Interactive Cutting of Soft Tissue Models for Surgery Simulations

（申请上海交通大学工学硕士学位论文）

系　　　别	：	电子信息与电气工程学院计算机系
学科专业	：	计算机应用
研究方向	：	虚拟现实
研　究　生	：	Jalda Dworzak
导　　　师	：	顾力栩

二〇〇七年三月

Interactive Cutting of Soft Tissue Models for Surgery Simulations

Thesis submitted to

Shanghai Jiao Tong University

in partial fulfillment of the requirement

for the degree of

Master of Science

by

Jalda Dworzak

(Computer Application)

Thesis Supervisor: Prof. Gu Lixu

March, 2007

Interactive Cutting of Soft Tissue Models for Surgery Simulations

摘　要

做手术通常需要高超的能力和技巧。医生为了准确而安全地执行手术，往往需要大量的练习。传统的训练资源包括人的尸体或者动物，这些资源数量有限，价格昂贵，并且具有道德上的争议，并不符合未来的需求。基于这些原因，手术仿真系统的开发尤其具有意义。无论是在开腔或是微创手术中，切割操作都是基本的手术步骤之一。因此，仿真系统中需要有效的切割算法来快速准确地改变器官模型的拓扑结构，同时还要尽可能地维护可形变模型的稳定性。人们提出了一些满足上述条件的算法。

本文提出了三种切割算法，它们比传统算法更好地保证了网格质量，降低了模型的复杂度。这三种算法在模拟切入时都采用了基于对称模式或者基于最小元素模式的细分方法。它们还采用了状态机架构以管理渐进式切割的临时细分。

针对四面体网格模型，本文提出一种有效的混合型切割技术。它结合了渐进式和非渐进式切割算法的优点，前者具有良好的视觉效果，后者具有稳定和速度快的特点。渐进式切割算法被用于物体

表面，而非渐进式切割算法被用于物体内部。这种混合算法保证了切割中的体网格的质量，增加了可形变模型的稳定性。非渐进切割过程中还融入了弹回操作。

此外，本文还提出了其他两种渐进式切割算法。第一种算法采用了对称细分模式和最小元素模式来实现快速转换。第二种算法结合了最小元素细分和状态机架构。这种方法也被用在了混合切割算法中的渐进式切割部分。这两种算法的目的是减小切割过程中网格的复杂度。

本文提出了一种通用的评价方法，用于比较上面提及的三种切割算法。该方法包含了一个全局同质的标准和一个局部同质的标准。前者揭示了网格中体积的分布和大小，后者揭示了单个四面体的形状质量。

关键词： 虚拟手术，渐进式切割，非渐进式切割，四面体细分，拓扑网格的改变，软组织仿真的稳定性

Interactive Cutting of Soft Tissue Models for Surgery Simulations

ABSTRACT

Surgery, as a delicate and dangerous task, requires conscientious and extensive training in order to acquire the indispensable skills. Conventional resources for educational training environments, like human corpses or living animals, are limited, expensive, and not likely to withstand the ethical sociological and commercial advances in the future. For this reason, the prospect of simulating surgery by means of a virtual setup is more than promising. In open and minimal invasive surgery, cutting is a paramount and fundamental task. Consequently, efficient cutting methods that modify the anatomical representation accurately, fast and that maintain soft tissue simulation stability at the same time are needed. Few methods, which fulfill these requirement adequately have been proposed so far.

Three cutting methods are presented in the context of this thesis, that improve previous methods in their effect on the model's complexity and the mesh quality. All methods deploy subdivision for incision modeling part via symmetric patterns or minimal element patterns. The methods are embedded into a state machine framework that manages temporary subdivisions for progressive cutting.

An efficient hybrid cutting technique is suggested for volumetric tetrahedral meshed mass spring models. It combines two cutting methods, the visual appealing progressive cutting and the in stability and performance superior non-progressive cutting, for the benefit of both their advantages. Whether progressive or non-progressive subdivision procedures of the hybrid technique are applied to a model's element depends on its location. The approach reduces the deterioration of the mesh quality and improves the soft tissue simulation stability considerably by avoiding degeneracies due to temporary subdivisions and applying snapping to the non-progressively cut part of the mesh.

Additionally, two progressive cutting methods are presented. One applies symmetric subdivision patterns and minimal elements patterns in case of shortcut transitions. With the other algorithm minimal element subdivision is integrated into the state machine framework. This method is also used for the progressive cutting part of the hybrid approach. The intention of both methods is to reduces the decrease of the mesh complexity that is inflicted upon the model by cutting.

An universal and expressive metric for the evaluation of the mesh quality is proposed and deployed for comparison of our methods. It consists of a global homogeneity criterion that reveals information about the volume distribution and relations of volume sizes in the mesh. Moreover, it includes a local homogeneity criterion that sheds light on the shape quality of a single tetrahedron.

KEY WORDS: virtual surgery, progressive cutting, non-progressive cutting, tetrahedral subdivision, topological mesh modification, soft tissue simulation stability

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

In the last decades, computer graphics evolved rapidly in virtual reality technologies. Simulations have been developed in many areas, particular in fields where the driving forces are cost reduction, increase of effectiveness, shortage of resources or where the risk of disasters due to educational errors is unacceptable. In [3] some interesting application fields for simulators are pointed out that have been pursued in the past. Vehicle simulation for training, like steering or flight simulators, training simulations for space admission testing and special arrangement simulators for architectural and spatial design, just to name a few of them.

Surgery is and has always been a difficult and hazardous task. It requires conscientious planning and long term training to acquire the indispensable skills. However, at the same time, the resources to provide the educational environment to the surgeon are as costly as they are limited. In order to assure a consistent training, the samples need to be as close to reality as possible. Living persons are certainly not at disposal for surgical training and experiments. Especially in light of the issue that there is a certain margin for human errors that has to be considered. Facing these facts, substituting living animals for human life appears as a good deal. But yet again, it raises fundamental ethical questions such as if animals are dedicated to suffer for an advantage that only mankind can benefit from. In addition, animal physiology tends to deviate substantially from human physiology. Consequently, the training accuracy is often insufficient for human medical training. Similarly and as a second alternative, conducting training on human corpses, as commonly practiced, is often equally inaccurate, expensive, applicable for a limited number of students and very questionable from an ethical point of view. At the bottom line, the current training methods are not likely to withstand the ethical, sociological and commercial advances in the future. Therefore, the prospect of simulating surgery in a virtual setup is more than promising and has been pursued in recent years. Cutting is one of the most often performed and fundamental tasks in open and minimally invasive surgery. Efficient cutting methods that modify the underlying anatomical representation are needed and few methods that modify soft tissue models accurately, fast and maintain soft tissue stability in an acceptable way have been proposed so far. Cutting in context of surgery simulation is a challenging research area and worthwhile to be pursued by this thesis.

1.2 Main Contributions

The main contributions of this work are outlined in the following list.

- We introduce a novel hybrid cutting technique for volumetric tetrahedral meshed mass spring models. A progressive cutting method is applied to the model's surface elements, where the core of the model is cut non-progressively. Our work combines the advantages of both subdivision methods into one technique. It leads to an increase in the stability of the soft tissue deformation simulation and enhances interactivity, by reducing the cutting complexity.
- For stability increase, snapping was implemented for non-progressive part of the hybrid algorithm. In contrast to the work of Steinemann et al., snapping is also applied to cut topologies that do not lead to the dissection of an element.
- The state machine approach for progressive cutting via subdivision by symmetric subdivision patterns, similar as proposed by Biesler et al. [4], was implemented and improved by the following enhancements:
  - We extended the state machine framework by shortcut transitions, which use minimal element subdivision pattens. These shortcut transitions reduce the complexity of the mesh.
  - The search for state transition sequences are optimized by precalculation of a predecessor matrix before the surgery simulation starts.
  - Transition between temporary subdivisions have been realized entirely without removal operations. This is achieved with the help of temporal node assignment to the temporary sub-elements.
- A third cutting procedure is proposed that combines the state machine framework

with minimal elements subdivision patterns, in order to keep the mesh complexity low. Progressive cutting via symmetric patterns as realized in the method mentioned in the point above has the advantage that it gets by with minimal removal operations, but it increases the element count rapidly. In contrast to the original progressive minimal element approach by Mor and Kanade [5] our state machine approach with minimal elements subdivision patterns forgoes removal operation when possible.

- Prototypes were implemented for all three methods mentioned above as well as the ordinary progressive cutting algorithm. We developed universal and expressive metrics to evaluate cutting algorithms on the same basis. These give evidence about
  - how they modify the mesh complexity,
  - in which magnitude they decrease the mesh quality, and
  - what their impact on the simulation stability is.

