Computer-Aided Design and Applications

http://www.cadanda.com

Variational Tetrahedral Meshing of Mechanical Models for Finite Element Analysis

Matthijs Sypkens Smit[1] and Willem F. Bronsvoort[2]

[1]Delft University of Technology, M.SypkensSmit@tudelft.nl
[2]Delft University of Technology, W.F.Bronsvoort@tudelft.nl

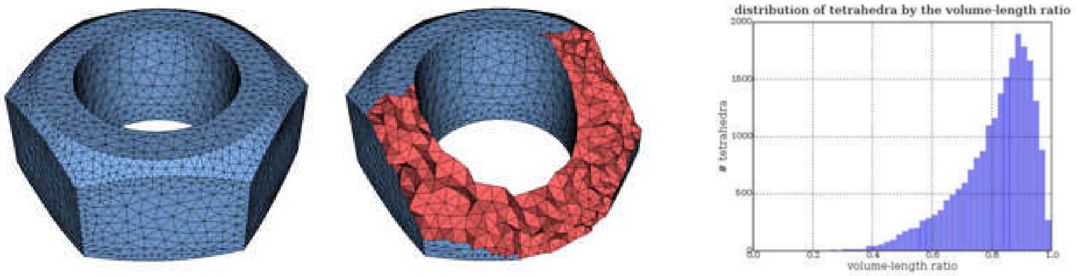
ABSTRACT

Tetrahedral meshing algorithms balance meshing speed, expected mesh quality, strength of quality guarantees, parameter control of the user, and feasibility of implementation.
Variational tetrahedral meshing focuses on creating meshes with highly regular tetrahedrons. The method has shown impressive results. However, most of the examples are smooth models, and comparatively little has been remarked about meshes of mechanical models for finite element analysis. The quality of the elements of such meshes is very important, but a good representation of the boundary is also essential for such analysis. Mechanical models often have boundaries with many distinctive features such as edges and corners that need to be well represented by the mesh.
We have explored the application of variational tetrahedral meshing to these kind of models and have developed enhancements geared towards obtaining a representation of the boundary that is suitable for analysis. These enhancements are described, and several meshes generated by applying them are presented.

Keywords: variational tetrahedral meshing, FEA, boundary conformance, Delaunay meshing.
DOI: 10.3722/cadaps.2008.228-240

1. INTRODUCTION

Meshing is still an active area of research with many applications and a large base of algorithms, each devoted to a specific type of mesh or problem [13]. Our focus is on tetrahedral meshing of mechanical models for finite element analysis (FEA).
There are many considerations that can be taken into account when creating a tetrahedral mesh of a model. No single algorithm honours all of them. If one asks for guarantees on the quality, then often there is little control over the number of vertices. If the highest quality is sought for, then the construction is going to be relatively time consuming.
When meshing for FEA, the quality of the mesh is very important. A single bad element in a mesh can degrade the quality of the analysis [12]. Quality is foremost determined by the size and the angles of the elements. More specifically, in tetrahedral meshes, such as in Fig. 1(a), large dihedral angles (close to 180°) cause problems with the interpolation, whereas small dihedral angles (close to 0°) cause bad conditioning of the stiffness matrix. From this perspective, the elements in a quality tetrahedral mesh should all have a shape similar to a regular tetrahedron. The sizes of the elements are also of influence. Smaller elements lower the discretisation error, but large differences in size between the smallest and the largest element are again bad for matrix conditioning [12].
Analysis as part of product design is mainly driven by practical considerations: the engineer wants to know whether the product functions to specification and by what margins. The quality of a mesh is optimal in the context of a particular analysis if the results fall within the required error bounds, for minimal computational effort. Many methods improve quality through (Delaunay) refinement, but slivers often remain [9]. The number of elements after refinement can come out higher than necessary. Some methods to optimize the quality keep approximately the same number of elements by changing node locations and the connectivity, but at higher computational cost.



(a) Tetrahedral mesh of a nut model

(b) Distribution of volume-length ratio of elements in the mesh in (a)

Fig. 1: Mesh and quality of a nut model.

Variational tetrahedral meshing [3] achieves a quality mesh with a fixed number of nodes. The demonstrated results for smooth models are impressive, but little has been remarked on its applicability to meshing of mechanical models for FEA. Meshes for analysis, in particular, need to represent the boundary accurately, including all features such as edges and corners. While acknowledging that their method approximates the boundary, the authors of [3] do mention the possibility of meshing models with sharp features and show images of two examples, but a real discussion is lacking.

We have investigated the application of variational tetrahedral meshing to mechanical models for FEA. In particular, we have looked at the representation of the model boundary by the mesh. We have developed enhancements geared towards obtaining an accurate representation of the boundary, and demonstrated that with these adaptations this approach is feasible for generating high quality meshes of mechanical models that are suitable for analysis. We also offer some practical considerations for applying the method.

We start with a review of the original variational tetrahedral meshing algorithm in Section 2, followed by a discussion of the aspects where the method is lacking with respect to the meshing of mechanical models in Section 3. In Section 4 and Section 5 we propose enhancements to the algorithm, followed by some practical considerations for applying the method in Section 6. Finally, we show several meshes generated by this approach in Section 7, and conclude the paper in Section 8.

Note on terminology: when we speak of a (Delaunay) triangulation in 3D space, this is the same as the (Delaunay) tetrahedralisation of the points. To measure the quality of a tetrahedron, we consistently use the volume-length ratio (see Fig. 1(b)) instead of the radius ratio that is used in [3]. The visual difference between the histograms of the quality of all elements by the two measures is minimal. Within the context of analysis there seems to be a slight preference for using the volume-length ratio [8].

2. VARIATIONAL TETRAHEDRAL MESHING

The quality of systematically constructed meshes in 3D is generally far from optimal. Since long, people have been improving the quality after the initial construction. Laplacian smoothing has been a popular method since the early days, moving nodes to relax the mesh for the given connectivity. However, this simple procedure can improve tetrahedral meshes only to a certain degree, leaving many elements of bad quality in existence. A significant step forward for the optimization of tetrahedral meshes has been the focus on centroidal Voronoi triangulations (CVTs) [7], where nodes are moved in order to approach the centroid of their Voronoi cell. Later Chen and Xu in [6] and [5] introduced the optimal Delaunay triangulation (ODT), further improving the quality of the primary mesh elements.

