegments and the same holds for each subsegment recursively.
We will denote E_0 the initial set and E_i the result of the i-th split of a segment. It holds $\forall i : \bigcup_{e \in E_0} e = \bigcup_{e' \in E_i} e'$.

3.1.2 Output guarantees and termination

To prove the validity of Rule Set S (p. 30), we proceed as follows: We show that the algorithm terminates. The quality that can be guaranteed for the resulting mesh follows from the restrictions on α and B that will be derived during the proof.

We define $d(e, p) := \min\{d(x, p) \mid x \in e\}$ and

$$l_e := \inf\{d(e_i, e_j) \mid e_i \cap e_j = \emptyset, e_i, e_j \in E\}$$

the minimal distance between to segments. For E being a finite set, $l_e = \min\{d(e_i, e_j) \mid e_i \cap e_j = \emptyset, e_i, e_j \in E\} > 0$. Note that unlike in $\mathbb{R}^2$ there might be no pair $a, b \in P_E$ of endpoints of these segments with $d(a, b) = l_e$ since the pair segments where the minimum is acquired might be skew.

To keep it simple, we make some additional assumptions at this point:

Asmp1 Constrained segments are within the boundary only, i.e. $E \subset \partial O$.
Asmp2 No two segments of E are incident, i.e. $\forall e, e' \in E : e \cap e' = \emptyset$.
Asmp3 For all $e \in E$: length$(e) < l_e$.

Asmp3 implies that $\forall e \in E$, the segment e is in Del(P_E).
While the latter two assumptions will actually be relaxed in later versions of the algorithm, dropping Asmp1 is only discussed theoretically.

We will prove an analogue to lemma 2.3.4.2, showing that the algorithm still terminates. It is a theorem in this this thesis for three reasons: (1) it is the main theoretical result proved in this thesis, (2) the set of rules is larger and exhibits more interrelations, and (3) the assumptions are more general than in the original lemma (esp. see remark 2.3.4.1).

Theorem 3.1.2.1. *For each $E \subset \partial O$ not exhibiting incident segments and suitable sizing field σ, there exist α, B such that the algorithm terminates.*

The proof will be split into three parts, corresponding to the rules S1, S2 and S3. The first lemma states that S1 will never be triggered at all and the second lemma is a small helping lemma for later proofs. The subsequent lemmas show that bounds necessary to prove theorem 3.1.2.1 can be derived. The part of the algorithm when S3 has not yet been triggered is called the *surface mesher*. From the first time at which S3 is triggered on, the algorithm is called *volume mesher*. Both parts are based on the same Rule Set S (p. 30). The proof of the theorem handles the whole rule set and concludes this section.

Lemma 3.1.2.1 (Rule S1 is never triggered.).

Proof. We prove this by induction. On initialization, by Asmp3, all constrained segments are shorter than their minimal distance. Hence no constrained segment can be encroached by another's endpoints. A segment is in the Delaunay triangulation if it is not encroached.

Suppose a point insertion due to Rule S2 or S3 inserts a point p that encroaches on a constrained segment e, which is subsegment of some $e_0 \in E_0$. Then p was inserted on a constrained segment. But p was certainly not inserted on a constrained segment e' which is a subsegment of the same original segment e_0. Instead e' is a subsegment of $e_1 \in E_0$. But then again the diametral ball of e has radius less than the distance $d(e_0, e_1)$ of these segments and hence p cannot encroach on e. □

Lemma 3.1.2.2 (Helping lemma)**.** *Let f, f' be two simplices of a Delaunay triangulation and $B(c,r), B(c',r')$ associated balls such that their boundaries pass through all the vertices of the resp. simplex exactly. Let furthermore $c \in B(c',r')$. Then $2r' \geq r$.*

Proof.

$$\begin{aligned}
2r' &= r' + r' \\
&= r' + d(c',p) && \text{where } p \text{ a vertex of } f' \\
&\geq d(c,c') + d(c',p) && \text{by assumption } c \in B(c',r') \\
&\geq d(c,p) && \text{by triangle inequality} \\
&\geq r && \text{because } f \text{ is a Delaunay simplex, no point } p \text{ is within } B(c,r)
\end{aligned}$$ □

Lemma 3.1.2.3 (Insertion radius of surface mesher)**.** *The following condition is verified if S3 has not yet been triggered*

$$\text{(E1)} \qquad \forall p \in P : r(p) \geq \frac{\sigma_1'(p)}{2+\alpha}$$

where $\alpha < 1$, $\delta(p)$ the Euclidean distance from p to ∂O and $\sigma_1'(p) := \min\{\alpha\sigma(p), \kappa_1\}$ with $\kappa_1 = \min\{d(p,q) \mid p,q \in P_0\}$ the shortest distance between two points in the finite initial point set.

Proof. We prove this lemma by induction. Initially E1 is veryfied by the choice of κ_1.

The rules of S2 cannot be applied for a facet having not having its vertices on the boundary because the initial point set has all vertices on ∂O. S2 inserts vertices only in the boundary also. S3, especially Rule S3.4 has not yet been triggered by assumption.

Suppose rule S2.1 is applied because $r > \alpha\sigma(c)$. Then E1 is satisfied because

$$\begin{aligned}
2r' &\geq r && \text{by lemma 3.1.2.2}\\
&\geq \alpha\sigma(c) && \text{by assumption}\\
&\geq \alpha\left(\sigma(c') - d(c,c')\right) && \sigma \text{ is 1-Lipschitz}\\
&\geq \alpha\left(\sigma(c') - r'\right) && c \text{ within } B(c',r')\\
r' &\geq \frac{\alpha\sigma(c')}{2+\alpha}\\
&\geq \frac{\sigma_1'(c')}{2+\alpha} && \alpha\sigma(p) \geq \sigma'(p)
\end{aligned} \tag{S2.1bSurf}$$

Suppose rule S2.2 is applied because $r > \alpha\sigma(c)$. Then E1 is satisfied.

$$\begin{aligned}
r &> \alpha\sigma(c) && \text{by assumption}\\
&\geq \sigma_1'(c) && \text{by defintion of } \sigma'
\end{aligned} \tag{S2.2bSurf}$$

□

Lemma 3.1.2.4 (Result of the surface mesher)**.** *The surface mesher terminates and the output point set is finite.*
Let P be the point set at any state of Rule Set S *(p. 30) where neither Rule S1 nor S2 are applied. If $\forall x \in \partial O : \sigma(x) \leq 0.09 d_M(x)$ then P is a loose 0.09-sample and theorem 2.3.4.2 holds. And $d(q, P_{|\partial O}) \leq 0.09 d_M(q)$*

Sketch of proof. The first two statements are equivalent in our setting. The distance of the points can be lower bounded because of lemma 3.1.2.3 where σ_1' is a positive function on a compact set. As all points are inserted in a compact set and the algorithm is greedy, there can be only finite points. Because one point is added at each iteration, the surface mesher terminates.

We rely on the corresponding results of Boissonnat and Oudot (2005) and Oudot et al. (2005). Our algorithm implies a sampling that is at least as dense as stated in these papers. □

Lemma 3.1.2.5 (Insertion radius of volume mesher)**.** $\exists C > 2 + \alpha$, $D > 1$ *such that the following conditions are verified*

$$
\begin{array}{lll}
\text{(F1)} & \forall p \in P: & r(p) \geq \dfrac{\sigma_2'(p)}{C} \\
\text{(F2)} & \forall p \in P \setminus P_{|\partial O}: & \delta(p) \geq \dfrac{1}{D(1-\gamma)}\sigma_2'(p)
\end{array}
$$

where $\delta(p)$ *is the Euclidean distance from* p *to* ∂O *and* $\sigma_2'(p) := \min\{\alpha\sigma(p), \kappa_2 \cdot 0.09\sigma_0(p)\}$ *with* $0 < 0.09\kappa_2 < \alpha < \kappa_2 < \min\{\frac{1}{3}, \kappa_1\}$.

Remark 3.1.2.1. The proof is an extension of the proof in Oudot et al. (2005). We try to employ the correct sparseness of the initial point set though (see remark 2.3.4.1).

Proof. We prove this lemma by induction. Until S3 is triggered for the first time, lemma 3.1.2.3 holds. Hence on initialization E1 holds and implies F1. Additionally F2 holds because all points are in ∂O.
Rule S1 cannot be applied by lemma 3.1.2.1. We have to check for F2 in rule S3.4 only.

We will bound the insertion radius of the actually inserted point. This might be either r, r' or r'' according to the rule applied, where certainly $r := r(c), r' := r(c')$ and $r'' := r(c'')$. Moreover we might refer to *virtual insertion radii*, i.e. the insertion radius theoretically assigned to a point if it would have been inserted instead of causing another insertion.

We will make use of the Lipschitz-continuity of σ_2', i.e. $\sigma_2'(x) - \sigma_2(y) \leq \gamma \cdot d(x, y)$ where $\gamma := \max\{\alpha, \kappa_2 \cdot 0.09\} = \alpha$ (sic!).

Suppose rule S2.1 is applied because the facet does not have its three vertices on ∂O.