1.3 Thesis Outline

This thesis is organized in the following manner. In Chapter 2 on page 5 we generally introduce the problem of cutting and provide an overview of the related work organized by the main aspects of cutting, like model representation, means to define a cut path and the geometrical problem of incision modeling. An overview over existing, competitive methods is given and cutting of soft tissue models is discussed. Furthermore, we embrace improvement directions that address problems, which arise due to cutting in deformable meshes or due to increase of mesh complexity.

Chapter 3 on page 29 deals with the methods we propose to realize a cutting framework for our surgery simulation system and that have been implemented within the scope of this thesis. Three different cutting modules are introduced that can be used interchangeably within the surgery simulation system. We discuss requirements and architecture of our system and address concepts that are used in all three methods, cut path definition and intersection calculations. Afterwards, the three cutting methods are presented. With the third algorithm we introduce a novel hybrid cutting method that applies different subdivision methods to the model elements dependent on their location.

In Chapter 4 on page 62 we present an analysis of our methods and discuss experimental results. An evaluation according to defined metrics is conducted.

Finally, in Chapter 5 on page 82 a summery and the conclusion of this thesis is given.

Chapter 2

Related Work

This chapter provides the reader with the fundamentals of mesh cutting in a bottom-up manner such that the remainder of the thesis becomes self-contained. An overview over the problem of cutting in the context of deformable models with application to surgical operation on soft tissue is presented. We start off by briefly describing the general procedure of cutting. The major part of this chapter then examines cutting according to five aspects that form the problem of cutting. More concretely, we present the solutions how existing methods address the following five constituents of cutting: how an incision along a cut path is modeled, in what way a cut path can be interactively defined by a user, at which particular time cutting is performed, how the model is represented and what methods can be chosen to realize the deformation simulation. The chapter concludes with improvement directions of recent years.

2.1 General Cutting Procedure

This section introduces the essential parts of the cutting procedure that are similar for most of the present methods. The cutting process works on a virtual artifact that we call model. A model is a geometrical representation of an anatomical structure that undergoes topological modifications and simulates soft tissue behavior. There are several ways to define such a model representation. However, the majority of the models are either just a simple surface or have more complex volumetric mesh representation (see Section 2.2 on the next page and Section 2.6.1 on page 22). The location and direction of a cut is defined by a cut path representation. The means of defining a cut path is realized in most methods as interactive tools or templates which can be freely moved in virtual space. When a cut is defined, the actual intersection points between the cut path and the model have to be determined via intersection detection (see Section 3.4 on page 33). Since it is computational costly to check the entire model for intersections a prior collision detection roughly identifies parts of

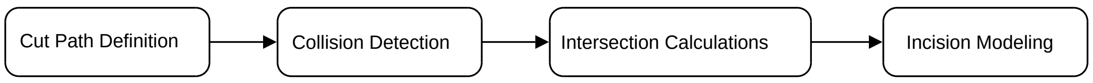


Figure 2.1: The general process of cutting.

the model that are possibly affected by the course of the cut path. The intersection detection is then only performed on this part. The next task of the the cutting algorithm is the geometrical design of the incision according to the intersection occurrences (see Section 2.4 on page 10). The geometrical incision design differs from method to method and is usually considered as the one of the most difficult parts of the problem as a solution needs to be found under regard of many constraints, such as simulation stability, speed, and visual accuracy. Figure 2.1 shows the the parts in chronological order.

The next sections focus on how existing methods differ from each other and compare assets and drawbacks.

2.2 Model Representation

The subject to surgery simulation is a virtual representation of an anatomical structure. Volume definitions of anatomical structures, like voxel-based 3D data sets from Magnetic Resonance Imaging (MRI), Computer Tomography (CT), or Positron Emission Tomography (PET), have to be converted into a geometrical representation, as for example in the work of [6]. This is achieved by extracting shapes of specific regions of interest and prepossessing the data. A commonly used method for extracting shapes and areas of different properties from voxel-based representation is segmentation. Meshing algorithms are then applied to the surface information in order to generate a spatially discretized mesh of the anatomical structure, that consists of vertices and edges.

Different representations of deformable models have been used for cutting in soft tissue simulations. Most of simulations make use of either volumetric models [7–9, 5, 10, 2, 1, 11, 12] or surface models [13–16]. Few research has been done with hybrid or mesh-free models [17, 18].

2.2.1 Surface Models

Surface models represent the outer hull of an object and thus do not model any interior structure. They have been the first models used for topological modification, like cutting, tearing [19] and suture [20][21]. Usually a surface model is represented by a triangular mesh that is placed into three dimensional space. This way it is possible to display the exterior shape of a 3D model.

On the one hand, performing cutting on surface meshes is less elaborate than in volumetric meshes and it is computationally less complex. On the other hand, the depth information can not be visualized with surface models. The cutting algorithm of [22] approaches the problem by inserting grooves and additional surfaces along the cut trajectory while the cut is performed. This way the model appears volumetric and the actual absence of the depth of a cut can be mimicked. Considering the fact that deformation simulations of surface models have a suboptimal accuracy compared to volume models due to their lack of an interior structure, they are not applicable for authentic volumetric tissue simulation. Nevertheless, it is worthwhile to use surface models where an interior structure is not needed. For instance for blood vessels [21], gallbladder or similar organs that have a fluid-like interior.

2.2.2 Volumetric Models

Volumetric models can simulate the interior of an object thus they are better suited for soft tissue simulations. This is important because organs mostly contain parts with certain density and viscosity. Moreover, with volumetric models a more realistic biomechanical deformation can be simulated. Common are tetrahedral or hexahedral mesh representations. The majority of volumetric models are tetrahedral meshes, because they have a better simulation behavior. In addition, the graphic adapters on current computers are triangle based so that tetrahedral meshes can be simply mapped on triangle meshes at render time.

However, the memory consumption with volumetric models is higher than with surface models. Generating a homogeneous mesh by meshing is a more difficult and topological modifications and deformation simulation of volumetric models are highly complex and computational expensive [11][23][24].

Consequently, there are two factors that influence the choice between surface and volumetric models. One is the precision and accuracy of the simulation, the other is the computational complexity that has to be kept as low as possible to ensures a certain degree of interactivity.

2.2.3 Hybrids

2.2.3.1 Layered Surface Models

A hybrid method that interconnects two surface layers by edges between the vertices of both layers in order to form a volume representation has been proposed by Mazzella et al. [25]. Complex anatomical structures are modeled similar to surface models, except that the outer hull consist of two connected surface layers. Since only an outer margin of an anatomical structure can be modeled, surgical interactions can only be applied perpendicular to the surface and very shallow in depth. Furthermore, for deformation simulation only a small-area-deformation can be achieved. Thus, layered surface models are suited for skin simulation and the simulation similar shallow organs. The method described previously was advanced by Bryuns et al. [26] with modeling multi-layered objects with this technique. Here it is also possible to severe layers by cutting interconnecting edges between surface layers.

2.2.3.2 Surface Mesh with Ridged Interior

In [27] it has been proposed to employ a model representation that is a hybrid structure of an interior rigid body and a surface mass-spring model that is connected to the rigid interior by so called home-springs. Incision surfaces to both sides of the cutting are represented as grooves that are proportional in depth to the penetration depth of the cutting tool. The work focuses on joining the start and end points of loops that have been cut into material. A fast simulation of the surface tissue part can be achieved by simply neglecting the interior simulation. However, by modeling just the outer deformable hull of the model, the simulation is restricted to very shallow and unrealistic deformations. Moreover, no dissection of the material can be arranged.

2.3 Cut Path

Surgery simulation systems need to provide an interface for surgical intervention. Cutting, in particular, requires to disconnect elements from each other or decompose the elements in that way that they are spatially incohesive along the cut path. The user of a surgical simulation system should have interactive control over the position and shape of the cut. Three different approaches realizing this interactivity have been implemented so far.

2.3.1 Seed Points

One approach is to successively define two or more seed points on top of the model's surface as shown by Wong et al. [28]. These seed points can be manually set by the user. The seed points are then connected using either Dijkstra's shortest path algorithm, the Euclidean shortest distance or computing the geodesic path across the surface. The disadvantage of Dijkstra's shortest path is that only mesh points can be used, the cut path runs along existing edges and thus the path becomes angular.

The other algorithms enable a relatively smooth cut. The drawback of seed points method in general is that no cutting depth can be defined when applied to volumetric meshes or it has to be defined in an extra step as in the work of Salb et al. [29]. Another drawback is that a cut can be visualized no earlier than when the second sample point is defined. Thus, no steady-going cut can be visualized.