From these ideas, variational tetrahedral meshing (VTM) was conceived, a method that does not optimize mesh quality as an after-thought, but integrates the optimization in the construction procedure. The pivotal idea of VTM is to optimize the quality of the elements and to work on boundary conformance in a single iterative process. Commonly, first the boundary is meshed and then the interior is meshed, conforming to the boundary. In VTM the mesh on the boundary evolves during the optimization process. The connectivity of the mesh is governed by the Delaunay criterium and thus, unless the model is convex, the final mesh has to be extracted from the Delaunay mesh of the nodes. Also standing out, is the use of an indicative number of nodes to construct the mesh. When performing an analysis, the number of elements or nodes that can be handled is often roughly known. In those cases, a method that can generate a quality mesh with the specified maximum number of nodes, can offer an advantage.

We summarize the working of the algorithm. For a more detailed treatment, we refer to the original presentation of the algorithm [3]. The algorithm consists roughly of four steps:

1. initialise data structures, sizing field
2. distribute nodes
3. iteratively optimise nodes
4. extract mesh

2.1 Initialise Data Structures, Sizing Field

An efficient point location test is needed, both for the distribution of nodes and for the extraction of the final mesh from the resulting Delaunay mesh that covers the convex hull of the nodes. For the latter we need to decide for tetrahedrons whether they fall inside or outside the model boundary. A constrained or conforming Delaunay mesh of the original model is used for this. We call this the *control mesh*. The control mesh accurately represents the boundary, which is important for the overall accuracy of the procedure. Also, for the generation of a graded mesh, a mesh sizing field μ needs to be constructed. For this we use an approximation of the local feature size (*lfs*) constructed with a procedure based on the work in [4]. The lfs captures the shortest distance of a point to the nearest geometric feature of the model that the point itself is not a part of. Other measures, mostly similar in spirit, could also be used, such as [11] or [10]. Variational tetrahedral meshing uses the mesh sizing field as a relative measure, since it works with a fixed number of nodes.

2.2 Distribute Nodes

The requested number of nodes is spread out in accordance with the sizing field. This is done by iterating the cells of a grid that covers the bounding box of the model. In a first iteration, a sum weighted by the sizing requirements in each cell is calculated. Based on this sum, in a second iteration, a fair proportion of the nodes are placed randomly inside the grid cells that have the center of the cell inside the model. The cells are traversed in serpentine order, spilling over any non-integer number of nodes that are called for into the next cell. For example, if 0.2 nodes are called for in a particular cell, then no node is placed there, but the 0.2 is added to the number of nodes that, in accordance with the sizing field, are called for in the next cell. After this process we end up with a cloud of nodes that covers approximately the volume of the original model, with the node density varying relative to the sizing field.

2.3 Iteratively Optimize Nodes

During the optimization process, the nodes fall in two categories: boundary nodes and interior nodes. Every node starts as an interior node, but can become a boundary node when it is selected as such during the determination and repositioning of the boundary nodes. Each iteration of the optimization loop starts with the identification and positioning of the boundary nodes. After that, the rest of the nodes, deemed part of the interior set, is optimized.

So at the start of each iteration it is determined which nodes are part of the boundary. These boundary nodes are then (re)positioned in accordance with the sizing field, aiming for balanced node spacing on the boundary. This is achieved by employing a fine set of so-called quadrature samples. These samples are points lying as a fine-mazed net over the boundary of the control mesh, and collectively represent the boundary of the model. Normally, when meshing a CAD model, the samples will have a connectivity. They are essentially the nodes of a fine-sampled surface mesh.

For each quadrature sample, at position $\mathbf{x}$, we locate its nearest node and have the sample exert a virtual pull on that node proportional to the area that the sample covers (ds) and consistent with the sizing field: $ds/\mu^4(\mathbf{x})$. This is the weight of the pull or quadrature value of the sample. The nodes that have at least one sample pulling on them, are now considered part of the boundary set. They are moved to the average location of the pulling samples, weighted by the quadrature values. The rest of the nodes belongs to the interior.

To ensure that nodes end up at the edges where a clear separation between faces exists and at the corners, we treat their samples differently from the surface samples. The quadrature value for samples on edges is computed as $dl/\mu^3(\mathbf{x})$, with dl the length that the sample covers, and to corner samples an infinite quadrature value is assigned, to ensure the assignment of nodes there. The procedure starts with the regular surface samples pulling in and repositioning nodes, then the edges and finally the corners are taken care of.

After dealing with the boundary, the Delaunay mesh is reconstructed as nodes have moved. Then each interior node, i.e. each node x_i that was not selected as a boundary node, is moved according to the formula:

$$\mathbf{x}_i^{\text{new}} = \frac{1}{s_i}\sum_{T_j \in \Omega_i} \frac{|T_j|}{\mu^3(\mathbf{g}_j)}\mathbf{c}_j \text{ , with } s_i = \sum_{T_k \in \Omega_i} \frac{|T_k|}{\mu^3(\mathbf{g}_k)} . \qquad (1)$$

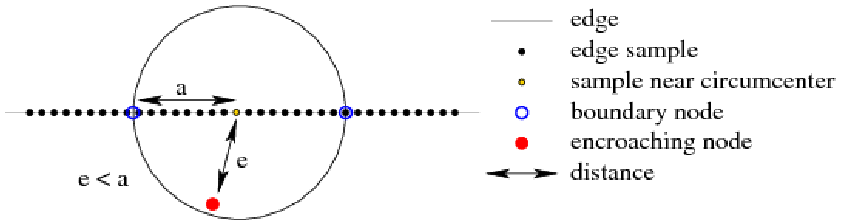


Fig. 2: Encroaching of part of edge.