Then F1 is satisfied after the insertion of c' because

$2r' \geq r$ by lemma 3.1.2.2

(3.1a)

$$
\begin{aligned}
&= d(c,p) && \text{choose } p \in P \setminus P_{|\partial O} \text{ vertex the facet}\\
&\geq \delta(p) && \text{because } c \in \partial O\\
&\geq \frac{1}{D(1-\gamma)}\sigma_2'(p) && \text{by F2}\\
&\geq \frac{1}{D(1-\gamma)}\left(\sigma_2'(c') - \gamma d(p,c')\right) && \sigma_2' \text{ is } \gamma\text{-Lipschitz}\\
&\geq \frac{1}{D(1-\gamma)}\left(\sigma_2'(c') - \gamma\left(d(p,c) + d(c,c')\right)\right) && \text{by triangle inequality}\\
&\geq \frac{1}{D(1-\gamma)}\left(\sigma_2'(c') - 3\gamma r'\right) && \text{using (3.1a) and } c \text{ within } B(c',r')
\end{aligned}
$$

(S2.1a)

$$r' \geq \frac{\sigma_2'(c)}{2D(1-\gamma)+3\gamma}$$

Suppose rule S2.1 is applied because $r > \alpha\sigma(c)$. Then F1 is satisfied because

(S2.1b) $$r' \geq \frac{\sigma_2'(c')}{2+\alpha}$$ as in eq. (S2.1bSurf)

Suppose rule S2.2 is applied because the facet does not have its three vertices on ∂O. Then F1 is satisfied because

$$
\begin{aligned}
r &= d(c,p) && p \in P \setminus P_{|\partial O}\\
&\geq \delta(p) && \text{because } c \in \partial O\\
&\geq \frac{1}{D(1-\gamma)}\sigma_2'(p) && \text{by F2}\\
&\geq \frac{1}{D(1-\gamma)}\left(\sigma_2'(c) - \gamma d(c,p)\right) && \sigma_2' \text{ is } \gamma\text{-Lipschitz}
\end{aligned}
$$

(S2.2a) $$r \geq \frac{\sigma_2'(c)}{D(1-\gamma)+\gamma}$$

as in Oudot et al. (2005), Lemma 6, proof for Rule S1.

Suppose rule S2.2 is applied because $r > \alpha\sigma(c)$. Then F1 is satisfied.

(S2.2b) $\quad r > \sigma_2'(c) \quad$ as in eq. (S2.2bSurf)

Suppose rule S3.1 is applied because $r > \sigma(c)$. Then

$$\begin{aligned}
2r'' &\geq r && \text{by lemma 3.1.2.2}\\
&> \sigma(c) && \text{by assumption}\\
&> \alpha\sigma(c) && \text{choose } \alpha < 1\\
&\geq \sigma_2'(c) && \text{by definition of } \sigma_2'\\
&\geq \sigma_2'(c'') - \gamma d(c, c'') && \sigma_2' \text{ is } \gamma\text{-Lipschitz}\\
&\geq \sigma_2'(c'') - \gamma r'' && c \text{ within } B(c'', r'')
\end{aligned}$$

$$r'' \geq \frac{\sigma_2'(c'')}{2+\gamma} \tag{S3.1a}$$

Suppose rule S3.1 is applied because $\frac{r}{l_{\min}} > B$. Then

$$\begin{aligned}
r &> Bl_{\min} && \text{by assumption}\\
&\geq Br(p) && p \text{ being an endpoint of the shortest edge}\\
&\geq \frac{B}{C}\sigma_2'(p) && \text{by induction of F1}\\
&\geq \frac{B}{C}\left(\sigma_2'(c) - \gamma d(p, c)\right) && \sigma_2' \text{ is } \gamma\text{-Lipschitz}\\
&\geq \frac{B}{C}\left(\sigma_2'(c) - \gamma r\right) && \text{by choice of } p \text{ as vertex of tetrahedron } t
\end{aligned}$$

$$r \geq \frac{B}{C+\gamma B}\sigma_2'(c) \tag{3.1b}$$ an. to Oudot et al. (2005) up to C

$$\begin{aligned}
2r'' &\geq r && \text{by lemma 3.1.2.2}\\
&\geq \frac{B}{C+\gamma B}\sigma_2'(c) && \text{using (3.1b)}\\
&\geq \frac{B}{C+\gamma B}\left(\sigma_2'(c'') - \gamma r''\right) && \sigma_2' \text{ is } \gamma\text{-Lipschitz}
\end{aligned}$$

$$r'' \geq \frac{B}{2C+3\gamma B}\sigma_2'(c'') \tag{S3.1b}$$

Suppose rule S3.2 is applied because $r > \sigma(c)$. Then

$$
\begin{aligned}
& & 2r' &\geq r && \text{by lemma 3.1.2.2} \\
& & &> \sigma(c) && \text{by assumption} \\
& & &> \sigma_2'(c) && \text{by definition of } \sigma_2' \\
& & &\geq \sigma_2'(c') - \gamma d(c, c') && \sigma_2' \text{ is } \gamma\text{-Lipschitz} \\
& & &\geq \sigma_2'(c') - \gamma r' && c \text{ within } B(c', r') \\
&(3.1c) & r' &\geq \frac{1}{2+\gamma}\sigma_2'(c') && \\
& & 2r'' &\geq r' && \text{by lemma 3.1.2.2} \\
& & &\geq \frac{1}{2+\gamma}\left(\sigma_2'(c'') - \gamma d(c', c'')\right) && \text{using (3.1c) and } \sigma_2' \text{ being } \gamma\text{-Lipschitz} \\
& & &\geq \frac{1}{2+\gamma}\left(\sigma_2'(c'') - \gamma r''\right) && c' \text{ within } B(c'', r'') \\
&(S3.2a) & r'' &\geq \frac{1}{4+3\gamma}\sigma_2'(c'') &&
\end{aligned}
$$

Suppose rule S3.2 is applied because $\frac{r}{l_{\min}} > B$. Then

$$
\begin{aligned}
& & 2r' &\geq r && \text{by lemma 3.1.2.2} \\
& & &\geq \frac{B}{C+\gamma B}\sigma_2'(c) && \text{in analogy to (3.1b)} \\
& & &\geq \frac{B}{C+\gamma B}\left(\sigma_2'(c') - \gamma r'\right) && \sigma_2' \text{ is } \gamma\text{-Lipschitz} \\
&(3.1d) & r' &\geq \frac{B}{2C+3\gamma B}\sigma_2'(c') && \text{in analogy to(S3.1b)} \\
& & 2r'' &\geq r' && \text{by lemma 3.1.2.2} \\
& & &\geq \frac{B}{2C+3\gamma B}\sigma_2'(c') && \text{using (3.1d)} \\
& & &\geq \frac{B}{2C+3\gamma B}\left(\sigma_2'(c'') - \gamma r''\right) && \sigma_2' \text{ is } \gamma\text{-Lipschitz} \\
&(S3.2b) & r'' &\geq \frac{B}{4C+7\gamma B}\sigma_2'(c'') &&
\end{aligned}
$$

Suppose rule S3.3 is applied because $r > \sigma(c)$. Then

$$\begin{aligned}
2r' &\geq r && \text{by lemma 3.1.2.2}\\
&\geq \sigma(c) && \text{by assumption}\\
&\geq \sigma_2'(c) && \text{by definition of } \sigma_2'\\
&\geq \sigma_2'(c') - \gamma d(c,c') && \sigma_2' \text{ is } \gamma\text{-Lipschitz}\\
&\geq \sigma_2'(c') - \gamma r' && c \text{ within } B(c',r')
\end{aligned}$$

$$\text{(S3.3a)} \qquad r' \geq \frac{\sigma_2(c')}{2+\gamma} \qquad \text{in analogy to (3.1c)}$$

Suppose rule S3.3 is applied because $\frac{r}{l_{\min}} > B$. Then

$$\text{(S3.3b)} \qquad r' \geq \frac{B}{2C+3\gamma B}\sigma_2'(c') \qquad \text{in perfect analogy with (S3.1b)}$$

Suppose rule S3.4 is applied. Analogously to Oudot et al. (2005), Lemma 6, proof of Rule R3.1

$$\text{(S3.4a)} \qquad r \geq \sigma_2'(c) \qquad \text{for } r > \sigma(c) \text{ using } \sigma \geq \sigma_2'$$

$$\text{(S3.4b)} \qquad r \geq \frac{B}{C+\gamma B}\sigma_2'(c) \qquad \text{for } \frac{r}{l_{\min}} > B \text{ using (3.1b)}$$

In analogy with Lemma 6, proof of Rule S3.1 in Oudot et al. (2005), we check F2. Note that eq. (S3.4b) $\Rightarrow$ eq. (S3.4a) because of the choice of α: $\frac{1}{\alpha} \geq \frac{B}{C+\gamma B} \Rightarrow \sigma(c) \geq \frac{B}{C+\gamma B}\sigma_2'(c)$ and hence

$$\begin{aligned}
\delta(c) &= d(c,q) && q \in \partial O \text{ closest to } c\\
&\geq d(c, P_{|\partial O}) - d(q, P_{|\partial O}) && \text{by triangle inequality}\\
&\geq r - d(q, P_{|\partial O}) && \text{no point within } B(c,r)\\
&\geq \frac{B}{C+\gamma B}\sigma_2'(c) - d(q, P_{|\partial O})\\
\text{(3.1e)} \qquad &\geq \frac{B}{C+\gamma B}\sigma_2'(c) - \frac{1}{\kappa_2}\sigma_2'(q)\\
&\geq \frac{B}{C+\gamma B}\sigma_2'(c) - \frac{1}{\kappa_2}\left(\sigma_2'(c) + \gamma d(c,q)\right) && \sigma_2' \text{ is } \gamma\text{-Lipschitz}
\end{aligned}$$

$$\text{(S3.4c)} \qquad d(c,q) \geq \frac{1}{\gamma+\kappa_2}\left(\frac{B\kappa_2}{C+\gamma B} - 1\right)\sigma_2'(c)$$