2.3.2 Template Method

The intersections of the model with a predefined template, for instance a primitive like a plane or triangle, into the model can serve as cut path definition [30][31]. The cut path is not defined by the templates movement, but by the templates shape and position. The user interactively places the primitive into the model. Then the user triggers a signal that executes intersection detection between the template and the model and the cutting procedures that follow.

2.3.3 Virtual Tool

The most intuitive and common approach is a virtual cutting tool that can be moved through the virtual environment. The movement of the cutting tool over a certain time interval defines a cut path . There are different ways to model such a virtual tool.

A single point is suited to be used to cut surface meshes. The method proposed in [32] uses a point for cuts on surface meshes in 3D. The points view port position is projected onto the closest face of the mesh. The point can be dragged over the surface and thus defines the path of the cut. This tool representation is certainly not suitable for volume meshes.

For volume models, a primitive like a single segment or a triangle can be employed. It is also possible to use an object consisting of multiple primitives that approximates the shape of an surgery tool which is required to display an actual scalpel. The benefit of the first option is that it supports simple intersection tests. For the letter method a simplified model for intersection computations, like central axis, is often used. The rendering of the complex tool supports the realism of the simulation, but also requires higher computational resources.

2.4 Incision Modeling

Cutting methods can be classified according to the way they realize a cut on the most elementary level. That means how they model an incision into the model's material. Common methods can be distinguished between techniques that decompose their elements into smaller elements along the cut trajectory and those that avoid decomposition.

2.4.1 Non-decomposing Methods

There are three methods that are characterized by the fact that they do not subdivide elements of the mesh. They have the benefit of being fast and they leave the model in a state that is numerically reliable for deformation simulations. However, the visual result is not very appealing.

2.4.1.1 Cauterization Method

The simplest algorithm proposed by [19] and [33] is a cauterization method. When a bipolar cautery instrument intersects with elements of the deformable model, it simply deletes the elements that are in intersection. So the incision gap is modeled by the freed space of the intersection tetrahedra. The work of [34] also realizes cutting on a tetrahedral mesh by deleting the intersected tetrahedra. Furthermore, their algorithm preserves the manifoldness and conformality of a mesh. The manifoldness is a topological mesh property that ensures mesh consistency and conformality of a mesh is given if the intersection of two tetrahedra is either empty, or a common vertex, edge or face. A tetrahedron can be deleted only if the manifoldness criterion for the mesh is preserved.

The cauterization method keeps the complexity of the mesh low and is easy to implement. Nevertheless, the visual result highly depends on the mesh refinement. With a low resolution mesh, bumpy faces are generated. In addition, one has to keep in mind that deleting elements from the mesh comes with an alteration of the volume, because the model is partially erased from the scene. On the contrary, cutting anatomical structures in reality physically preserves the mass and volume of the structure. The principle of the cauterization method is sketched in Figure 2.2 on the following page(a).

A more elaborate method of cutting by cauterization is proposed by [35]. It comes with a local mesh refinement near the cut trajectory that avoids creating a extended and jagged gap, but that adjusts the element size around the incision to the size of the cautery tool.

2.4.1.2 Nearest Node Methods

Another technique is to approximate the cut along existing nodes of the model that are nearest to the cut path, as depicted in Figure 2.2 on the next page(b). The nodes near the cut path are duplicated such that the adjacent elements are detached from each other [36][37]. In the 2D case, the doubled edges define the cut path while in 3D the doubled faces form the cut path. This method increases the number of nodes in the mesh, but does not introduce any new elements to the model and the volume is minimally altered due to the small replacements of the duplicated nodes away from the original node's position. However, approximating the cut this way does not describe the cut path as exactly as the user defined it. In addition, this method results in an uneven cut surface.

2.4.1.3 Snapped Nearest Node Method

The third approach without element decomposition suggested by [15] and [38] is similar to the one described above, but it overcomes the problem of the inaccurate cut surface approximation. Again, the closest nodes to the cut path are determined. Then, these nodes are displaced into the trajectory of the cutting tool and the nodes are duplicated, as illustrated in Figure 2.2 on the following page(c). The benefit is a more accurate cut representation. However, regular meshes become irregular. That leads to several difficulties when dealing with deformable models. In addition, depending on the mesh resolution the cut freedom is limited as sharp curves are likely to appear jagged .

2.4.2 Subdivision Methods

Another way of cutting that has been realized is the subdivision method. Informally, subdivision can be described as the replacement of a fully or partially cut element with a new set of sub-elements, which are not in intersection with the cut trajectory. The new sub-elements occupy the same volume as the former element except that the cohesion of the sub-elements in edge and vertices is chosen in that way that a cut is modeled

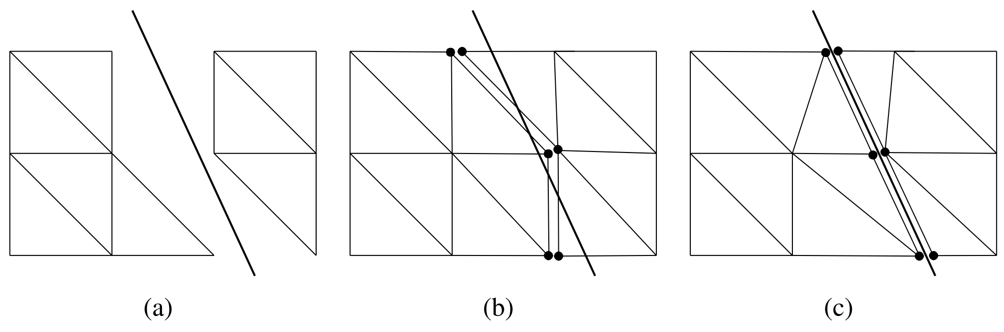


Figure 2.2: Cutting methods without element decomposition: (a) element deletion, (b) cut path along the nearest nodes, and (c) nearest nodes shifted into the cut trajectory.

along the the intersections with the cut path.

The methods using the subdivision approach normally define a finite set of subdivision patterns. These subdivision patterns help to discretize the large number of possibilities to cut an element and they restrict the subdivision to a few cases. Consequently, a subdivision pattern approximates the topology of a specific cut for an element of the model. When intersections are determined and a suitable cut pattern is found, the tetrahedron can be subdivided according to the cut pattern.

Several methods that differ in their subdivision patterns have been proposed in the past. In the following we mostly focus on the description of subdivision methods on tetrahedral volumetric meshes in particular. Methods on surface models or volumetric models that are meshed with different geometrical primitives follow the same principles.

2.4.2.1 Tetrahedral Presplit

One of the first subdivision approaches for volumetric tetrahedral meshes was reported by Bielser et al. [1]. They restrict cutting to five topological different cases. These five rotationally invariant cases are shown in Figure 2.3 on the next page. The algorithm works on a tetrahedral volume mesh. The number of intersections into a tetrahedron is constraint to one intersection per tetrahedral edge and face. This means the the cutting tool can only cut a tetrahedron once per simulation, thus cuts can not be joined and two incision that cross each other can not be modeled appropriately. Additional incisions are neglected.

In this work, only one single generic pattern is introduced that is adapted to the particular requirement of the cut case. The generic pattern is realized as a tetrahedral presplit of every element into 17 sub-tetrahedra on the data structure level for poten-

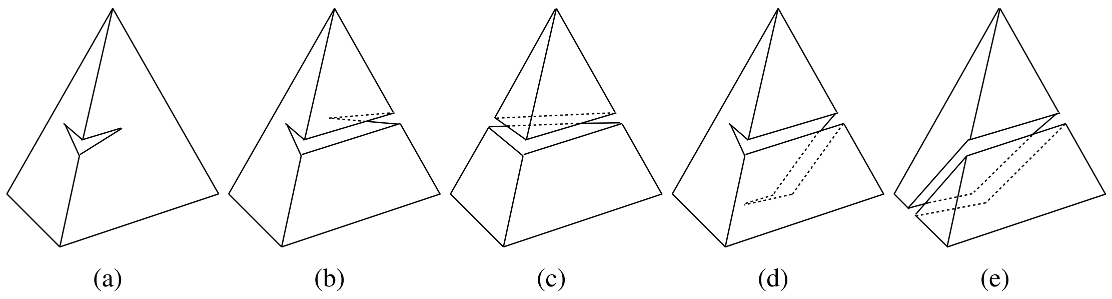


Figure 2.3: Five cut cases as described in [1]: (a) one edge cut, (b) two edges cut, (c) and (d) three edges cut, and (e) four edges cut.

tially cut tetrahedra. When a tetrahedra is split according to one of the five cut cases, the predetermined coordinates are adjusted to the current intersection point with the cutting tools trajectory and required edges, faces and new points are inserted. All the cutting operations are stored in a look-up-table (LUT) to enable efficient processing. Consequently, there exists one LUT entry for each topological different case.