Here Ω_i denotes the one-ring of tetrahedrons that share node x_i, $|T_j|$ denotes the volume of tetrahedron T_j, $\mathbf{c}_j$ is the circumcenter of tetrahedron T_j, and $\mathbf{g}_j$ is the centroid of tetrahedron T_j. The effect of this relocation is similar in idea to the relocation of a node towards the center of its Voronoi cell. Instead of optimizing the compactness of the Voronoi cell, this operation aims at improving the compactness of the tetrahedrons in the one-ring around the node. For a more detailed motivation we refer to [3]. If the new location would invalidate the Delaunay property of the mesh, then the connectivity is changed to keep it a Delaunay mesh.
The optimization loop alternates between these two phases of 1) determining and repositioning the boundary nodes, and 2) optimizing the location of the interior nodes.

2.4 Extract Mesh

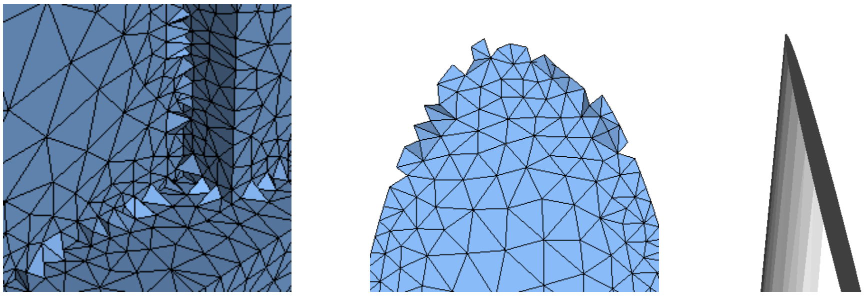
After a sufficient number of optimization steps, the mesh representing the model has to be extracted from the resulting Delaunay triangulation, which covers the convex hull of the nodes. As the model is usually not convex, it needs to be decided which tetrahedrons contribute to the model, i.e. are inside, and which tetrahedrons fall outside the model. This process is called *peeling*, as it can be envisioned as the removal of tets that are outside of the mesh that represents the real model boundary. It is worthwhile to ask why this is possible in the first place. Normally, the Delaunay triangulation of a node set of a model does not contain the complete boundary; some faces and edges have to be recovered. Why can we expect the model boundary to be present in the triangulation after the optimization procedure in VTM?
There are no theoretical guarantees that the boundary will be present, but with the quadrature samples being finer, the chance of success increases; see Fig. 2. If a node encroaches upon the minimal circumsphere of two nodes that currently represent part of a boundary edge, the node is likely to be drawn to the boundary, since a sample near the center of the edge between the two nodes will most likely have the encroaching node as its nearest node. Similarly for the surface area of the boundary, a node that encroaches upon the minimal circumsphere of a triangle that needs to represent part of the boundary is likely to be drawn to the boundary. This effectively results in a mesh for which all boundary elements have minimal empty circumspheres. With a higher number of boundary samples, the likelihood that this ad hoc protection procedure works, increases. An edge or face with its minimal circumsphere empty of other nodes is called *Gabriel* and is guaranteed to be part of the Delaunay triangulation. If all edges and faces on the boundary are Gabriel, then the boundary is present in the Delaunay triangulation. This Gabriel restriction is stronger than required. There are cases though where, even with ample quadrature samples, the boundary might not be present. We will come back to this.
The procedure for mesh extraction that is followed in [3] is quite simple: a tetrahedron T_i is definitely part of the model if its circumcenter falls inside the control mesh. If the circumcenter falls outside, tetrahedron T_i is still considered to represent part of the model if

$$d(\mathbf{c}_i, \partial\Omega)/r_i < 0.4\,.$$

Here r_i denotes the circumradius of tetrahedron T_i and $d(\mathbf{c}_i, \partial\Omega)$ the distance of the center of the circumsphere to the boundary. This causes tetrahedrons that have their circumcenter only just outside the control mesh, relative to the size of the circumradius, to be considered as part of the model as well.

3. ENHANCHING VTM FOR MESHING MECHANICAL MODELS

The VTM algorithm has deficiencies when meshing for FEA. In particular, the representation of the boundary and the extraction of the right mesh are an issue. The algorithm only aims at approximating the boundary, but for mechanical models it is essential that the boundary is accurately represented by the mesh. This means that all essential features, such as edges and corners, should be present with the right shape and size. VTM often fails at this point.

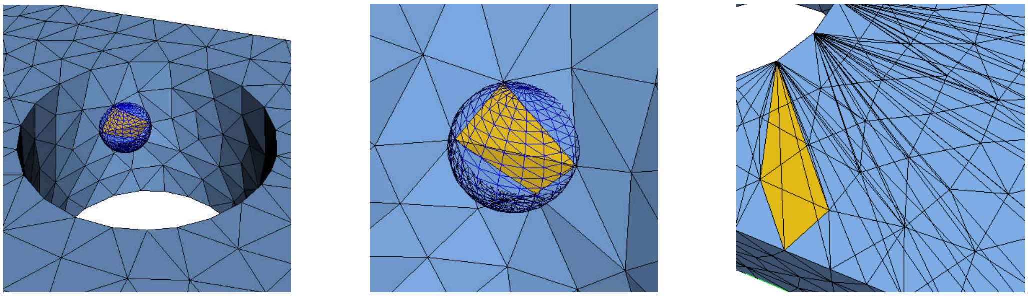


(a) Excess elements along edges in concave areas

(b) Missing elements near small dihedral angle

(c) Object in (b) from the side

Fig. 3: Failing mesh extraction.



(a) Sliver in concave area

(b) Zoom-in of (a)

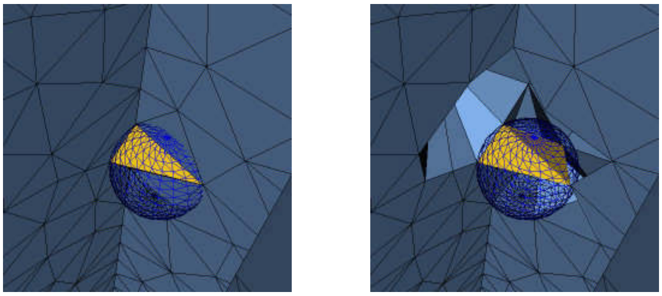
(c) Slivers on top of a nearly flat face

Fig. 4: Slivers on the boundary.