Justifying eq. (3.1e). The goal is to bound $d(q, P_{|\partial O}) \leq \frac{1}{\kappa_2}\sigma_2'(q)$. When rule S3 is triggered, S2 is fulfilled. This implies lemma 3.1.2.4, saying that theorem 2.3.4.2 holds. Hence, there is a surface Delaunay ball $B(c', r')$ containing q and through some $p \in P$ yielding $d(q, P_{|\partial O}) \leq d(q, p)$. Because $p, q \in B(c', r')$, $d(q, p) \leq 2r'$

$$\begin{aligned}
r' &\leq \alpha\sigma(c') && \text{Rule S2 not applied for this surface facet}\\
&\leq \alpha\left(\sigma(q) + d(c', q)\right) && \sigma \text{ is 1-Lipschitz}\\
&\leq \alpha\left(\sigma(q) + r'\right) && q \in B(c', r')\\
r' &\leq \frac{\alpha}{1-\alpha}\sigma(q) && \text{(3.1f)}
\end{aligned}$$

Hence

$$\begin{aligned}
d(q, P_{|\partial O}) &\leq d(q, p)\\
&\leq 2r'\\
&\leq \frac{2\alpha}{1-\alpha}\sigma(q) && \text{using eq. (3.1f)}\\
&\leq \frac{1}{\kappa_2}\alpha\sigma(q) && \text{by choice of } \kappa_2\\
d(q, P_{|\partial O}) &\leq 0.09 d_M(q) && \text{lemma 3.1.2.4}\\
&= \frac{1}{\kappa_2}\left(\kappa_2 \cdot 0.09\sigma_0(q)\right) && \text{by def. } \sigma_0 \text{ and } d_M \text{ coincide on the boundary}
\end{aligned}$$

Because of the definition of $\sigma_2'(x) = \min\{\alpha\sigma(x), \kappa_2 \cdot 0.09\sigma_0(x)\}$, we conclude $d(q, P_{|\partial O}) \leq \frac{1}{\kappa_2}\sigma_2'(q)$. **End of justification of eq. (3.1e).**

In order to fulfill (S3.4c) we proceed as in Oudot et al. (2005), Lemma 6, proof of Rule R3.1 but with fundamentally different constants

$$\begin{aligned}
\frac{1}{\gamma + \kappa_2}\left(\frac{B\kappa_2}{C + \gamma B} - 1\right) &\geq \frac{1}{D(1-\gamma)}\\
\Leftrightarrow D(1-\gamma)(B\kappa_2 - C - \gamma B) &\geq (\gamma + \kappa_2)(C + \gamma B)\\
\Leftrightarrow D &\geq \frac{(\gamma + \kappa_2)(C + \gamma B)}{(1-\gamma)(B\kappa_2 - C - \gamma B)} && \text{(S3.4d)}
\end{aligned}$$

requiring $B \geq \frac{C}{\kappa_2 - \gamma}$.

Putting all the results together, we conclude that F1 is fulfilled if all of the above bounds can be lower bounded by $\frac{1}{C}\sigma_2'(c)$ (or $\sigma_2'(c'), \sigma_2'(c'')$ respectively). Below we list the upper bounds for $\frac{1}{C}$ imposed by this condition and try to reduce the set:

(3.2a)	1	(S2.2b), (S3.4a)
(3.2b)	$\frac{1}{2+\gamma}$	(S2.1b), (S3.1a), (S3.3a)
(3.2c)	$\frac{1}{4+3\gamma}$	(S3.2a)
(3.2d)	$\frac{1}{D(1-\gamma)+\gamma}$	(S2.2a)
(3.2e)	$\frac{1}{2D(1-\gamma)+3\gamma}$	(S2.1a)
(3.2f)	$\frac{B}{C+\gamma B}$	(S3.4b)
(3.2g)	$\frac{B}{2C+3\gamma B}$	(S3.1b), (S3.3b)
(3.2h)	$\frac{B}{4C+7\gamma B}$	(S3.2b)

As a first simplification we can omit some conditional equations because

$$eq.\ (3.2c) \Rightarrow eq.\ (3.2b) \Rightarrow eq.\ (3.2a)$$

$$eq.\ (3.2e) \Rightarrow eq.\ (3.2d)$$

$$eq.\ (3.2h) \Rightarrow eq.\ (3.2g) \Rightarrow eq.\ (3.2f)$$

Further simplifying the set by we can reduce more constraints because they are implied by others. Finally we choose γ, B, C, D such that they satisfy the following set of equations:

$$C \geq 4 + 7\gamma \tag{3.3a}$$

$$C \geq 2D(1-\gamma) + 3\gamma \tag{3.3b}$$

$$B > \frac{C}{\kappa_2 - \gamma} \tag{3.3c}$$

$$D \geq \frac{(\gamma + \kappa_2)(C + \gamma B)}{(1-\gamma)(B\kappa_2 - C - \gamma B)} \tag{3.3d}$$

where the choice of $\gamma, \kappa_2, B, C, D$ only depends on the domain O and the inital segments E. □

We can interpret the last set of equations. We want to choose γ as large as possible in order to have less point insertions due to rule S2. However such a choice increases the lower bound on B and C significantly. Note that we did not show that the algorithm does not terminate for values not respecting these bounds. The proof also applies if no segments are constrained. Hence our bounds cannot be tight, as the bounds given in lemma 2.3.4.2 are essentially smaller.

Proof of theorem 3.1.2.1. We compose lemma 3.1.2.4 and lemma 3.1.2.5. We follow the results from Oudot et al. (2005):
We can lower bound the distance of two points in the resulting point set by the use of σ_2', because it is γ-Lipschitz. Balls can be assigned to each point such they do not mutually intersect. Further there is a lower bounded ratio of these balls contained in the domain O. Since O is compact there can only be finite points. $\square$

3.2 Preparing arbitrary segments

This section introduces the approach to handle arbitrary segment configurations. Several assumptions were made in section 3.1 on the set of constrained segments E to show termination of the algorithm corresponding to Rule Set S (p. 30). We provide a stand-alone algorithm that has no restrictions on the input set while keeping in mind that it should fit the rule set strategy later on.

Let E' be a finite set of segments. For the segments to occur in a Delaunay triangulation, they must respect the properties of a simplicial complex (see definition 2.2.1.3). We resolve the inconsistencies by deriving a minimal complex C such that $\forall e \in E' \; \exists D \subset C : e = \bigcup_{d \in D} d$. Its construction is straightforward: Compute all intersections in E' and split the segments accordingly. Let the result be the set E of 1-dimensional faces of C. Then add all endpoints of the segments of E, P_E, which is the minimal set of 0-dimensional faces we need to add in order to have a valid complex structure.

We will propose a greedy algorithm that subsequently splits encroached segments into subsegments in order to recover them all in the Delaunay triangulation. Our goal is to derive a point set $P_n \supset P_E$ such that $\forall e \in E \; \exists Q \subset \mathrm{Del}(P_n) : \bigcup_{q \in Q} q = e$, i.e. all segments of E occur in the Delaunay triangulation of P_n in the conforming sense. It is sufficient to show that no segment is encroached in E_n. We call $E_0 := E, P_0 := P_E$ the initial set of segments resp. points and E_i, P_i the sets after the i-th point insertion.

In section 3.1, we introduced the notion of encroachment for a constrained segment in order not to inadvertently insert points in its diametral ball that extinct the segment from the Delaunay triangulation. Now we are in a different setting, because it cannot be assumed that the segments are in the initial Delaunay triangulation for arbitrary E and corresponding point set P_E. We generalize our definition as a tribute to this fact: A segment $e \in E$ is said to be *encroached* if it is not present in $\mathrm{Del}(P_E)$ or if there is a point in P_E that lies in its diametral sphere.

Segments that share a point may introduce difficulties. If we handle them with a primitive strategy, infinite recurrences might occur. Figure 3.2 shows such an example.

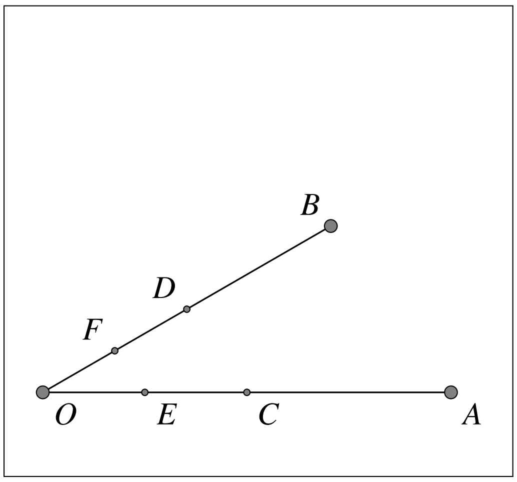


Figure 3.2: A small angle might cause infinite refinements in a split-at-half-if-encroached strategy. Points $C, D, E, F, \dots$ are inserted. Because each split of one a subsegment encroaches on the segment incident at O, this process might never terminate, depending on the angle and the length ratio.

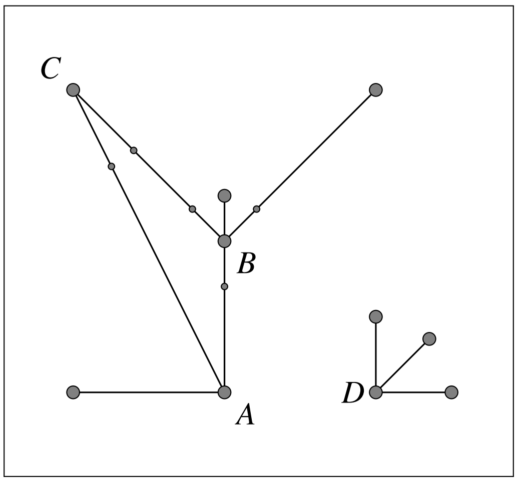


Figure 3.3: The type of star vertices in P and P_i. Small points are in $P_i \setminus P$. A, B, C, D are star vertices because they all have at least two adjacent points in P. The other points are non-star vertices. D is free because it is neither adjacent to any other star vertex nor do the incident segments differ in length. A is non-free, B, C are free in P_i.