However, the generic subdivision approach leads to a rapid increase in the number of elements and produces hanging nodes without adjacency to other tetrahedra. Hanging nodes evolve because the adjacent tetrahedra in the generic 1:17 tetrahedra split are not re-meshed and not adjusted to the explicit coordinates of the face intersections. The mesh becomes inconformal. These mesh inconsistencies certainly have effect on the physical representation and result in reduced accuracy of the deformation simulation. This subdivision method gives the user the possibility to define the cutting path with certain degree of freedom.

2.4.2.2 Symmetric Face Triangulations

In a work of [2], a subdivision method with symmetric subdivision patterns is introduced. Subdivided faces of a cut tetrahedra are restricted to three types as depicted in Figure 2.4 on the following page. These face subdivision schemes introduce two additional face midpoints per incised face that are not essential to the subdivision, but lead to more symmetry of the decomposition.

Combinations of the three face types result in the actual five tetrahedral subdivision patterns in Figure 2.5 on the next page. In contrast to [1], the subdivision is conformal with the tetrahedral representation of the model. The problem of hanging nodes is solved by subdividing only those edges and faces that were actually intersected by the cut path. The sub-tetrahedra nodes are adjusted according to the intersection points in the manner that a correct junction between the neighbouring elements can be assured.

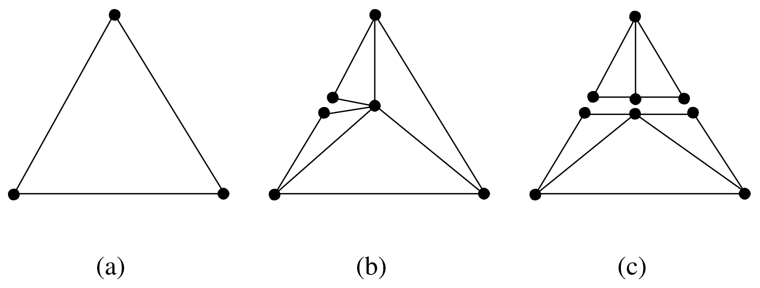


Figure 2.4: The face subdivision scheme of [2].

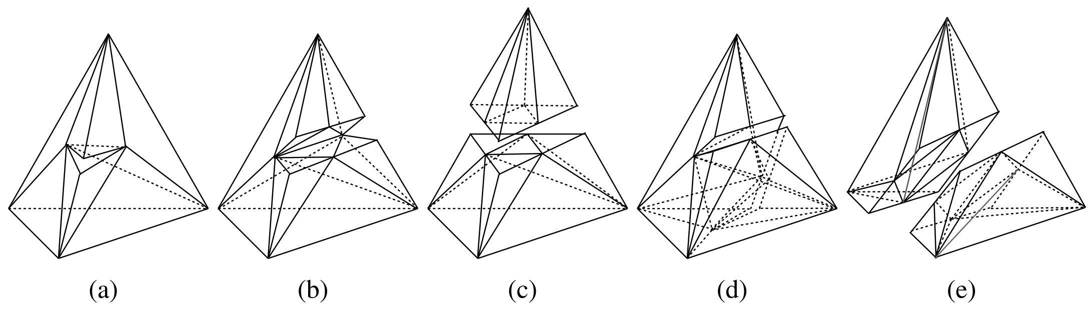


Figure 2.5: The symmetric subdivision patterns.

Accordingly, the faces of the adjacent sub-tetrahedra within one subdivided parent element do match.

The symmetric subdivision approach has the advantage of approximating the cut path more closely. In comparison to other methods, the additional face midpoint leads to an extended degree of freedom in defining the cut path and produces similar shaped subdivision elements. Similar shaped elements are an advantage, because a homogeneous mesh leads to a stable simulation behavior of the underlying deformable model. Yet a symmetric pattern has its disadvantages. The element count per subdivision is higher than with other methods and this in turn intensifies the mesh complexity and slows down the simulation speed.

The subdivision patterns of [2] have an additional drawback. Subdivision patterns Figure 2.5(b) and Figure 2.5(c) are potentially self-intersecting cases. A self-intersection scenario of case in Figure 2.5(c) is displayed in 2.6. In order to avoid self-intersections, it is necessary to conduct intersection test for the pattern in question and to replace them by alternative pattern if required. However, self-intersection tests are time consuming and preferably to be avoided.

Bruyns et al. [10] adapt these symmetric subdivision patterns and bypass self-

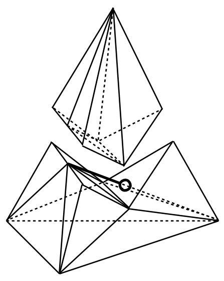
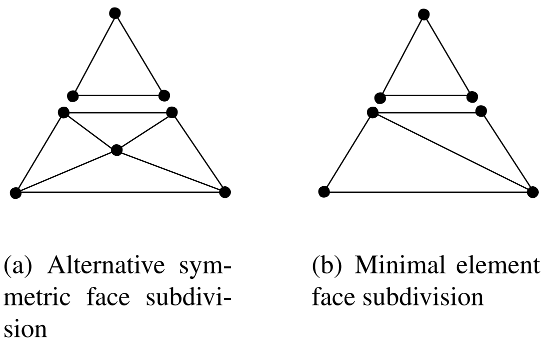


Figure 2.6: A self-intersection of a symmetric subdivision pattern.



(a) Alternative symmetric face subdivision

(b) Minimal element face subdivision

Figure 2.7: Two alternative face subdivision schemes.

intersection tests by an alternative configuration of the internal sub-tetrahedral structure. They also propose a generalized framework for interactively cutting tetrahedral meshes. All subdivision methods described so far use the same five cut cases as shown in Figure 2.3 on page 13. The work of Bielser et al. [4] also carries on using the symmetric scheme, but they extend the five cut cases with an additional case. The additional pattern takes cuts into account that cut those two edges that are do not share a common vertex.

An alternative symmetric subdivision approach was implemented by Steinemann et al. [8]. They use the same face subdivision scheme as [2] for partially incised faces as depicted in Figure 2.4 on the preceding page(b), but they deploy a different face subdivision scheme for face dissection as shown in Figure 2.7(a). Here again the combinations of the face subdivision schemes give the subdivision pattern according to the cases in Figure 2.3 on page 13.

A benefit of the alternative face triangulation technique in Figure 2.7(a) is that to some degree the shape and size of the tetrahedral elements is more regular. Yet again, the element count is not optimal.

2.4.2.3 Minimal Elements Subdivision

The need for effective simulation computation and thus a generation of as few as possible new elements on account of cutting leads to a subdivision algorithm that uses a minimal set of elements for the subdivision. This method exclusively uses the necessary evolving new points from splitting edges due to intersections with the cutting path. In other words, no symmetry points in dissected faces are used as illustrated in Figure 2.7 on the preceding page(b). The minimal number of elements that are necessary to model a particular cut topology are inserted. Apart from the advantage that unnecessary elements are not added to the mesh, there is the drawback that the mesh tends to become inhomogeneous with cutting, because the shape of subdivision elements is more irregular than with the symmetric approaches. Counter measures with the purpose to prevent insertion of bad shaped elements have to be taken. Section 2.7.1 on page 26 will introduce one of those counter measures in detail. This minimal element subdivision was used for triangular [32][27][22][36] as well as for tetrahedral [39][5][7][9] meshes, respectively.