Even assuming that the model boundary is accurately represented in the mesh, the procedure for mesh extraction proposed in [3] does not consistently determine the correct boundary. Elements that should be considered outside are concluded to be inside and vice versa. This is in particular clear near edges, where a failure to extract the right mesh causes (part of) an edge to be missing or hidden behind excess elements. See Fig. 3(a), which shows excess elements near concave edges. These elements can actually be removed (they do not lie partly inside the model), but they are considered part of the model nonetheless. Fig. 3(b) shows an example where the same procedure has removed elements that are necessary to accurately represent the model boundary. An acceptable boundary was present in that case. Fig. 3(c) shows the model corresponding to this mesh from the side. The dark face on the right is the face in view in Fig. 3(b).

Also, there can be slivers (elements with a reasonable node spacing, but nearly flat as a whole) left flat on the boundary. This happens in particular in concave areas, as can be seen in Fig. 4(a) and its zoom-in in Fig. 4(b). The blue sphere is the circumsphere of the selected tetrahedron. All four corner nodes are in view. The only edge that is not visible runs at the back between the bottom and top node. In addition, when working with a robust Delaunay implementation, slivers can easily arise in flat faces if not all vertices lie exactly in the same plane. This is rarely a problem when following the procedure described in Section 2.4; however, when using alternative criteria in the procedure for mesh extraction, such as the ones we propose further on, results such as in Fig. 4(c) can arise. Here a handful of very flat tets lie on top of each other and on top of quality tets. We should be careful to avoid including these tets in the extracted mesh.

Another problem is that an acceptable representation of the boundary of the model is not always present in the Delaunay mesh that results from the optimization step. One can end up with elements that cross the boundary, such as illustrated in Fig. 5. The left image shows the boundary of the extracted mesh; in the right image some adjacent tets have been removed from the view. The selected tetrahedron has one vertex in the interior of the mesh and three on the boundary, but not all on the same boundary face. One of its faces is outside the boundary, one is inside, and the remaining two cross the boundary. This was not a problem for the original VTM algorithm, since is was primarily intended for meshing models for which an approximate boundary is acceptable. However, a better boundary representation is required for the analysis of mechanical models.



(a) Selected element crossing the boundary

(b) Selected element free from surrounding elements

Fig. 5: Elements crossing the boundary.

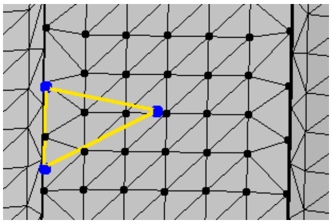


Fig. 6: Quadrature samples on the boundary of region in Fig. 5.

The problems mentioned can for a large part be eliminated. We will first discuss the presence of an accurate representation of the boundary in the mesh and describe some enhancements to the algorithm that aim to enforce such a representation. Thereafter, in Section 5, we will discuss an alternative approach to mesh extraction that performs better at extracting the intended mesh.

4. CONSTRUCTING THE BOUNDARY

As indicated in the previous section, we should take precautions to prevent ending up after the optimization step with a triangulation from which the boundary cannot be recovered. This problem can mostly be prevented by making a good choice upfront for the number of boundary samples and the number of nodes. We discuss these two aspects in Section 4.1 and Section 4.2. During the optimization step we pay close attention to signs that indicate a potential problem. In such a case we can apply node splitting, which we discuss in Section 4.3. Finally, after the optimization loop, we make sure that the mesh is suitably prepared for the extraction procedure, which we describe in Section 4.4.

4.1 Number of Boundary Samples

We discussed in Section 2.3 the procedure that aims to protect the boundary by pulling a node towards the boundary if it encroaches upon the minimal circumsphere of a boundary element. With sufficient boundary samples, practically every edge and boundary face will consist of elements with an empty minimal circumsphere. The problem in Fig. 5 is indeed caused by a lack of quadrature samples. Fig. 6 shows a mesh of the quadrature samples for the model; the outline of the boundary face that should have been present to avoid the problem is included. For the selected tetrahedron in Fig. 5(a), the corner node in the interior of the mesh is inside the minimal circumsphere of the missing face. If there would have been a quadrature sample close enough to the centroid of the missing face, then the interior node would have been pulled to the boundary, effectively emptying the minimal circumsphere of the missing triangle and increasing the density of the nodes on the boundary relative to those in the interior. An accurate representation of the boundary through a triangulation of the boundary nodes, has smaller minimal circumspheres, which are less likely to be encroached by interior nodes.

In our experience, applying this protection procedure with roughly 10 quadrature samples locally available around each node on the boundary, makes the occurrence of an element crossing the boundary extremely rare. We remind the reader that an empty minimal circumsphere of a boundary face is a stronger restriction than necessary for it to appear in the Delaunay triangulation. Thus even if an encroaching node fails to be pulled to the boundary to guarantee protection, the boundary can still be present in the Delaunay triangulation. This is even likely if the number of samples is high. However, some models are just inherently difficult to mesh with a conforming Delaunay triangulation, mostly due to small dihedral angles. The risk of boundary encroachment near such an angle cannot be completely eliminated by our heuristics. Failure to correctly represent the boundary should be detected during or after the mesh extraction.

4.2 Number of Nodes

The boundary protection procedure works by locally making the density of the nodes on the boundary slightly higher than on the interior. This procedure can only work if there are enough nodes in the area. The other factor of importance for an accurate representation of the boundary is thus the number of nodes relative to the complexity of the geometry. A certain number of nodes is necessary to be able to capture the boundary. Even constrained Delaunay

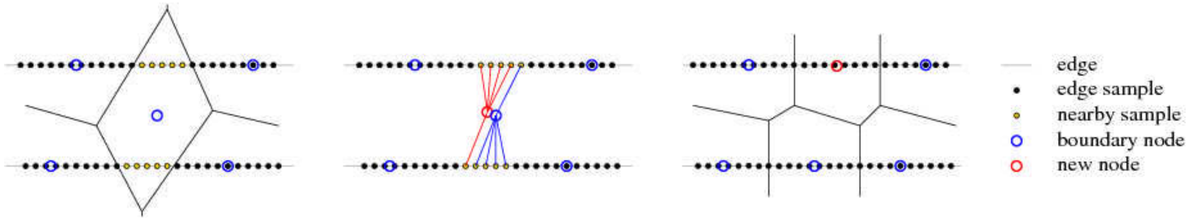


Fig. 7: Split example.