As a tribute to the observation in fig. 3.2, we slightly modify the strategy at vertices where segments join. The vertices $p \in P_E$ that are endpoints of at least two constrained segments on initialization, are called *star vertices*. A star vertex p is said to be *free* in

E_i if all its adjacent vertices are non-star vertices and all its incident segments have the same length (see fig. 3.3)

3.2.1 Algorithm

As an initialization step, our algorithm has to recover the constrained segments. Although it fitting the Delaunay refinement scheme of the overall meshing approach, we will describe and analyse it separately first and then put the parts together for convenience. Again our algorithm will subsequently add points and recompute the Delaunay triangulation. We initialize $P_0 = P_E$ and call P_i, E_i the set of vertices resp. constrained segments after the i-th point insertion.

For each star vertex $p_i \in P$ let $l_{\min,i}$ be the length of its shortest incident constrained segment in E.

The strategy to conform to E is to loop over this set of rules:

Rule Set Q

- **Q1** If there is a non-free star vertex p_i, insert points on all segments incident to p at distance $\frac{l_{\min,i}}{3}$ from p.
- **Q2** If some encroached e is incident to a star-vertex p_i, insert the midpoints of all segments incident to p_i.
- **Q3** If e is encroached, insert the midpoint of e.

Remember l_e to be defined as the minimal distance of two non-incident segments in E. We refine this definition using e_k to access the respective value after the k-th point insertion. We additionally need the minimal angle of all incident segments and define

$$\begin{aligned} e_k &:= \min\{d(e_i, e_j) \mid e_i \cap e_j = \emptyset, e_i, e_j \in E_k\} \\ l_{\min} &:= \min\{l_{\min,i}\} \\ \alpha_k &:= \min\{\measuredangle(e_i, e_j) \mid e_i \cap e_j \neq \emptyset, e_i, e_j \in E_k\} \end{aligned}$$

Again all these can be expressed in minima as long as E_k and P_k are finite. If $\alpha_k > \frac{\pi}{3}$ or undefined, set $\alpha_k := \frac{\pi}{3}$.

3.2.2 Termination

We will prove termination by deriving a lower bound on the distance of two points in the resulting point set P_n. We will exploit the strategy to handle star vertices separately.

Lemma 3.2.2.1 (Encroachment at star vertices)**.** *Suppose all star vertices are free. Let e be a segment incident to a star vertex q and encroached by p. Then p is not on a segment incident to q.*

Proof. Suppose p was on an segment e' incident to q. By assumption q is free, i.e. $|e| = |e'|$ they have equal length l. Let r the midpoint of e and $\measuredangle(e, e') = \gamma$, see fig. 3.4. Then

$$\begin{aligned} d(p,r)^2 &= d(q,r)^2 + d(q,p)^2 - 2\,d(q,r)\,d(q,p)\cos\gamma \\ &= \frac{l^2}{4} + l^2 - l^2\cos\gamma \\ &= \frac{l^2}{4}(5 - 4\cos\gamma) \end{aligned}$$

and hence p encroaches on e iff $\frac{l^2}{4} > d(p,r)^2 = \frac{l^2}{4}(5 - 4\cos\gamma) \geq \frac{l^2}{4}$ which is a contradiction. □

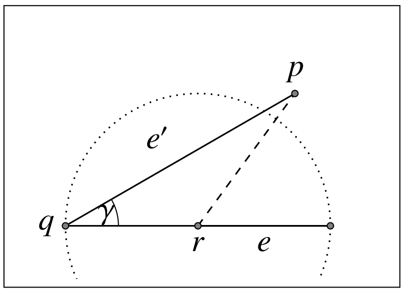


Figure 3.4: An endpoint p of a segment e' cannot encroach on an equally long incident segment e if they meet at a free star vertex q.

Lemma 3.2.2.2 (Lower bound the distance in P_n)**.** *Looping over Q1, Q2, Q3 in this very order does not split the segments to arbitrarily small length:*

$$\exists r > 0\,\forall p \in P_n : d\,(p, P \setminus \{p\}) \geq r$$

Proof. Let P_k be the set of points when rule Q1 cannot be applied any more for the first time. It will never be applied again during the course of the algorithm because there are only finitely many star vertices in P_0 and each provocation of Q1 frees a star vertex.

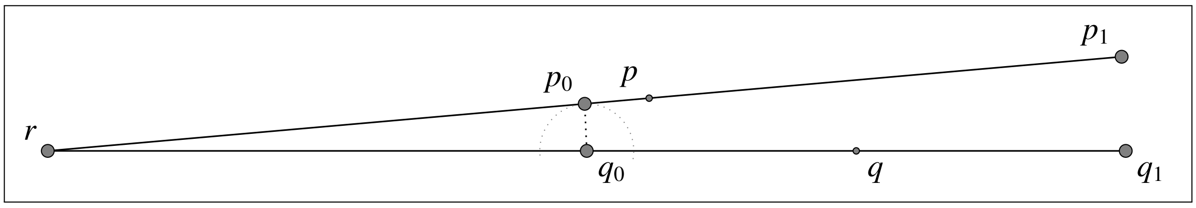


Figure 3.5: Points p, q lie on segments e_p, e_q that are subsegments of two incident segments. p and q are not incident to the star vertex r. Hence there are points p_0, q_0 inserted due to rule Q1 or Q2 lower bounding their distance $d(p,q) > d(p_0,q_0)$.

Rules Q2, Q3 do never *unfree* a star vertex again. Their adjacency has been resolved by Q1 and further refinement of a segment incident to a star vertex can only be imposed by Q2 which preserves the length equality of the incident segments.

Thus **Q1** is provoked only finite times. Because the maximal degree of a star vertex is bounded (and does not change when splitting segments), only finitely many points are in P_k. And in E_k all star vertices are free whenever Q2 or Q3 is triggered.

Q2. Let q be a point inserted due to Q2. Let P_i the set before the provocation of Q2, P_j the set after all points around the concerned star vertex are inserted, and $P_d = P_j \setminus P_i$ the set of points inserted due to Q2 in this step. Then $d(q, P_j \setminus \{q\}) = \min\{d(q,P_i), d(q, P_d \setminus \{q\})\}$ which we will bound separately.
All star vertices are free. By lemma 3.2.2.1 the point $p \in P_i$ encroaching on the segment where q is inserted cannot be adjacent to the relevant star vertex r. Hence $d(q,P_i) \geq \min\{e_k, d(q,r)\}$. All segments at the star vertex had the same length $l_{\text{old}} = 2d(q,r)$ and because one of it has been encroached by p it holds that $\frac{l_{\text{old}}}{2} \geq e_k$. Finally $d(q,P_i) \geq e_k$.
Let $p, q \in P_d$ with respective segments e_p, e_q of length l joining at star vertex r. Then $d(q,p)^2 = d(p,r)^2 + d(r,q)^2 - 2 \cdot d(p,r) \cdot d(r,q) \cos\measuredangle(e_p,e_q) = 2l^2(1 - \cos\measuredangle(e_p,e_q))$. Furthermore p, q adjacent to a star vertex where one of its incident segments of length $l_{\text{old}} = 2l$ has been encroached by a point of P_i. Hence $\frac{l_{\text{old}}}{2} \geq e_k$ and $\cos\measuredangle(e_p,e_q) \leq \cos\alpha_k$.
In summary $d(q,p)^2 \geq \frac{e_k^2}{2}(1-\cos\alpha_k)$ and thus $d(q, P_d \setminus \{q\}) \geq e_k\sqrt{\frac{1}{2}(1-\cos\alpha_k)}$.

Q3. Let q be a point inserted to P_i due to Q3. Then $d(q,P_i) \geq \frac{e_k}{2}$ as long as we can assure that the point p encroaching on q is not on a segment $e_p \in E_k$ that was incident to a segment $e_q \in E_k$ containing q in P_k. Suppose this was the case now, see fig. 3.5. Because we are in rule Q3, not in rule Q2, p, q themselves cannot be incident to the imagined star vertex. Hence their distance is lower bounded by the results of Q2, i.e. $d(p,q) \geq e_k\sqrt{\frac{1}{2}(1-\cos\alpha_k)}$.

Certainly $\alpha_k = \alpha_0$. To lower bound e_k we have to consider the distance between segments that are no more incident due to the application of Q1. Their distance can be lower bounded by the points introduced in Q1 because they are not skew. Let $P_d = P_k \setminus P_0$ the set of points introduced due to Q1. Then

$$\begin{aligned}
e_k &= \min_{q \in P_d}\{e_0, d(q, P_0), d(q, P_d \setminus \{q\})\} && \text{where} \\
d(q, P_0) &\geq \min\{e_0, l_{\min}\} && \\
d(q, P_d \setminus \{q\}) &\geq \sqrt{2 l_{\min}(1 - \cos\alpha_0)} && \text{and hence} \\
e_k &\geq \min\{e_0, l_{\min}, \sqrt{2 l_{\min}(1 - \cos\alpha_0)}\}. &&
\end{aligned}$$

Choose

$$\begin{aligned}
r &= \min\left\{\frac{e_k}{2}, e_k\sqrt{\frac{1}{2}(1 - \cos\alpha_k)}\right\} && \\
&= e_k\sqrt{\frac{1}{2}(1 - \cos\alpha_k)} && \text{by definition of } \alpha_k.
\end{aligned}$$

□

Lemma 3.2.2.3 (Termination of the conformation)**.**

Proof. E is compact and the algorithm is greedy and inserts in E only. The balls $\left(B(p, \frac{r}{2})\right)_{p \in P_n}$ are pairwise disjoint. Hence P_n is finite. □

As a result, no segment $e \in E_n$ can be encroached because of Q3. As denoted earlier, this implies that (but is not equivalent to) all constrained segments in E_n occur in the Delaunay triangulation $\mathrm{Del}(P_n)$.