2.4.2.4 Face Consistent Subdivision

The methods described so far do not necessarily leave the mesh consistent and conformal after a topological modification via cutting. That means for adjacent elements that the faces should be congruent with each other. Ganovelli et al. [7, 40] put special emphasis on this aspect. Their subdivision patterns are conditioned by consistent face triangulation. For adjacent tetrahedra it is required that bordering faces are triangulated in that way that the new triangles faces match. There are two possible triangulations for a split face [a,b,c,d] depicted in figure Figure 2.8 on the following page(a). One of those two triangulations is chosen for both adjacent faces. In Figure 2.8 on the next page(b) the subdivision pattern is visualized as proposed by Ganovelli et al. [7, 40] that can handle the consistent face triangulation. The pattern uses a “Steiner” point and it subdivides the tetrahedron into eight sub-tetrahedra. The conformality of the mesh is defined by a simplex structure. Additionally, Ganovelli et al. extend their cutting method to multiresolution grids. Thus, different mesh refinements dependent on the required level-of-detail can be used. Fewer new elements are created in comparison to the approach in [1]. Nevertheless, the consistent triangulation requires the introduction of more sub-tetrahedra than necessary and an overhead of storage for managing these consistencies. Consequently, not an optimal number of sub-tetrahedra are used with this approach. With this method too, precautions against badly shaped elements have

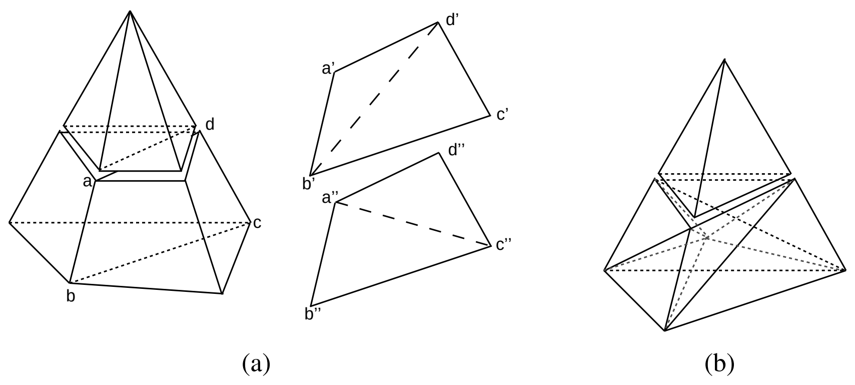


Figure 2.8: Consistent face triangulation: There are two possibilities to remesh the quadrangle [a,b,c,d] into triangles in (a). For both cut tetrahedra the same subdivision should be chosen. The subdivision pattern in (b) makes a consistent face triangulation possible with usage of a a “Steiner” point.

to be taken.

On data structure level Ganovelli et al. [7, 40] use a digit code in order to encode if an edge of an tetrahedron is cut or not, evaluation of the code gives information about which subdivision pattern has to be used. They also use a LUT approach to store the elementary cut operations necessary for a specific subdivision. The edge code evaluation leads to the direct access of an LUT-entry.

2.5 Time when Cutting is Performed

All cutting methods that use subdivision in order to model the cut can be further classified according to the point in time when subdivision is applied to the elements

2.5.1 Non-progressive Cutting

Non-progressive cutting methods are all those techniques that subdivide their elements after the tool has left the element. Thus, the tool is not in intersection anymore with the element that is supposed to be subdivided. All intersection points are predetermined before the subdivision is conducted. A lot of methods follow this approach [36][41][7][5][32]. Since the cut is not visualized for the duration of the tool’s first entry into the element until the tool’s exit a noticeable lag between the actual tool’s position and the cut occurs. Especially for low refined meshes this has an affect on the realism of the simulation.

2.5.2 Progressive Cutting

Progressive cutting methods avoid the lag-behind effect, which occurs when non-progressive cutting is performed. Progressive cutting has the advantage that the cut is modeled up to the cutting tool's position at all times. This is accomplished by already subdividing elements while the tool is still in intersection with them. That means not all intersections have been determined yet. If the tool cuts more edges or faces in one of the next simulation steps, while it moves further into the model, it is necessary to substitute the previously inserted substructure of the previous element's subdivision by a new one. This new subdivision corresponds to the new intersection situation of the element. In the following we will refer to these subdivisions that are made before the final intersection configuration for an element can be determined, as temporary subdivisions. Since more and more edges can be cut successively from time step to time step an element can pass through several cut topologies. For example, if at first one edge of a tetrahedron is cut, then a second, a third, and a fourth edge, then the tetrahedron passes through the cut cases Figure 2.3 on page 13(a), 2.3(b), Figure 2.3 on page 13(d) and finally 2.3(e). The sub-tetrahedral configuration keeps changing as new intersections are found and the subdivision is final when the tool swept completely through the element as in case Figure 2.3 on page 13(e). The first and all intermediate sub-tetrahedral configurations have to be modeled by temporary subdivisions. Thus, progressive cutting can be regarded as on-the-fly re-meshing that adapts dynamically to the topological situation of a cut mesh element. In comparison to non-progressive cutting, this procedure is more complex and elaborate than the simple once-only insertion of a substructure with non-progressive cutting. Then again, it yields a visually more satisfying result.

Two main concepts for dealing with temporary subdivisions have been realized so far. The next two sections introduce those concepts.

2.5.2.1 Removal and Replacement of Temporary Subdivisions

The first progressive approach was presented by Mor et al. [5] for tetrahedral volume meshes. Mor et al. solve the problem of the substitution of temporary subdivisions by simply deleting the hole subset of temporary subdivision that has been inserted for the cut tetrahedra. The emerging free space is filled by a new set of tetrahedra corresponding to the new subdivision pattern. For each possible change from one temporary subdivision to another specific deletion and insertion operation are defined. It is necessary to store temporary subdivision tetrahedra for subsequent deletion.

The large number of deletion and insertion operations, that are executed whenever the temporary subdivision changes, are costly and effect the performance the algorithm [4]. Furthermore the algorithm does not consider successive cuts into a model, where current incision cross the path of previous ones.

For progressive cutting an intersection detection and storage is necessary for each edge and face. The storage of the subsequent subdivision of an element is also inevitable, since this data is needed beyond the time of one simulation step. Subdivisions are made according to the number and distribution of face and edge intersections.

Similar progressive simulation methods have been realized by [27] and [22] for cutting of surface meshes. An intersected surface triangle is deleted and a temporary subdivision of the triangle is inserted at its position. Two temporary subdivision patterns are used for this purpose. One pattern serves for the case that the cutting tool enters a triangle via an edge (see Figure 2.4 on page 14(b)), the other considers the case that the tool enters the triangle via a triangle vertex.

2.5.2.2 State Machine Approach

The second approach presented in [4] confines the problem of the numerous deletion operations to a minimal number of removal operations. This is possible due to the topological properties of the symmetric subdivision pattern that are used (see Figure 2.5 on page 14). Instead of deleting tetrahedra of temporary subdivisions, new tetrahedra are successively added to the existing ones, when a tetrahedron passes from one temporary subdivision to another. The sub-tetrahedra kept from the previous temporary subdivision are adjusted to the requirements of the new one.

In addition, the work of [4] introduces the usage of a concept that assists in organizing and managing the pass-through of a tetrahedron from an undivided condition to a final subdivision, a state machine. Their algorithm also handles recursive continuation of progressive cutting. The subdivision is continued on the sub-tetrahedral level, in cases that particular incision can not be realised in the current level.

In the following the state machine concept is explained more explicitly.

2.5.2.2.1 State machine

The state machine provides a concept for handling the complex subdivision process for one tetrahedron. It tracks the different temporary subdivisions from an undivided condition (*initial state*) to the final subdivision (*end state*) and in that way it controls all necessary update operations for progressive cutting. There is one state machine for each tetrahedron in the mesh.

The complexity of the progressive cutting procedure comes from two main factors. There is a large number of different possible combinations of edge and face intersections. The subdivision patterns cover the topology of one orientation of a certain edge and face intersection distribution only. For example, there are six possibilities to cut a tetrahedron at exactly one edge. In order to be able to subdivide tetrahedra disregarding to its bearing with equal subdivision procedures, transformation operations are utilized. Transformations are rotation or mirroring operations that map the particular topological cut case onto the orientation of the subdivision pattern for which the subdivision operations are implemented.

The second reason for the high complexity of progressive cutting is that there are various transition possibilities for a cut tetrahedron from one temporary subdivision to the next one until it reaches an end state and by that the final cut configuration.

2.5.2.2.2 State Machine Definition

The cutting state machine consists of:

- a finite, non empty set of states S,
- one initial state s_0, that is an element of S,
- I that holds a specific intersection occurrence, on edges or faces,
- δ a state transition function $\delta : S \times I \rightarrow S$, that maps states from one to another with the input of certain intersection occurrence, and
- F a set of final states referred to as end states, with $(F \subseteq S)$.

Moreover, there is a set of transformations T, each state transition is associated with one specific transformation.

2.5.2.2.3 States

Every tetrahedra of the model is assigned to a state in its state machine. A state defines the current condition of a tetrahedron with respect to its subdivision and its intersections at edges and faces. It pinpoints which subdivision pattern is currently applied to the tetrahedron. A state encodes the number and distribution of face and edge intersections.