triangulations can need extra nodes (Steiner nodes) just to be able to construct a triangulation in the first place, without any consideration of quality. The number of nodes obviously depends on the complexity of the boundary, with a more complex boundary calling for more nodes. The number of nodes used in a constrained triangulation, such as produced by TetGen [14], is of course the bare minimum. As we build a conforming, instead of constrained, triangulation and we strive for quality elements, we need many more nodes. The required number depends on the local feature size. The average edge length in the triangulation cannot be longer than the lfs in a successful capture of the boundary. Since we need some flexibility, the number of nodes should be such that the average edge length is smaller than the lfs by at least a factor two. As we want all corners to be represented by the mesh, we add all corner samples explicitly to the mesh as nodes, and we effectively ignore them during the boundary procedure.

4.3 Node Splitting

Since we are working with heuristics, the use of a reasonable number of nodes at the start is still no guarantee that locally always the right number of nodes is available for capturing the boundary accurately. We can, however, during the boundary phase of the optimisation loop detect symptoms of failure to represent the boundary. As we have detailed already, each boundary node is normally pulled upon by multiple quadrature samples. This helps to balance their spatial distribution on the edges and the faces of the model. However, nodes should only be balanced within a single face or edge. Whenever a node is pulled on by quadrature samples belonging to multiple edges or multiple faces, the node will end up hanging between the two and thus contribute nothing to the representation of the boundary. Categorising all the quadrature samples into sets that belong to the same edge or face, we can count the number of sets that are pulling on a specific node. If more than one set is pulling on the node, this indicates that there are locally too few nodes to represent the boundary such that we can be reasonably sure that it can be recovered from a Delaunay triangulation.

If this is the case, we can add a second node right next to that node, effectively splitting it, leaving one node for one edge or face, and one for the other. This process is illustrated in Fig. 7. From left to right we see 1) a node that is pulled on by samples of two different edges, 2) the node split; the new node is placed randomly at a small distance from the original node, and 3) the final situation wherein each node is only pulled on by samples from a single set of edge samples and thus balanced within that edge. In order to carry out this procedure, we must have categorised all samples according to their edge or face of origin. It takes a couple of iterations to reposition the nodes in the vicinity of the split. That is why we split nodes only at certain iterations of the optimisation loop, and always continue after this with a couple of iterations in which no splits are performed.

The effectiveness of the approach is illustrated by Fig. 8. Fig. 8(a) shows the model that was used as input and Fig. 8(b) shows the resulting mesh when no node splitting is employed. This mesh has 703 nodes. The mesh in Fig. 8(c) also has 703 nodes, but it was started with just 55 nodes (54 corner nodes and 1 regular node in the interior). Through subsequent node splits, due to the pulling of edge samples from multiple sets, the number of nodes was increased to 703. After reaching this number of nodes, no more splits were called for. These nodes all lie on the boundary. The mesh consists of 1858 tets. Comparing Fig. 8(b) and Fig. 8(c), it is clear that the node splitting considerably improves the quality of the representation of the boundary.

The previous example stresses the effectiveness of the splitting procedure, but we do not recommend the construction of meshes with the use of relatively many splits. A small number of splits is acceptable and can further strenghten the likelihood of a successful recovery of the boundary, but a large number of splits should be interpreted as an indication that we might have distributed too few nodes or that the geometry poses difficulties in the creation of a conforming Delaunay triangulation. We therefore only split nodes hanging between edge samples from different sets, as splitting nodes on their proximity to different surfaces rarely leads to just a few splits. A small fraction of the nodes can be closest to boundary samples from two different surfaces, without leading to any problems.

4.4 Preparation for Mesh Extraction

Before starting with the actual mesh extraction, which will be described in the next section, we want to be sure that all boundary nodes actually lie on the boundary of the control mesh. The boundary nodes are those nodes that were identified as such during the last iteration of the optimization loop, which means that they were the closest node to at least one of the boundary samples. Given that we split nodes when samples from multiple edges are pulling on it and that edge samples are handled after the surface samples, all nodes that are attracted by an edge actually end up on that edge. Since we do not split nodes pulled on by samples from multiple surfaces, it might occasionally happen that a node is caught between two surfaces. An example of such a node is illustrated in Fig. 9. The node in the center of the 'dent' is a boundary point. Two relatively flat tets cover it, but they have been removed for the illustration. This node is being pulled on by a quadrature sample from both the horizontal and the vertical face. The edge samples all had other nodes closer. If the covering tets are not removed, this particular example does not yield a wrong mesh, but it does affect our reasoning about the mesh extraction where we expect all boundary nodes to actually lie on the boundary of the control mesh.

Therefore we project all boundary nodes to the boundary of the control mesh, before starting the mesh extraction. For all nodes lying in flat faces, the projection distance will be zero or negligible. Boundary nodes caught between faces, as in Fig. 9, will have a larger projection distance. In our experiments we noticed no problems related to this kind of projection. In particular if the number of such nodes is large, we could alternatively decide to project only if the projection distance is small; if a node is not close to the boundary, we change its status from a boundary node to an interior node.

This projection of boundary nodes to the boundary of the control mesh, also affects another set of nodes, namely those in curved faces. Since each node is placed were it is being pulled to at average, it will not always be placed exactly on the boundary of the control mesh. In concave areas, the node will be located slightly outside the boundary of the control mesh, whereas in convex areas the node will be located slightly to the inside of the boundary of the control mesh. The deviation, and thus the projection distance, depends on the local number of nodes that is used, relative to the local curvature.

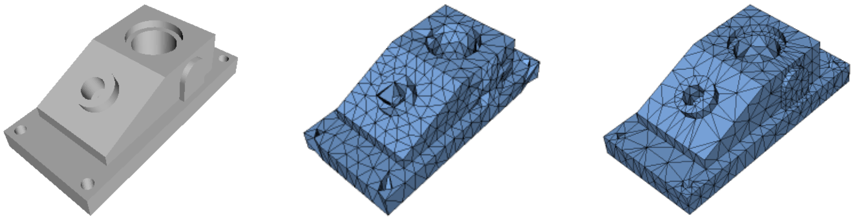
If necessary, after projection the mesh connectivity will be updated to keep the Delaunay property.