3.3 Extended mesh generation algorithm

In this section we will merge the preparatory algorithm of section 3.2 and the volumetric mesh generation approach of section 3.1. The properties of the result from the preparatory algorithm and the input requirements of the mesh generation method are gradually adopted in order to cascade them. The separate algorithms' borders are dissolved, yielding our final algorithm.

3.3.1 Relaxing previous assumptions

Dropping Asmp3. In section 3.1 we assumed that all segments are smaller than their minimal distance. We used this assumption for two implications: (1) The constrained segments are Delaunay elements on initialization and (2) splitting a constrained segment does not encroach on another.

If we prepend the application of Rule Set Q (p. 43) to Rule Set S (p. 30), then property (1) is satisfied. We design extended rule sets of Rule Set S (p. 30) that maintain the properties that were essential for the proofs in section 3.1 but do not assume (2).

Rule Set S+

S1 If a constrained segment e is not shorter than l_e, insert its midpoint
S2 as before
S3 as before

Rule Set S+ (p. 47) is no different from Rule Set S (p. 30) apart from a finite refinements due to rule S1 before the requirements of Asmp3 are met and the proofs of section section 3.1 apply. The number of points added by the application of S1 is upper bounded by $k \cdot 2^{n+1}$ where $k = |E|$ and $n = \lceil \log_2 \frac{l}{l_e}$, l upper bounding the length of the segments of E. Figure 3.6 shows that the strategy inserts unnecessarily many points and we propose the more adaptive Rule Set S++ (p. 48).

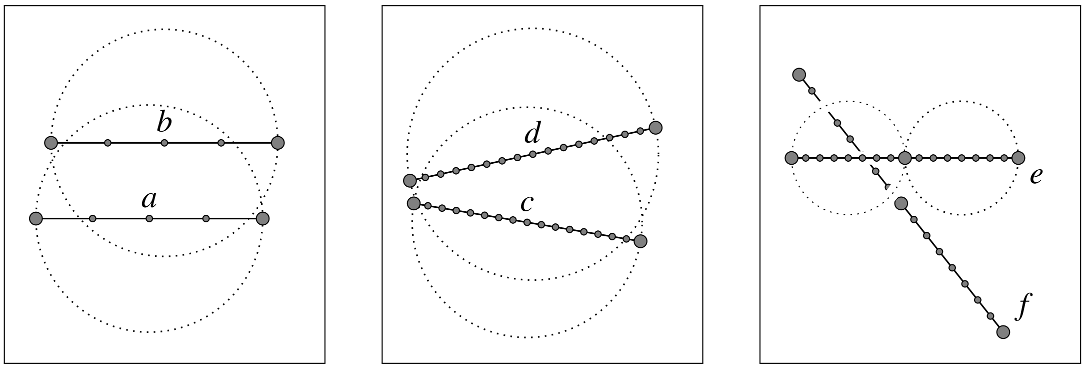


Figure 3.6: Superfluous refinements in the conformation step. Planar cases (left, center): Segments a, b, c, d are not encroached, hence would be in the Delaunay triangulation of the large points. The algorithm inserts the small points nevertheless to have all segments smaller than the minimal distance. Skew case (right): Skew segments e, f are no more encroached after midpoints have been inserted, algorithm inserts the small points nevertheless.

Rule Set S++

S1 If a constrained segment e is encroached, insert its midpoint
S2 as before
S3 as before

For Rule Set S++ (p. 48) it looks as if it generates only a subset of points of the result of Rule Set S (p. 30). However we cannot guarantee that because the Delaunay triangulations differ and the point insertions depend thereon. However the property we have to maintain is the bounded insertion radius. We did not exploit Asmp3 in the proofs concerning S2 and S3. We replace RuleS1 (p. 31).

Lemma 3.3.1.1 (Termination with S++)**.** *Theorem 3.1.2.1 applies for* Rule Set S++ *(p. 48) also, replacing the definition of κ_1 in lemma 3.1.2.3 with* $\kappa_1 = \min_{p,q \in P_0}\{l_e, d(p,q)\}$.

Proof. If no constrained segment is encroached after point is inserted due to rule S2 or S3, nothing has to be shown.
If a constrained segment e is encroached by a point p inserted due to rules S2 or S3, then p itself is on a constrained segment e'. e' is not subsegment of the same segment as e. Further the point q to be inserted in e has insertion radius $d(p,q)$, because no other point is within the diameter ball of e. We can bound $d(p,q) \leq l_e$.
If a constrained segments is encroached by a point inserted due to rule S1, the same holds.

In lemma 3.1.2.3 the induction hypothesis holds and hence the theorem follows. □

Relaxing Asmp2. We relax the incidence assumption. In Rule Set S+ (p. 47) incident constrained segments might induce point insertions due to rule S1. We briefly review why and emphasise that the results of these observations have already been considered in when designing the preparatory algorithm (Rule Set Q (p. 43)) in section 3.2:

Let e_1, e_2 be two constrained segments incident at v and $\measuredangle(e_1, e_2) = \alpha$. If $\alpha \geq \frac{\pi}{2}$. Then no point on e_1 can ever encroach e_2 and conversely because e_1 is entirely outside of the diametral sphere of e_2.
If $\frac{\pi}{2} > \alpha \geq \frac{\pi}{3}$, then this is no longer true. With the preparatory step done we can assume that e_1, e_2 have the same length. If a facet (S2) or tetrahedron rule (S3) triggers

the refinement of, say e_1, then a refinement of e_2 will reassure this property. In theorem 3.1.2.1 showing termination of Rule Set S (p. 30) we assumed that rule S1 is not triggered after the preparatory step. This property has been relaxed Rule Set S++ (p. 48) for isolated constrained segment configurations. Now rule S1 might be provoked also for constrained segments incident to e_1. For the point p_1 inserted in e_1 we proved that it satisfies condition D1 and D2 of theorem 3.1.2.1, i.e. its insertion radius is lower bounded. Because e_2 was not encroached before e_1 was refined, there can be no point in its diametral sphere apart from p_1. In addition $r(p_1) = d(p_1, v) = d(p_2, v)$ and by the angle assumption $d(p_1, p_2) \geq r(p_1)$. Hence $r(p_2)$ has the very same insertion radius. The same is true for all other points inserted around v due to the insertion of p_1 (as long as all angles that link the insertions are not smaller than $\frac{\pi}{3}$). We proceed with a short note on the possible amount of insertions triggered in this configuration. Note that there might be (finite) interrelations with non-incident segments also.
For a ball with radius r the volume of a spherical sector is $\frac{2}{3}\pi r^3(1 - \cos\frac{\gamma}{2})$ where γ the maximal angle of the cone, The small computation

$$\begin{aligned} n\frac{2}{3}\pi r^3(1 - \frac{\sqrt{3}}{2}) &\leq \frac{4}{3}\pi r^3 \\ n &\leq \frac{4}{2 - \sqrt{3}} \\ &\approx 14.9 \end{aligned}$$

shows that there can be at most 14 non-intersecting spherical sectors of angle $\geq \frac{\pi}{3}$ which is hence a bound for the number of points inserted around v due to p_1.
As a matter of fact, splitting one of the incident segments of equal length does not encroach the other segment. Further refinements of the part being incident to the star vertex will eventually encroach on the other segment - a property that will be exploited later. We state it here as a

Lemma 3.3.1.2 (Angle depending mutual encroachment of equally long incident segments)**.** *Let a, b two incident segments of length l with midpoints m_a, m_b and* $0 < \measuredangle(a, b) = \alpha < \frac{\pi}{2}$. *Then*

$$\alpha \geq \frac{\pi}{3} \iff m_a \text{ does not encroach on } b.$$

Proof. m_a encroaches on b iff $d(m_a, m_b) < \frac{l}{2}$. By the law of cosine $d(m_a, m_b)^2 = \frac{l^2}{2}(1 -$

$\cos\alpha$), yielding

$$d(m_a, m_b) < \frac{l}{2} \Leftrightarrow \frac{l^2}{2}(1 - \cos\alpha) < \frac{l^2}{4}$$
$$\Leftrightarrow \alpha < \frac{\pi}{3}$$

Furthermore the line containing a is a secant to the diametral sphere of b. □

Difficulties arise for $\alpha < \frac{\pi}{3}$. Not only is it true that a refinement of e_1 ultimately encroaches e_2 by lemma 3.3.1.2, but also the insertion radius of points inserted after p_1 is upper bounded by their mutual distance, which depends on the length of the original segments and α. This necessarily crashes the termination proof given for theorem 3.1.2.1. We will discuss this later.

We propose Rule Set T (p. 50) as the composition of Rule Set Q (p. 43) and Rule Set S++ (p. 48). A constrained segment is called *bad*, if (1) it is incident to a star vertex p_i and has length larger than $\frac{l_{\min,i}}{3}$, (2) it is incident to a star vertex and is longer than some other segment incident to the same star vertex, or (3) if it is encroached.