One can distinguish between three types of states. Initial, intermediate and end states. The initial state depicts the situation where no intersections have taken place yet. All tetrahedra that are not effected by an incision yet, are in the initial state. A

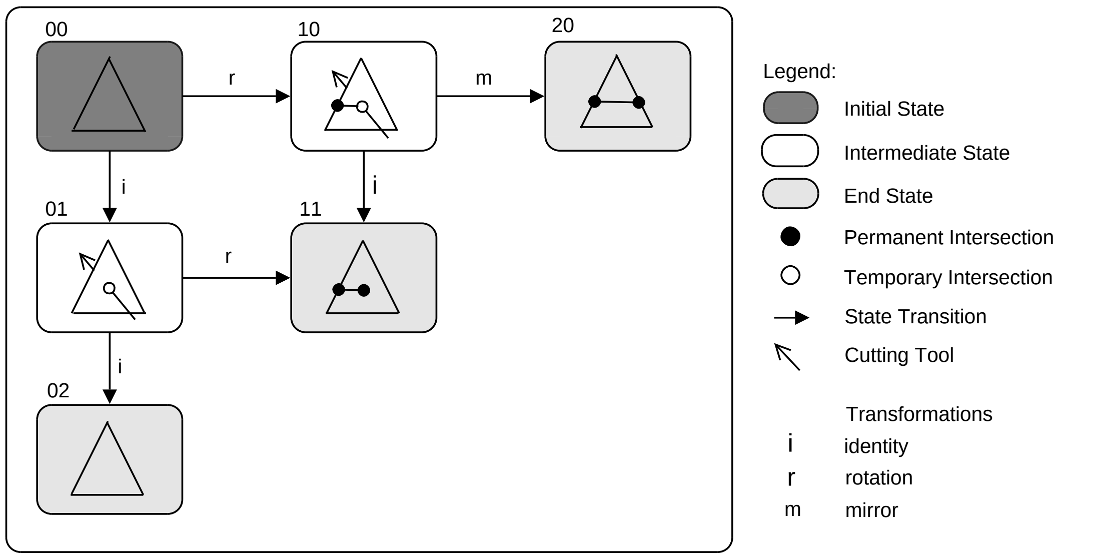


Figure 2.9: The 2D state machine example.

tetrahedron reaches its end state when the final subdivision pattern is determined, that is when the cutting tool leaves the tool and no further intersection will occur. Intermediate states are states where the tool is still in intersection with tetrahedron and the current subdivision is not entirely determined.

In case new intersections occur, the tetrahedron changes its state. Obviously, each tetrahedron stores its current state information.

2.5.2.2.4 State transitions

Transitions are changeovers from one state to another. They are associated with conditions that have to be fulfilled in order to enter the new state. These conditions are occurrences of edge and face intersections to a tetrahedron.

A state transition results in a specific operation set that replace the interior of the tetrahedra by a substructure. The operation set is a precise sequence of elementary cut operations (splitting edges and faces, inserting new nodes edges, etc.) that is unique for each different state transition. These elementary cut operations can be realized as LUT entries. State transitions only occur if there are new intersections to a tetrahedron. There are only specified transitions between states possible. A state transition table defines all possible state transitions.

An simple 2D example of a state machine for progressive cutting of triangular meshes is given in Figure 2.9 (for the 3D case see [4]). The state machine for elements of a triangular mesh depicts three topological different cases of a triangle for which subdivision operations will be applied: a triangle that has none (in state 00, 01 and 02), one (in state 10 and 11), and two edge intersections (in state 20). Initial, intermediate and end state can be identified by the different background shading. Every state has a possibly empty set of permanent and temporary intersections and is labeled uniquely. A state transition, illustrated as arrows between states, corresponds to the modification of the current temporary subdivision, if new intersection are found and in case the cutting tool leaves a triangle. If the tool sweeps out of the element, the triangle reaches an end state. The subdivision patterns of the three end states are possibly are possibly those depicted in Figure 2.4 on page 14. Next to the transition arrow the transformation operation is noted, which has to be considered in order to map the triangles orientation of the previous state to the to subdivision implementation of the coming state.

2.6 Cutting of Deformable Models

The class of anatomical structures that is cut in surgery usually consists of soft tissue. Soft tissue is elastic and consequently deformable. It deforms under a external force up to the point where its internal elastic forces counterbalance the external force. Thus, accurate modeling techniques for soft tissues that consider deformations are required. Those methods have to cope with topological modifications of the underlying model when cutting is performed.

2.6.1 Deformable Modeling

There are three different methods widely used for modeling soft tissue behaviour in surgery simulation applications, we want to give a rough overview about. These are Mass Spring Systems (MSS), the Finite Element Method (FEM) and the Chain-Mail Method. Broader surveys of deformable modeling techniques, and soft tissue modeling in particular, are given in [42][24][43].

2.6.1.1 Mass Spring System

In MSS the soft tissue is represented as a discrete model of mass points that are connected by springs of linear elasticity. Each mass point corresponds to a vertex in a mesh, has a position, velocity, mass value and a force applied to it. Springs link two mass points to each other and possess stiffness and damping constants. Deformation behaviour is simulated with the springs that propagate displacements between masses into the model. The motions of mass points, in response to applied forces, correspond to a system of ordinary differential equations (ODEs). Numerical time integration methods, like Euler Method and Second or Fourth Order Rung Kurt Method, are used solve these ODEs. The choice of the integration method has an impact on accuracy and stability of the deformation simulation.

MSS are computational efficient and useful for real time applications, but inaccurate especially for large area deformations. A MSS requires a balanced parametrization in order to ensure that the elastic behavior of the model approximates the physical properties of soft tissues. The interaction of coupled parameter, like stiffness, damping and duration of integration steps have an impact on accuracy, speed and stability of the simulation. Moreover, topological changes require updating those parameters, if for example new masses and springs are added to the model. MSS can only approximates the real physical behaviour of a continuous soft body. It does not consider other physical laws as mass and volume preservation. Solutions for this have to be added by other means.

Implementations of soft tissue simulations with surgical intervention and topological modification of mass spring systems have been realised in [16, 1, 2, 10, 26].

2.6.1.2 Finite Element Method

With the FEM method, the global deformation problem is treated in a continuous manner. The model consist a of finite set of geometrical elements that are connected by discrete nodes and do not overlap. Deformations and external forces correspond to a set of partial differential equations (PDEs).

FEM is physically the most sophisticated method and it is possible to achieve accurate and precise deformations. However, FEM is unsuited for an interactive surgery simulation that is required to run in real time and that has to undergo topological model modifications, due to high demands on computational resources. FEM is often applied to applications, where real-time computations are not of particular importance. For example, in the simulation of post operative outcomes as demonstrated by Koch et al. [44]. Other cutting approaches that use FEM can be found in [19, 14, 15, 23, 40, 36, 12].

2.6.1.3 Chain-Mail Method

While the two previous methods are physics-based, the Chain-Mail Method introduced by S.F. Gibson [45–47] is an example for methods that rely on kinematics. It does not consider physical properties like the effects of forces and models masses. Deformations are achieved by displacement of volumetric elements. An element is connected to its six spatially surrounding neighbours within a certain minimum and maximum distance. In case the movement of an element violates the distance thresholds of minimum or maximum distance to its neighbour the neighbour is moved as well. This way deformations are propagated though the model.

An advantage of this method, in comparison to surface or volume representation that assume a homogeneous inner structure, is that the volumetric representation can include more detailed anatomical information, like material types. Volumetric data generated by scanners can be used with its original resolution and do not need to be converted. Moreover, it is a fast and computational inexpensive method to implement deformation behaviour. However, the Chain-Mail Method can not incorporate physical properties of soft tissue and approximates soft tissue deformations only roughly.

2.6.2 Topological Model Modifications

Interactive cutting deals with the topological modification of a model's mesh representation. These changes of the underlying representation of the soft tissue model take place during deformation simulation and they affect the behavior of the simulation.

Most incision modeling methods discussed in Section 2.4 on page 10 need to perform element deletion, as in the cauterization method, or insertion of new elements, as in all decomposing methods. This leads to a increase or decrease of the overall number of elements in the model. In addition, detaching elements from each other requires the insertion of new mass points. Splitting edges into two, as it is carried out in subdivision methods, oblige to insert both, new springs and new mass points.

2.6.2.1 Modification Impact to a Mass Spring System

Since cutting alters the underlying topology of the deformable model it directly affects accuracy, stability, realism and performance of the deformation simulation. Variations of the element count in the model effects the the simulation speed. The insertion of new mass alters the overall mass of the model, if the points with equal masses are inserted. For this reason masses of present points have to be adapted with a mass redistribution algorithm [10]. When an edge is cut, a spring has to be replaced by two shorter ones. Rest length and stiffness constants have to be adjusted accordingly. What is more, springs are added to keep a geometrical consistent mesh structure. When tetrahedral meshes undergo subdivision only tetrahedral elements should be add to the structure. This way the connectivity of the mesh is altered and the number of springs connected to one mass point in the mesh is increased. A larger amount of springs to a mass point that apply forces to it, increases the stiffness in the mesh. This also influences the model's stability.