5. ENHANCED MESH EXTRACTION

In the previous section we have discussed how to increase the likelihood that a correct boundary is present in the Delaunay mesh. Now we will discuss the enhanced mesh extraction procedure.

As we have seen in Section 3, the original mesh extraction procedure does not always yield correct results. Leaning on the knowledge that all boundary nodes actually lie on the boundary, our adapted version of the decision procedure is as follows:

1. All tets are assumed to be inside at the start. On inspection we can decide to remove one.
2. All tets that have at least one interior point as one of its four nodes are inside, and therefore will never be removed. We thus only consider removing those tets that have four boundary nodes.
3. If the centroid of a tet that has four boundary nodes falls outside the control mesh, then the tetrahedron is directly considered outside.
4. The only tets we are left with are those that have four boundary nodes and the centroid inside the control mesh. Assuming that a correct boundary is present, we must conclude that these tets are inside; a tet that has its centroid inside the mesh is either completely inside or it intersects the boundary. Only if a tet has a volume-length ratio smaller than 0.1, we add it to a list to be considered for a *clean-up peeling*.



(a) Input model (b) Mesh with 703 nodes without node splitting (c) Mesh started from 55 nodes with node splitting

Fig. 8: Model based on the anc101 model.

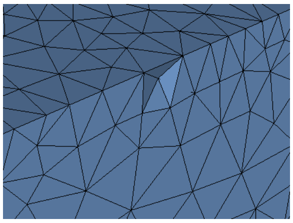


Fig. 9: Boundary point not on boundary.

5. Clean-up peeling: we want to remove very flat tets that might lie on the boundary and can be removed without negatively affecting the quality of the boundary representation. We try to remove the tets iteratively, by inspecting for removal only those tets that are currently considered on the boundary. If such a tet has two or more faces on the current boundary, it is removed. If it has only one face on the current boundary, it is only removed if its volume-length ratio is smaller than 10^{-4}. This process is continued until no more tets can be removed.

The decisive values 0.1 and 10^{-4} for the volume-length ratio are chosen based on our experience. They might not be optimal to clean-up the flat tets of every mesh. Lower values of these parameters indicate a more conservative approach. This procedure is justified as follows.
We know that all boundary nodes actually lie on the boundary of the control mesh and the remaining nodes in the interior. All tets that have at least one interior node must obviously be inside.
For the remaining tets, with four boundary nodes, the decision is firstly based on the location of the centroid of the tet. We consider two cases: a convex area and a concave area. In a convex area, if the centroid of such a tet is inside, the tet must be inside. In a concave area, if the centroid is outside, the tet must be outside. See Fig. 10 for an illustration in 2D near a convex part of the boundary. The gray triangle elements are considered inside, because their centroid is inside the control mesh (marked in red). The concave case is identical, except for the reversed orientation of the boundary. In that case the gray triangle elements are considered outside, because their centroid is outside the control mesh. This reasoning holds as long as the number of nodes is not larger than the number of samples from the control mesh, a requirement that is of course met.
In strictly flat areas there should be no elements formed between four boundary nodes. However, because of finite machine precision, such elements sometimes *do* appear, such as depicted in Fig. 4(c). It is these elements that we aim to remove during the clean-up peeling, together with very flat tets in convex areas. The elements that have only boundary nodes inside a convex area tend to be flat. If they are very flat, it is better to remove them. In case the flat element does not lie on the boundary, we could attempt to remove it by inserting the circumcenter of the tet as an interior node of the mesh.
Fig. 11 shows the meshes of the models that were not recovered well with the original extraction procedure, as they are extracted by the procedure just described. Fig. 11(c) shows the mesh of the model of Fig. 3(c) from the curved backside. Exactly the same triangulations were used for comparing the effectiveness of the recovery. The described procedure handles the extraction of these meshes correctly.
Despite the methods described in Section 4 to achieve this, there is no guarantee that the correct boundary is present in a mesh. We can try to detect this after the extraction procedure. A simple way to do this is to traverse all the boundary faces of the mesh after the extraction and compare the normal at the centroid of each face, with the normal of the control mesh for the projection of the centroid to the boundary of the control mesh. If the deviation between the normals supersedes a certain threshold, depending on the problem at hand, a problem with the boundary is likely. A relatively large distance between the centroid of a face on the boundary and its projection to the control mesh is also an indication of a potential problem. We have not investigated the robustness of using these two measures to detect

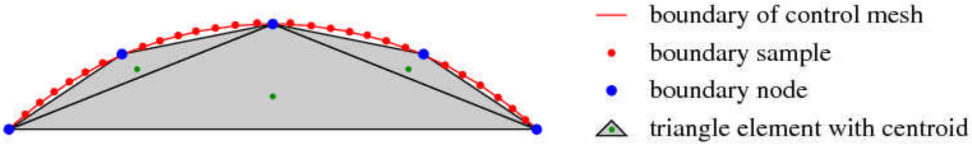
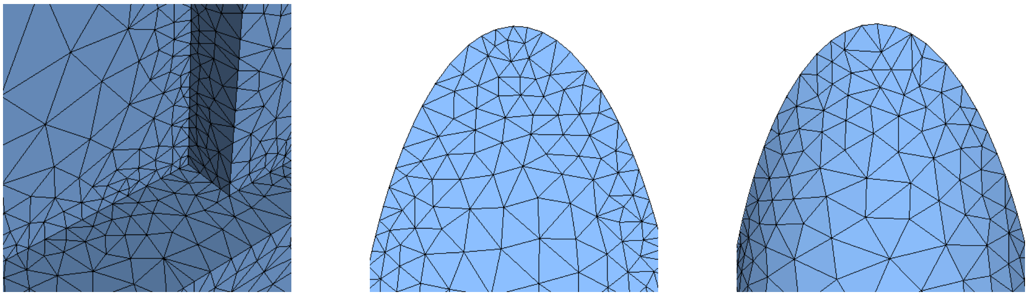


Fig. 10: Triangles and their centroid near a convex part of the boundary.