Rule Set T

T1 If a constrained segment e is bad

- **T1.1** if e is incident to a star vertex p_i and has length $> \frac{l_{\min,i}}{3}$, insert the point at distance $\frac{l_{\min,i}}{3}$ from p_i on e
- **T1.2** else insert the midpoint of e

T2 If a facet f does not have its three vertices on ∂O or has a surface Delaunay ball $B(c, r)$ with ratio $\frac{r}{\sigma(c)} > \alpha$, then:

- **T2.1** if c is included in a segment diameter ball $B(c', r')$, insert c'
- **T2.2** else insert c

T3 If a tetrahedron t with circumcenter c has a circumradius r greater than $\sigma(c)$ or radius-edge ratio $\frac{r}{l_{\min}}$ greater than B, then

- **T3.1** if c is included in a segment diameter ball $B(c'', r'')$, insert its center c''
- **T3.2** else if c is included in a surface Delaunay ball $B(c', r')$ and c' is included in a segment diameter ball $B(c'', r'')$, insert c''
- **T3.3** else if c is included in a surface Delaunay ball $B(c', r')$ insert c'
- **T3.4** else insert c

We briefly argue that this indeed is nothing but a composition:

- For all non-free star vertices, Q1 inserts points on all segments incident to p_i at distance $\frac{l_{\min}}{3}$ from p. T1.1 mimics this because all non-free star vertices have at least one bad incident segment. The order of point insertions is extraneous because it does not depend on the Delaunay triangulation but only on the initial point set and E. If Q1 is fulfilled (T1.1 has been applied finite times), T1.1 will never be applied again.
- For Q2 and Q3 we cannot explicitly state that the order of point insertions is equivalent to those performed by T1.2 but we claim that the results are. Both rules of Rule Set Q (p. 43) insert midpoints of segments, so does T1.2. By the adapted notion of a bad segment the same cases are considered. The resulting point sets are the same basically because the fact that a point encroaches on a segment cannot be abolished by another point insertion than the segment splitting. As a consequence, the point sets at the time, when T1 is satisfied for the first time and when the preparatory algorithm is done, coincide.

3.3.2 Termination and Output guarantees

In terms of input to the algorithm, we want to clearly distinguish the three cases for Rule Set T (p. 50): The algorithm terminates provably; the algorithm does provably not terminate; and we did not show any termination statement. We discuss the guaranteed quality of result and investigate whether we were able to provide an upgrade to the original method by Oudot et al. (2005).

If $E \subset \partial O$ and ∂O smooth. Furthermore $\sigma \leq 0.09 d_M$ and the minimal angle in E not smaller than $\frac{\pi}{3}$. If α and B are chosen adequately (theorem 3.1.2.1) then the the application of Rule Set T (p. 50) **terminates** and the result is a good approximation of the domain O because of theorem 2.3.4.2. All elements have size at most σ. The resulting mesh is not necessarily a high quality mesh w.r.t. minimal dihedral angle but post-processing methods can be applied, e.g. Edelsbrunner et al. (2000) or Cheng et al. (2000).

If the minimal angle α_0 in E is smaller than $\frac{\pi}{3}$ and $B < \frac{1}{2 \sin \alpha_0}$ then the algorithm will **not terminate**.

Discussing Asmp1. In section 3.1 we assumed that the set of constrained segments is entirely contained in the domain boundary. This assumption is a strong limitation because only few boundaries exhibit regions where segments can be embedded, e.g. surfaces with planar regions or ruled surfaces.

Segments could also be prescribed in the interior of the domain O. We claim that as long as there is no intersection with the domain boundary ∂O, the Rule Set T (p. 50) can be applied as-is. Termination follows from a lower bound on the insertion radius, which can be derived for all segment configurations not exhibiting an angle smaller than $\frac{\pi}{3}$.

Segments, whose endpoints are in ∂O but that are not entirely contained in the domain boundary, are problematic w.r.t. the termination proofs shown, because the points inserted when splitting into subsegments are not part of the boundary. Hence they will eventually be vertices of a facet not having its three vertices on the boundary (compare rule R1, S2, T2) and cause point insertions that are likely to encroach on the subsegments again. To overcome this problem all points inserted on constrained segments are declared points of the boundary themselves, although this might not conform with the oracle. We claim that no fatal difficulties are introduced by this ambiguity as long as the constrained segment is close enough to the actual boundary in terms of desired discretisation fineness.

3.3.3 Discussion

Our goal was to provide an extension of Oudot et al. (2005) that enables the preservation of piecewise linear features. We came up with a strategy that integrates into the original method and proved that none of the constrained segments will be violated during the mesh generation process. Our strategy is guaranteed to terminate if no small angle is prescribed by the set of constrained segments. Our algorithm transfers the approximation results of Oudot et al. (2005) and obtains a good geometric accuracy for smooth surfaces.
We are lacking some technical result, especially if the domain boundary is not smooth: In the resulting mesh, no subsegment of the constrained segments will be encroached indeed but we did not prove that the subsegments are in the boundary of $\text{Del}_{|O_i}$, because the sizing field cannot be assumed to imply the ϵ-sampling of the domain boundary in that region (there is no lemma similar to lemma 2.3.5.1). As a consquence our algorithm

assures to maintain the prescribed features, provided the sizing criterion implies a sufficiently dense sampling, although we have no theoretical indicator for what sufficiently dense means.

We motivated the extension of the method of Oudot et al. (2005) by the use of the oracle, because it is particularly advantageous when meshes for objects that are not originally combined have to be generated. We highlighted that the oracle avoids computing the intersections explicitly in order to obtain a single consistent domain representation. Still we cannot preserve features not know explicitly with our extension.
It does apply for cases though, where objects have to be merged and features of the separate domains have to be preserved. In other words, if geometric features do not result from the fusion of the separate domains or if they can be estimated a-priori with less computational effort than an explicit boundary fusion, then our extension offers a method to generate adequate meshes of the overall domain.

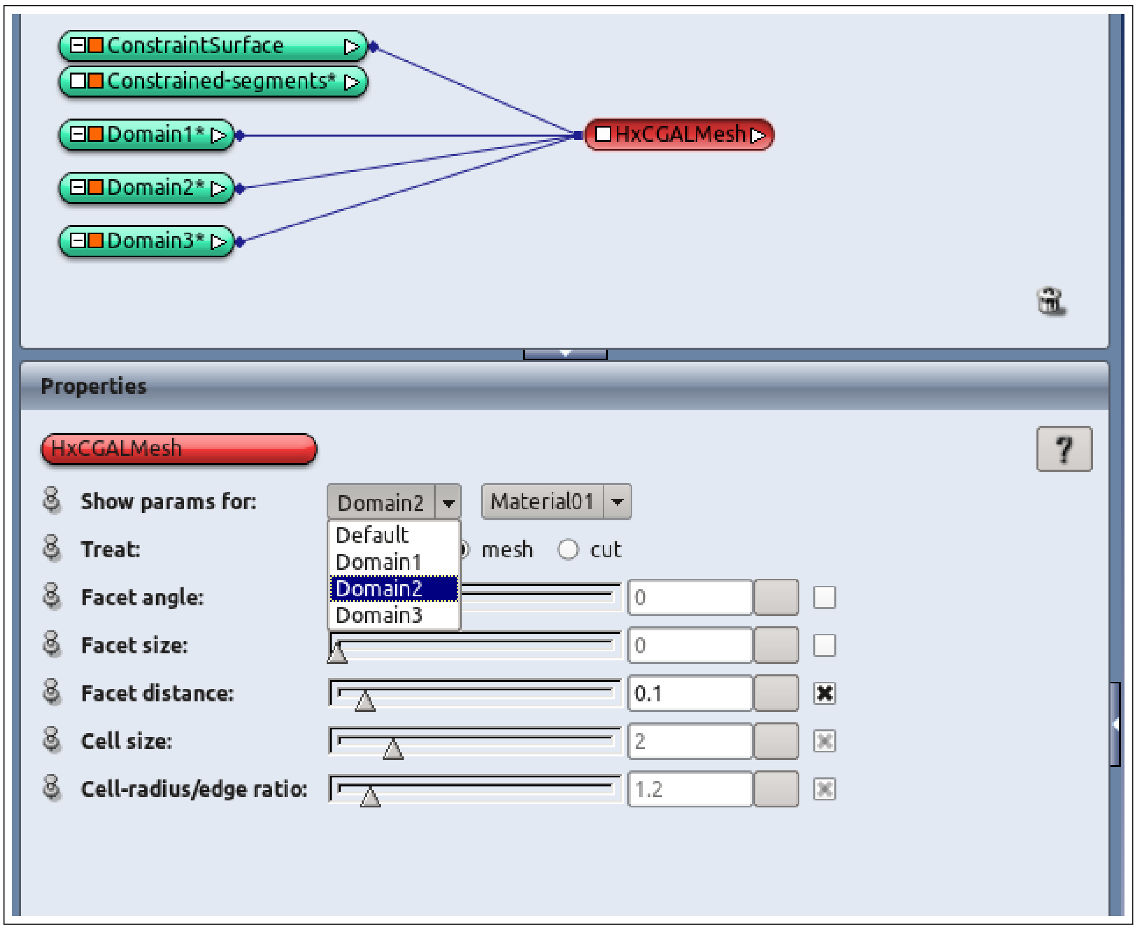
4 Implementation and Applications

4.1 Implementation

An algorithm corresponding to Oudot et al. (2005) along with the multi-material adaptation proposed by Pons et al. (2007) is available in CGAL since at least version 3.5. Our current implementation is based on CGAL 3.7 (see Alliez et al. (2010)). Some implementation details differ from the assumptions of the main approximation theorems (see theorem 2.3.4.1, theorem 2.3.4.2): The input domain boundary might not be smooth, e.g. if triangular surfaces or voxel data describe the domain. Further the initial point set is not necessarily guaranteed to meet the presets defined in Boissonnat and Oudot (2005). Lastly the sizing field is not checked to be Lipschitz continuous neither necessarily satisfying the medial axis distance property.