2.7 Improvement Directions

In the last section the impact of cutting meshes and deformable models in particular was discussed. Modification of the mesh topology via cutting poses critical problems to the simulation of soft tissue models. In this section we give an overview of attempts to overcome certain obstacles, some of which are in direct correlation with the problems named in the last section. There are four categories pursued that address the simulation speed, simulation stability, mesh quality and accuracy of the cut representation. Improvements are achieved by

1. reduction of new element creations
2. constrains on edge aspect ratios of elements
3. avoiding a disconnected mesh structure after a cut
4. an accurate cut path representation

Since the simulation speed of the physical model is slowed down by an unnecessary number of elements, counter action were taken by minimal element subdivision methods or methods that align the cut path with existing edges or faces. The advancement to put constraints on aspect ratios arises due to the occurrence of degeneracies. Degeneracies are elements that have edges with very unbalanced aspect ratios or are flattened in shape. Degenerated elements are produced via subdivision when a cut occurs near a mass vertices or edges. Since degeneracies lead to severe stability problems of the simulation, counter actions in form of degeneracy removal or snapping were developed. The work of Ganovelli et al. [16] goes further into the removal of degenerated elements. They remove degeneracies by collapsing elements with short edges. This method is not capable of removing all degeneracies from a mesh and does not consider the principle of mass and volume conservation, since parts of the mesh are removed. An alternative measurement to deal with degeneracies is snapping, which is explained in the next section.

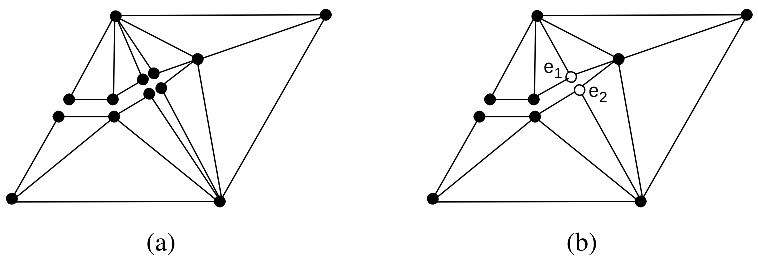


Figure 2.10: The cut adjacency of two faces: (a) result of a cut edge without taking the connectivity of adjacent edges into account, and (b) illustrates a cut edge with correct mesh connectivity.

Methods that do not consider the connectivity of adjacent elements when subdivision is used, as in the Tetrahedral Presplit Method in section Section 2.4.2.1 on page 12, lead to failures in the physical representation and cause an unrealistic simulation. When an edge is cut it has to be split into two edges and two new mass nodes are inserted into the mesh. If every edge is treated separately a undesired situation as displayed in Figure 2.10(a) would occur, where adjacent elements lose their connectivity. Adjacent tetrahedra are those that share either an edge or a face. For all n tetrahedra that shares the cut edge $2 \cdot n$ new, unconnected mass points would be added two the mesh. This would lead to serious problems of the physical representation of the deformable model. If the model is under stress during the simulation then gaps as in Figure 2.10(a) can be seen at places where edges have been cut. Most methods developed after [1] consider a proper mesh connectivity after topological modifications.

The symmetric subdivision method of Bielser et al. [2] is the implementation of a closer approximation of the user defined cut path.

2.7.1 Snapping

While the user defines a cutting path freely by the tool's motion it might occur that small and degenerated tetrahedra emerge. When the tool intersects an edge near one of its vertices the edge is split into two edges with an unequal aspect ratio. The subdivision tetrahedra, of the tetrahedron the edge belongs to, are consequently unequal in size and shape. Degenerated tetrahedra are also generated due to the temporary subdivision used by progressive cutting methods. Examples for both degeneration cases are illustrated in Figure 2.11 on the next page. If the first temporary subdivision is applied when the cutting tool, that is currently in intersection with the tetrahedron, is located very close to one of the tetrahedron's edges then two flattened, degenerated

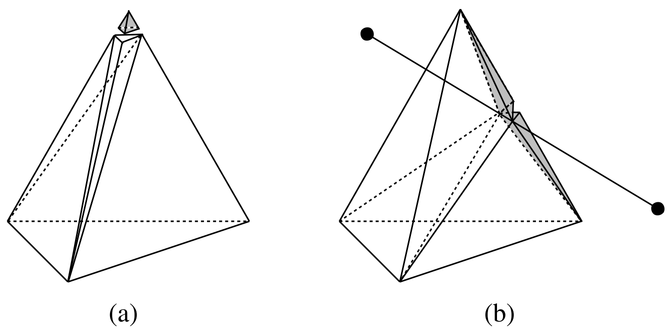


Figure 2.11: Examples for degenerated tetrahedra: (a) degeneration due to intersection close to a node, (b) degeneration due to temporary subdivision.

tetrahedra are generated. However, the simulation of the deformable model works best on a homogeneous meshes. The problem with degenerated tetrahedra is that they have a severe impact on the stability of simulation of the deformable model. The deformation simulation of a mass spring system copes best with a homogeneous meshed model. Degeneracies result in numerical instability of the deformation simulation. Thus, it is essential that degenerated elements are avoided. A method which accomplishes this is called node snapping and has been realized by [38][8][13][39].

Snapping comes with additional tests and iteration through subdivision templates, but it improves mesh quality and simulation stability.

2.7.1.1 Definition

Snapping is an advancement for subdivision methods. Appearances of degeneracies as a result of cutting close to mesh vertices or edges in a mesh is prevented on the fly. Instead of the conventional subdivision patterns compatible to a cut topology, that would generate a degenerated element an alternative pattern is applied. Vertices close to the cut path are mapped onto the path. A simplified pattern that subdivides an element into less sub-elements and produces more homogeneous subdivision can be applied.

2.7.1.2 Concept

For node snapping it is required to identify vertices close to the cut tool trajectory that have to be snapped onto the cut path. For intersected elements the absolute distances of all vertices to the cut path are determined. Points that go below a certain threshold distance d are marked as *cut points* (see Figure 2.12 on the following page(a)). Then a subdivision is conducted with the alternative patterns that take number and dis-

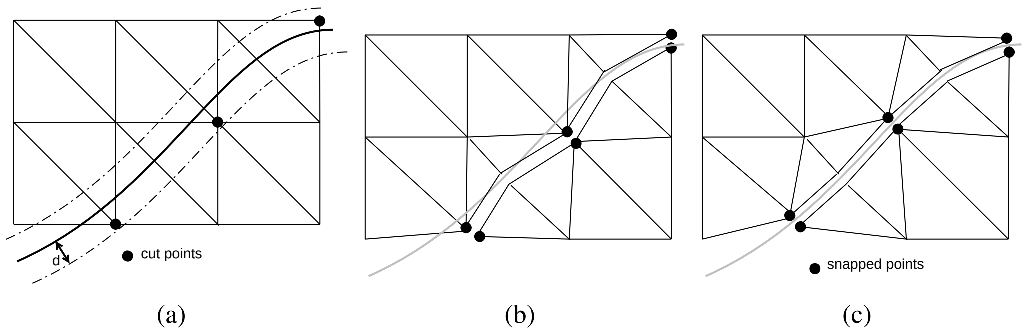


Figure 2.12: The snapping process in a 2D example: (a) determination of the cut points, (b) alternative subdivision, and (c) cut points are snapped onto the cut path.

tribution of the cut points int account. All cut point are duplicated (Figure 2.12(b)). Hereafter, the cut points are snapped to both sides of the the cut path, by orthogonal projection the vertices into the cut path as illustrated in Figure 2.12(c).

This procedure comes with a complication for elements, which lie at the object's boundary. If vertices lying at the objects surface would be snapped onto the cut path, the object would alter its original outer shape. This can be observed in Figure 2.12(b). The rectangular shape becomes jagged. Consequently, elements at the objects surface require special treatment. The problem can be solved by simply not applying snapping to the surface elements. Nienhuys et al. [13] solve the matter by projecting a surface vertex v orthogonally onto the intersections between cut path and all faces that share v. The point projected closest to the to the original vertex v is chosen as snapping position for v.
Snapping was implemented in various forms. For example, the work of Steinemann et al. [8] restricts snapping to those cases were the tetrahedron is dissected into two parts. They apply snapping with non-progressive cutting. Mor et al. [39] on the other hand apply snapping according to stability criteria also when degeneracies due to temporary subdivision occur.

Chapter 3

Our Approach

This chapter deals with the methods we used in order to realize a cutting framework for our surgery simulation system as it has been implemented within the scope of this thesis. Three different cutting modules are introduced that can be used interchangeably within the surgery simulation system. Firstly, algorithm requirements and an overview of the system's architecture are described. Then the concepts used in all three algorithms are introduced: the implementation of an interactive cutting tool, intersection calculations, and elementary subdivision procedures, like edge and face splitting. Thereafter, the algorithms are presented. The first two algorithms are solely progressive subdivision methods within a state machine framework. The first introduces an improved state machine approach. It deploys symmetric subdivision patterns and uses special shortcut subdivision patterns. The second cutting algorithm uses minimal elements subdivision patterns. With the third algorithm we introduce a novel hybrid cutting method that combines progressive and non-progressive subdivision and applies the different subdivision methods to the model's elements dependent on their location.