(a) Near edges in concave areas (b) Near small dihedral angle; front (c) Near small dihedral angle; curved backside

Fig. 11: Successful boundary recovery.

problems with the boundary recovery, but given the plethora of existing methods to compare boundaries, we feel that this is a surmountable problem. Most of these cases can be solved by the insertion of a single extra node.

6. PRACTICAL CONSIDERATIONS

Various aspects influence the effectiveness of the VTM method and the quality of the resulting elements. We discuss the most prominent considerations for applying the method.

As explained in Section 3, having more boundary samples, relative to the number of nodes, increases the likelihood that the boundary is well represented by the mesh. With 10 samples locally around each node, problems with the boundary are rare in our experience. In case of a uniform sizing field, this amounts to 10 times as many samples as the number of nodes that ends up on the boundary. If mesh sizing is employed, either the density of the samples needs to follow the sizing field or the ratio between the number of samples and boundary nodes needs to be higher. For models with a simple geometry, less boundary samples can suffice. Concave regions usually require a higher number of samples with respect to the number of nodes than convex regions.

Since the final mesh is a Delaunay triangulation that approximately conforms to the boundary of the control mesh, it is a requirement that the model admits a conforming Delaunay triangulation with a reasonable number of nodes. Models that require many nodes for their conforming Delaunay triangulation, will be hard to mesh with VTM or require at least as many nodes, possibly more than is acceptable for the analysis. Sharp dihedral angles and thin slits are most often the cause of a failure to represent the boundary or of a prohibitive numbers of nodes.

Most realistic and useful mechanical models do not exhibit geometric features that prevent the successful construction of a good quality conforming Delaunay mesh. All these models can be meshed with excellent quality elements by application of the presented approach. The optimization of element quality is for a substantial part effectuated by the relocation of interior nodes. Therefore the method is most suitable for creating meshes that have a substantial number of nodes in the interior or, more in general, many nodes relative to the complexity of the geometry.

The method optimises the elements for quality. However, we have made no attempt to specifically optimize the quality of the worst elements. The elements with the worst quality make up only a small percentage of all tets and most of them have more than one node on the boundary. By simple flipping we can increase the volume-length ratio of the worst elements in the mesh substantially. In our experiments we always succeeded to get it above 0.1, but higher values, depending on the complexity of the geometry, are not uncommon. If desired, the quality can be further improved by applying aggressive tetrahedral mesh improvement [8]. Since the number of poor quality tets is low, we expect that large improvements to the lower bound of the quality can be achieved in a short time.

7. EXAMPLES

To test the validity of our enhancements to the VTM algorithm, we have meshed many models. We show some more examples here. Fig. 12 shows the mesh of a gear model and Fig. 13 its mesh quality. The mesh has 20179 nodes and 77027 tetrahedrons, and a sizing field was used for its construction. Fig. 13(a) shows the distribution of the volume-length ratio for all tets and Fig. 13(b) the distribution of the minimum and the maximum dihedral angle for all tets. The regular tetrahedron has a dihedral angle of approximately 70.53° between all faces. Both small (close to 0°) and large (close to 180°) angles are detrimental to the quality of the analysis. From both graphs we gather that the majority of the elements has a shape close to that of a regular tetrahedron. The model boundary is correctly represented.

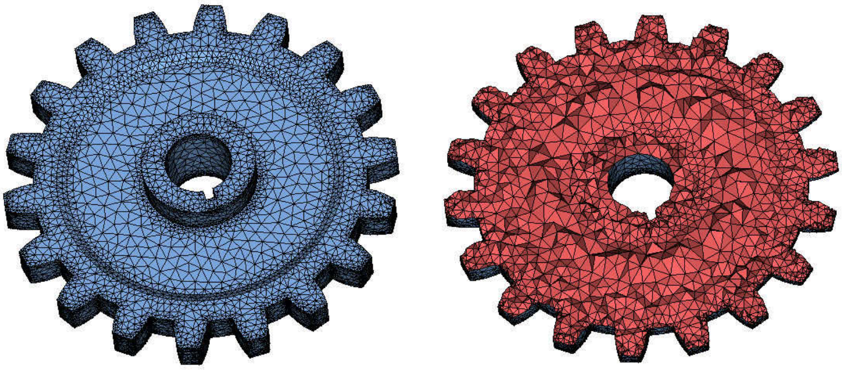
Fig. 14 shows two more examples, both created with a uniform sizing field. Both models have a boundary with several curved areas. The mesh in Fig. 14(a) has a relatively large volume with many interior nodes, whereas the other mesh has virtually no interior nodes. The quality distribution of these meshes is similar to that of the gear, with the one of the tube being slightly worse as it has few interior nodes. The large majority of the elements, however, is excellent and the boundary is correctly represented.

8. CONCLUSIONS

With variational tetrahedral meshing, high quality meshes can be generated. This is achieved through an optimisation procedure that makes changes to both the boundary and the interior nodes, alternating between the
two. The method as proposed originally has deficiencies that make it unsuitable for the generation of meshes of mechanical models for finite element analysis. This is mostly due to the boundary being represented incorrectly. We have presented several enhancements to overcome the deficiencies. With these enhancements, variational tetrahedral meshing becomes an attractive method to create good quality tetrahedral meshes of mechanical models.