The CGAL library has been integrated into the framework of ZIBAmira (see Stalling et al. (2005)), currently version 2012.03. ZIBAmira offers data structures to handle various different types of data and provides tools to visualize them adequatly.

A hierarchical oracle. An oracle has been implemented that defines the domain to be meshed. It allows to implicitly perform Boolean operations, because the mesh generation algorithm itself does only require two kinds of query operations: (1) determination of the material (domain index) which a given point belongs to and (2) computation of a point where a given segment, ray or line intersects a domain boundary. Being able to perform computations for (1), (2) can always be implemented by iterative bisection as a fallback technique. The hierarchical oracle essentially is the maintenance of a prioritized list of input domains, such that point queries are passed to lower priority domains if they are outside all higher prioritized inputs. That way the oracle is a blackbox that performs the Boolean operations on the input domains without explicitly computing the intersections and not caring for inconsistencies in data type and shape. The data



HxCGALMesh

Show params for: Domain2 | Material01

Default
Domain1
Domain2
Domain3

Treat: mesh ○ cut

Facet angle: 0 ☐

Facet size: 0 ☐

Facet distance: 0.1 ☒

Cell size: 2 ☒

Cell-radius/edge ratio: 1.2 ☒

Figure 4.1: Using the mesh generation algorithm in ZIBAmira. ZIBAmira object pool - upper part: The separate subdomains are connected sequentially to the mesh generation object in the ZIBAmira object pool. Additionally a *constraint surface* with associated surface path set containing the set of constrained segments is connected to the mesh generation module; Mesh generation module ports - lower part: Instead of a sizing field covering the whole domain space, the quality criteria can be bounded by constants. They can be defined separately for each material. Also it is possible to treat a material of a domain as *cut*, i.e. it will be treated such that all points falling within this material are handled as points outside of any domain with same or lower priority.

types that can currently be handled are (1) watertight triangular surface meshes with exactly two triangles joining at an edge, (2) labeled uniform voxel grids, and (3) implicit spheres, i.e. given by midpoint and radius.
The hierarchical oracle is what users of the software consider the most useful feature.

The features to be preserved are defined by attaching a vertex path set, i.e. a set of linearly connected nodes, to the mesh generator. In ZIBAmira suitable path sets can be defined on triangulations using the *Dijkstra* connector or, if beginning with an arbitrary surface path set, by retriangulating the surface according to the paths.

If specific points of the domain boundary are known a-priori these can be added to the initial point set in order to speed up the mesh generation process or simply to preserve the point set. The module offers a port to connect a geometry containing these points.

An additional option allows to change the way the materials are assigned to the tetrahedra. While the approach by Pons et al. (2007) assigns the materials according to the location of the circumcenters, in our implementation it is also possible to use the barycenter. This is an option in cases where the sizing field cannot guarantee that circumcenters of tetrahedra are mapped to the *right* domain by a sufficiently dense surface sampling.

We started to implemented a strategy to additionally handle planar input polygons (also non convex) and the according software layer has been added between the segment preservation and the facet layer. For each polygon G, let H by the hyperplane which contains the polygon and let $P_{|G}$ be the set of points in P that are contained in G. Then by assumption all bounding segments of the polygon are present in the conforming sense due to the higher priority preservation of the segments. Hence the 2-dimensional affine Delaunay triangulation $\mathrm{Del}_{|H,2}(P_{|G})$ of $P_{|G}$ within H yields a triangular subdivision of G. Comparing the connectivity in $\mathrm{Del}_{|H,2}(P_{|G})$ to the connectivity in $\mathrm{Del}(P)$ and determining the missing triangles, provides a good guess where to insert further points (namely at their circumcenters) in order to recover the faces in 3d space. The implementation of this strategy has not been completed. Proofs on achievable quality or termination of the corresponding algorithm have not been investigated.

4.2 Application example

We applied our implementation to an example from orthopedic surgery. In Galloway et al. (2010) finite element meshes of implanted tibiae were generated to compute strains at the bone-implant interface. The setup is described in fig. 4.2. The meshes in this previous study were generated using an advancing-front approach. A method dedicated to generate a single consistent surface representation of the bone-implant compound has been developed in order to have a valid input for the mesh generation process.

We randomly selected one hundred virtual total knee replacement settings from that study, including geometric representations of the tibia and the tibial component. For

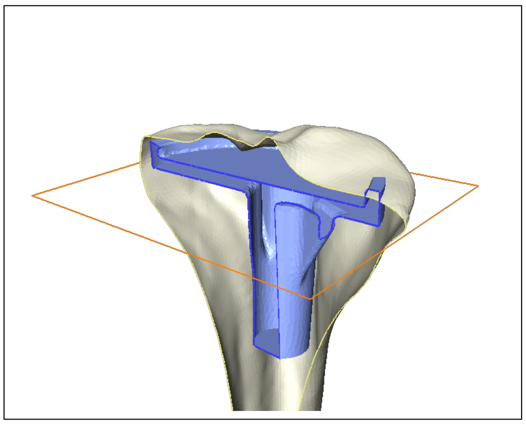


Figure 4.2: Mesh setup for tibial implant integration. Cross section of aligned bone and implant geometry. The enclosed volumes are inconsistent because the boundaries mutually penetrate. Additionally the orange rectangle specificies where protruding bone should be removed. The implant is prior to the hierarchical oracle. The cutting region was set before the bone in order to remove the superfluous parts.

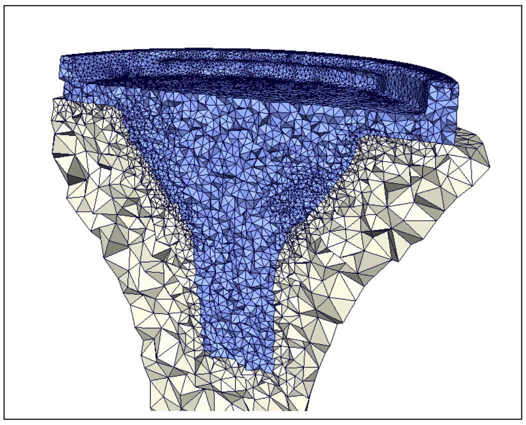


Figure 4.3: Resulting mesh. High geometric accuracy is obtained at the implant boundary, notably at the sharp features. Mesh size grades down where elements are far from the implant-bone material interace. Snapshots created using ZIBAmira 2012.03.

each implant the region where protruding bone is removed was given and its sharp edges were marked as line segments to be preserved. The one hundred datasets were then meshed with our implementation corresponding to Rule Set T (p. 50) and with the mesh generation method available in CGAL 3.7, corresponding to the method of Pons et al. (2007) and referred to as *the featureless approach*. In both setups the desired quality criteria were chosen as follows: (1) a maximal radius-edge ratio of 1.1 for all materials, (2) a maximal circumradius of 1 mm for the implant and 3 mm for the bone and (3) a maximal facet distance of 0.1 mm to approximate the surfaces for the implant and 2 mm for the bone material. The results of both approaches were post-processed by the sliver removal methods available in CGAL 3.7.

The tetrahedral meshes generated with our implementation met the quality criteria with 437,000 tetrahedra on average (range 295,000 to 707,992). The number of tetrahedra generated with the featureless approach was significantly larger with an average number of 900,898 tetrahedra per mesh (range 614,234 to 1,533,011). For the meshes generated by our method, the average minimal dihedral angle was 8.72°(±1.31°). The models that

where generated using an advancing front approach obtained an average minimal dihedral angle of 4.09°(±3.18°).
We attribute both results to the geometric configuration of bone and implant, see fig. 4.4.

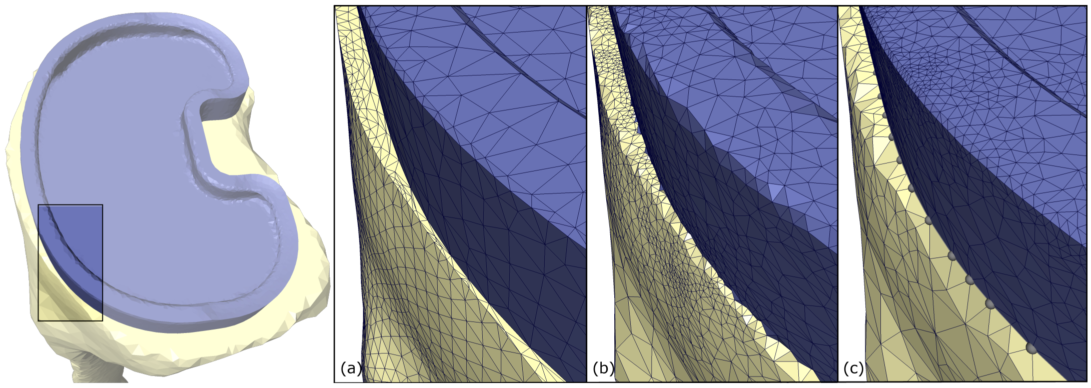


Figure 4.4: (a) The advancing-front approach preserves the surface triangulation, which enclose the volumes to fill with tetrahedra. It generates distorted elements in regions where implant and bone surface leave small gaps. (b) The featureless approach inserts many elements in the vicinity of these gaps to accurately recover the geometry of the implant. With the radius-edge ratio specified, that even leads to smaller elements on the bone surface. (c) These problems do not occur with the *guided* point insertion in our method.