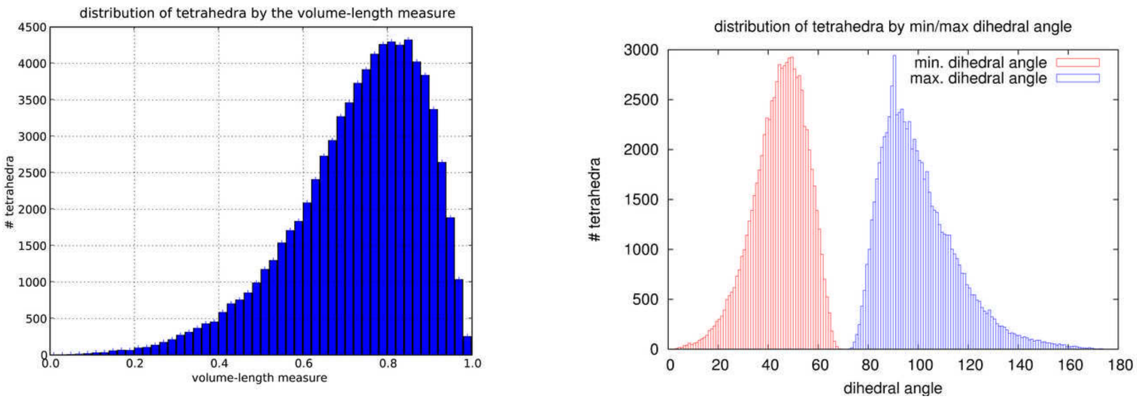
Some models, however, remain difficult to handle with this method. Since the resulting mesh is an approximate conforming Delaunay mesh of the control mesh, many elements can be needed to correctly represent areas where different parts of the boundary are close together. A constrained Delaunay mesh would use less elements in such cases. If a dihedral angle is too small, in particular in a concave region of the model, then we get either a cascade of node splits, or the boundary is likely to be misrepresented somewhere near the edge of the dihedral angle.

The enhancements to VTM come at some computational cost, but the running time of the new algorithm is still of the same order as that of the original algorithm; both are dominated by the cost of continuously updating the Delaunay



(a) Mesh of gear model (b) Interior view of gear mesh

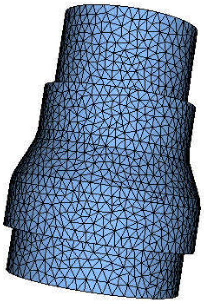
Fig. 12: Gear example.



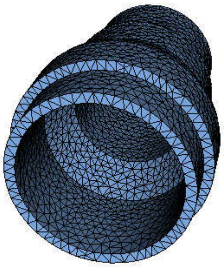
(a) Volume length ratio (b) Minimum and maximum dihedral angle

Fig. 13: Quality of gear mesh.

(a) Mesh of a construction piece



(b) Mesh of tube; side view



(c) Mesh of tube; bottom view

Fig. 14: Two more examples.

triangulation, which scales very well. The exact running time depends on many factors. The most important ones are the number of nodes, boundary samples and iterations of optimization. Using a sizing field can increase the setup-time substantially. During the first couple of iterations the overall quality of the mesh increases quickly, and then the improvements become smaller. Whether extra iterations are worth the time is thus a highly subjective matter. Most of the models we tested, with 20 iterations and at least 5000 nodes, could be meshed in a matter of minutes, with none taking more than an hour. In our view, the higher computational cost will not be prohibitive for most models in practice, since better meshes will often lower the time spent on analysis. The method has element quality as its principal objective and this comes at
the cost of a higher running time. We are not aware of any meshing method that achieves the quality distribution that VTM offers, but at significantly less computational effort.
We are currently looking into the efficient local remeshing of models with enhanced VTM after model modification, exploiting a method to determine the difference between two feature models [15]. This will further increase the attractiveness of VTM for meshing mechanical models for FEA.

9. ACKNOWLEDGEMENTS

This research has been supported by The Netherlands Organization for Scientific Research (NWO). For the implementation CGAL [1] and TetGen [14] have been used. CGAL was mainly used for its robust geometric predicates and Delaunay triangulation. TetGen was used to create the control mesh. The meshes of the gear (Fig. 12), the construction piece and the tube (Fig. 14) were based on example models included with HOOPS [2].

10. REFERENCES

[1] CGAL: The Computational Geometry Algorithms Library, http://www.cgal.org/, 2007.

[2] Tech Soft 3D: HOOPS, http://www.hoops3d.com/products/3daf.html, 2006.

[3] Alliez, P.; Cohen-Steiner, D.; Yvinec, M.; Desbrun, M.: Variational tetrahedral meshing, ACM Transactions on Graphics, 24(3), 617-625, 2005.

[4] Amenta, N.; Bern, M.: Surface reconstruction by Voronoi filtering, Discrete and Computational Geometry, 22(4), 481-504, 1999.

[5] Chen, L.; Xu, J.: Optimal Delaunay triangulations, Journal of Computational Mathematics, 22(2), 299-308, 2004.

[6] Chen, L.: Mesh smoothing schemes based on optimal Delaunay triangulations, In Proceedings of the 13th International Meshing Roundtable, 109-120, Williamsburg, VA, September 2004.

[7] Du, Q.; Wang, D.: Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations, International Journal for Numerical Methods in Engineering, 56(9), 1355-1373, 2003.

[8] Klingner, B. M.; Shewchuk, J. R.: Aggressive tetrahedral mesh improvement, In Proceedings of the 16th International Meshing Roundtable, Seattle, WA, October 2007.

[9] Li, X.-Y.; Teng, S.-H.: Generating well-shaped Delaunay meshed in 3D, In SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on discrete algorithms, 28-37, Philadelphia, PA, USA, 2001, Society for Industrial and Applied Mathematics.

[10] Persson, P.-O.: Mesh size functions for implicit geometries and PDE-based gradient limiting, Engineering with Computers, 22(2), 95-109, 2006.

[11] Quadros, W. R.; Shimada, K.; Owen, S. J.: Skeleton-based computational method for the generation of a 3D finite element mesh sizing function, Engineering with Computers, 20(3), 249-264, 2004.

[12] Shewchuk, J. R.: What is a good linear element? Interpolation, conditioning, and quality measures, In Proceedings, 11th International Meshing Roundtable, 115-126, Ithaca, NY, September 2002.

[13] Shimada, K.: Current trends and issues in automatic mesh generation, Computer-Aided Design & Applications, 3(6), 741-750, 2006.

[14] Si, H.: TetGen, a quality tetrahedral mesh generator and three-dimensional Delaunay triangulator, http://tetgen.berlios.de/.

[15] Sypkens Smit, M.; Bronsvoort, W. F.: The difference between two feature models, Computer-Aided Design and Applications, 4(6), 843-851, 2007.