4.3 Conclusion

We have extended the 3d mesh generation algorithm that is present in CGAL 3.7 to preserve prescribed segments. The implementation has been performed within the environment of ZIBAmira in order to benefit from its existing tools. A hierarchical oracle, which allows to implicitly perform some Boolean operations on the input data, has been implemented to allow for easy definition of the desired domain without time-consuming preprocessing. Where the set of constrained segments was benign w.r.t. angle of incident segments, practical experiments performed way better than the results of our termination proofs were able to guarantee.

5 Discussion and Conclusion

5.1 Conclusion

We investigated tetrahedral mesh generation in the context of generating discretisations of three-dimensional domains to perform finite element computations. Such a setting poses conflicting demands on a mesh, whose handling is addressed by several mesh generation methods. Among them, Delaunay refinement strategies promise provable results while simultaneously being applicable to various setups. In settings where a mesh has to be derived for objects that are not originally combined, the creation of an initial consistent boundary representation is an essential step in the mesh generation process. Such a boundary representation can be deficient or impose geometric constraints that affect the mesh generation process although they are not relevant for the finite element computation. The approach employing the oracle abstracts the domain and avoids these problems. However, geometric features cannot be preserved explicitly and a good approximation tends to introduce numerous superfluous elements in their vicinity.

We proposed a method to extend the oracle-based approach by Oudot et al. (2005) and proved that our algorithm terminates while preserving any set of segments not exhibiting small angles. Our implementation provides a hierarchical oracle that allows for an intuitive setup if simple Boolean operations describe the mutual relations of separate input domains. An example, based on a recent study where finite element meshes were required, highlights that our method simplifies mesh generation for specific setups while being able to provide good quality results.

5.2 Discussion

The concept of keeping the domain abstract to the mesh generation method offers great advantages in practice because time consuming and error-prone precomputations can be avoided. It is inherent to the oracle approach that the domain is not known explicitly by the mesh generation method. But if point insertion has to be guided by geometric features of the domain further knowledge has to be incorporated.

Boltcheva et al. (2009) observed this problem when generating tetrahedral grids for multi-labeled image data. They propose a method that extracts multi-material junctions from the input data, samples them according to a user-given parameter and preserves the edges connecting the sampled points throughout the mesh generation process. The mesh generator terminates but fails to obtain provable (or w.r.t. the constrained junctions optimal) quality.

Our approach assumes that feature segments are given as an input to the algorithm but we are not restricting to a specific input data representation. The constrained segments will ultimately be preserved in the Delaunay triangulation of the underlying mesh generator. But the method does not guarantee to terminate for all possible configurations. Further no proof is given that the segments actually represent the feature in the resulting mesh if the input is not smoothly bounded. The quality bounds that have been provided are not useful in practical application.

Preservation of segments is currently achieved by not allowing vertices inside their diametral balls. This criterion is not one-to-one. Diametral lenses provide a smaller region of encroachment. A one-to-one criterion would actually evaluate the encroachment of all tetrahedra that are incident to the respective edge. Also there is no stringency to split segments at their midpoint. A sophisticated splitting strategy, e.g. splitting at virtually or actually intersecting Voronoi facets could be a subject of further investigation (note that such points are not generally available).
W.r.t. feature preservation the restriction to segments is inconsequent because it lacks the generality of the oracle approach. The problem of generating Delaunay triangulations for PSCs incorporates the preservation of smooth curve segments and a strategy has been proposed in Cheng et al. (2010). To our knowledge this strategy has not been explicitly applied to the oracle approach by Oudot et al. (2005).

Future work. Further investigation in our method should refine the proof of termination and improve the bounds in order to provide useful shape quality guarantees too. This equally concernces the results of Oudot et al. (2005) as their radius-edge ratio bound is never below 4. A sizing field implying the approximation guarantees through the dense sampling of the domain boundary is currently required in the oracle approach. If possible at all, a reliable strategy to obtain a suffiently dense sampling without explicitly computing a sizing criterion would be of great use in practice.

We plan to implement a strategy like the *Terminator* proposed in Shewchuk (1998) in order to provide termination guarantee for all configurations of constrained segments. Furthermore heuristic approaches to remove slivers can be easily integrated in the criteria-driven refinement method and are desirable in pratical application (see Shewchuk (1997)) - certainly a termination criterion for such a strategy should be provided. It is planned to implement the possibility to incorporate a lower bounding sizing field.

Bibliography

Alliez, P., Cohen-Steiner, D., Yvinec, M., and Desbrun, M. (2005). Variational tetrahedral meshing. *ACM Transactions on Graphics (TOG)*, 24(3):617–625.

Alliez, P., Rineau, L., Tayeb, S., Tournois, J., and Yvinec, M. (2010). 3D mesh generation. In *CGAL User and Reference Manual*. CGAL Editorial Board, 3.7 edition. http //www.cgal.org/Manual/3.7/doc_html/cgal_manual/packages.html#Pkg Mesh_3.

Boissonnat, J. and Oudot, S. (2005). Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451.

Boltcheva, D., Yvinec, M., and Boissonnat, J. (2009). Mesh generation from 3d multi-material images. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009*, pages 283–290.

Bowyer, A. (1981). Computing dirichlet tessellations. *The Computer Journal*, 24(2):162–166.

Cheng, S., Dey, T., Edelsbrunner, H., Facello, M., and Teng, S. (2000). Silver exudation. *Journal of the ACM (JACM)*, 47(5):883–904.

Cheng, S., Dey, T., and Ramos, E. (2010). Delaunay refinement for piecewise smooth complexes. *Discrete & Computational Geometry*, 43(1):121–166.

Cheng, S., Dey, T., Ramos, E., and Ray, T. (2004). Quality meshing for polyhedra with small angles. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 290–299. ACM.

Chew, L. (1989). Guaranteed-quality triangular meshes. Technical report, DTIC Document.

Dey, T. (2007). *Curve and surface reconstruction: algorithms with mathematical analysis*, volume 23. Cambridge Univ Pr.

Dey, T. and Levine, J. (2009). Delaunay meshing of piecewise smooth complexes without expensive predicates. *Algorithms*, 2(4):1327–1349.

Edelsbrunner, H. and Harer, J. (2010). *Computational topology: an introduction*. Amer Mathematical Society.

Edelsbrunner, H., Li, X., Miller, G., Stathopoulos, A., Talmor, D., Teng, S., Üngör, A., and Walkington, N. (2000). Smoothing and cleaning up slivers. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 273–277. ACM.

Galloway, F., Seim, H., Kahnt, M., Nair, P., Worsley, P., and Taylor, M. (2010). A large scale finite element study of an osseointegrated cementless tibial tray.

Hege, H., Stalling, D., Seebass, M., and Zockler, M. (1997). A generalized marching cubes algorithm based on non-binary classifications.

Löhner, R. and Parikh, P. (1988). Generation of three-dimensional unstructured grids by the advancing-front method. *International Journal for Numerical Methods in Fluids*, 8(10):1135–1149.

Miller, G., Talmor, D., Teng, S., Walkington, N., and Wang, H. (1996). Control volume meshes using sphere packing: Generation, refinement and coarsening. In *Fifth International Meshing Roundtable*, pages 47–61.

Oudot, S., Rineau, L., and Yvinec, M. (2005). Meshing volumes bounded by smooth surfaces. In *Proceedings of the 14th International Meshing Roundtable*, pages 203–219. Springer.

Owen, S. (1998). A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, volume 3. Citeseer.

Pons, J., Ségonne, F., Boissonnat, J., Rineau, L., Yvinec, M., and Keriven, R. (2007). High-quality consistent meshing of multi-label datasets. In *Information Processing in Medical Imaging*, pages 198–210. Springer.

Radovitzky, R. and Ortiz, M. (2000). Tetrahedral mesh generation based on node insertion in crystal lattice arrangements and advancing-front-delaunay triangulation. *Computer Methods in Applied Mechanics and Engineering*, 187(3):543–569.

Rineau, L. and Yvinec, M. (2007). A generic software design for delaunay refinement meshing. *Computational Geometry*, 38(1-2):100–110.

Rineau, L. and Yvinec, M. (2008). Meshing 3d domains bounded by piecewise smooth surfaces. In *Proceedings of the 16th International Meshing Roundtable*, pages 443–460. Springer.

Ruppert, J. (1995). A delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585.

Schöberl, J. (1997). Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and visualization in science*, 1(1):41–52.

Shewchuk, J. (1997). Delaunay refinement mesh generation. Technical report, DTIC Document.

Shewchuk, J. (1998). Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 86–95. ACM.

Shewchuk, J. (2002). What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). *University of California at Berkeley*.

Si, H. (2006). A quality tetrahedral mesh generator and three-dimensional delaunay triangulator. *Weierstrass Institute for Applied Analysis and Stochastic, Berlin, Germany*.

Si, H. and Gärtner, K. (2005). Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th international meshing roundtable*, pages 147–163. Springer.

Stalling, D., Westerhoff, M., and Hege, H.-C. (2005). Amira: A highly interactive system for visual data analysis. In Hansen, C. D. and Johnson, C. R., editors, *The Visualization Handbook*, pages 749 – 767. Elsevier.

Watson, D. (1981). Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The computer journal*, 24(2):167–172.

Yang, Y., Yong, J., and Sun, J. (2005). An algorithm for tetrahedral mesh generation based on conforming constrained delaunay tetrahedralization. *Computers & Graphics*, 29(4):606–615.