3.1 Algorithm Requirements

This sections explains requirements and preconditions we have to consider for our algorithms and which provide reasons for choosing specific methods and solutions. The algorithms presented in this thesis have been developed in scope of a surgery simulation project of larger scale. Thus there is a closed set of predefined conditions that has to be met by the cutting algorithms.

Surgery simulators can either be used for training of surgical procedures or for modeling post operative outcomes. Both application areas have different demands on a simulation framework. While for post operative planning interactivity and computational efficiency is not a fundamental requirement, the precision and the accuracy of the result is paramount. However, for a surgical training simulation, the main needs are a real-time application and a high degree of interactivity. Although high simulation realism is preferable, the priority of the accuracy is reduced for the benefit of a realistic feedback to the trainee.

Our project addresses the training aspect for exercising surgery on organs of voluminous soft tissue. Thus, the application has to function in real time, for the sake of interactivity and simulation realism. Consequently, our cutting procedures are conceptually designed and tested for tetrahedral meshed volumetric soft tissue models, with a mass spring system for a fast deformable soft tissue simulation.

The topological modification inflicted upon the model by the cutting method must leave the mass-spring model consistent and as stable as possible. Since the increase of the number of tetrahedra slows down the simulation, the number of elements introduced by the model should be kept as small as possible. Furthermore, the user should be able to define a cut path interactively with a certain degree of accuracy and also get an accurate feedback at the same time. Consequently, subdivision methods with progressive subdivision are a good choice. Non-decomposing cutting methods or non-progressive cutting would cause major visual disturbances.

3.2 Architecture

In the following, we give a rough overview of the cutting components and how they interface with the surgery simulation system. A main architectural overview of our surgery simulator is presented in [48]. The simulation framework provides for meshing, mass spring simulation, simulation core [48], collision detection and collision response [49].

Figure 3.1 on the following page depicts the main components of a cutting module, which are the cutting tool, intersection detection, states tracker, virtual tetrahedra representation, and the subdivider. A cutting module interfaces with the simulation core of the surgery simulator. The simulation core provides a cutting module with the geometrical representation of the model. The intersection detection module calculates the intersections between elements of the mesh geometry and the cut path (see Section 3.3.2 on page 32). For each element in the model that is tested for intersections a virtual tetrahedron is created. It stores all the supplementary information needed for the subdivision, like its state machine information and intersection data. The states tracker manages the set of these virtual tetrahedra and tracks the state and transformation data. The subdivider conducts the actual modification of the geometrical model representa-

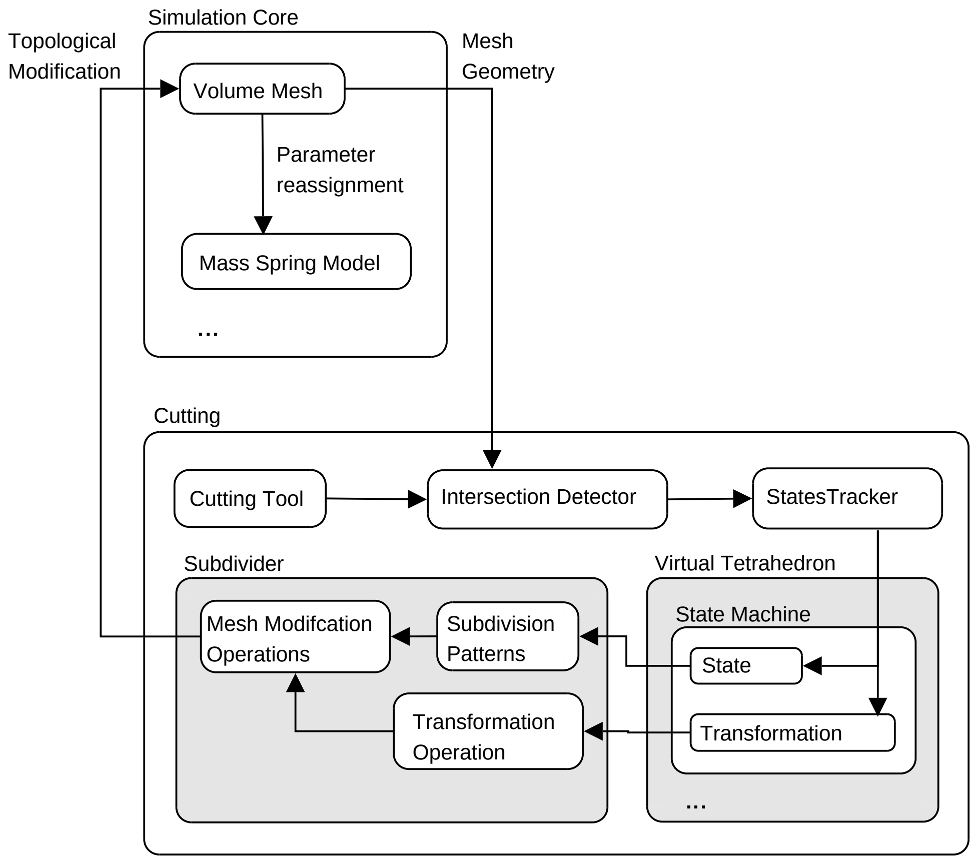


Figure 3.1: The main components of the cutting algorithm

tion on the basis of the data in the state machine for each virtual tetrahedron. Moreover, data that is necessary for the parameter reassignment of the mass spring model is delegated into the simulation core.

3.2.1 Cutting Procedure

Now we give a general overview of the main steps of the cutting procedure. It consist of the following steps:

- First the user freely defines a cut path, by moving the interactive *cutting tool* arbitrarily through 3D space. The swept surface is approximated by a triangle strip.
- A subset of the model's elements are determined via collision detection, which have been possibly intersected. This set is called *collision tetrahedra* in the following.

- Thereafter an *intersection detection* is performed between the set of *collision tetrahedra* and the cut path.
- Calculated intersections are stored for each tetrahedron. Edges and faces that are intersected are encoded and coordinates are saved as well.
- In the following, tetrahedra possessing an intersection with the cut tool are stored into another set of tetrahedra referred to as *penetrated tetrahedra.*
- Then the elements of the set of *penetrated tetra* are subdivided into smaller elements, in that way that the incision defined by the cut path is geometrically modeled. Edges and faces are split. New nodes and tetrahedra are inserted to the mesh.
- The topological modification of the underlying mass spring model requires to update the model. New nodes are assigned a mass. New springs are are inserted and parameters such as stiffness and rest length have to be adjusted.

3.3 Cutting Tool

This section describes our implementation of the interactive cutting tool. The cutting tool serves the user as virtual surgical tool. The user can define the location and direction of a cut with this tool over time.

3.3.1 Tool Representation

We chose to implement a virtual cutting tool as described in Section 2.3.3 on page 9, because it provides the user with the most interactive freedom.

For simplicity reasons, the cutting tool is defined by a line segment, which functions as the cutting edge. It is represented by two points; head and tail. The segment can be freely moved through 3D space.

3.3.2 Cut Path

By moving the cutting tool in 3D space over a certain time interval a manifold emerges that has to be approximated for fast intersection computations against elements of the model. This approximation is referred to as cut path.

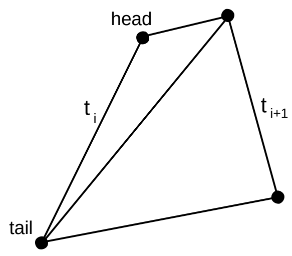


Figure 3.2: The cut path over one time interval $[t_i, t_{i+1}]$ spanned by the cut tool.

The cut path $P(t_i, t_{i+n})$ is a surface, swept by the cutting tools movement over a time interval $[t_i, t_{i+n}]; i = 0, ..., n$, and $n \in \mathbb{N}$. Where t_i are discrete time steps and the cutting tool's *head* and *tail* coordinates are known for each time step. This set of triples of coordinates and related time steps altogether define the cut path:

$$P(t_i, t_{i+n}) = \{(head, tail, t_j) | j \in [i, i+n]\} \tag{3.1}$$

The cut path is approximated by a strip of triangles. Therefore, a complete cut path consists of $2 * n$ triangles. The motivation for this approximation is that it is less complex to calculate intersections with plain primitives like triangles than through complex curves. Since the time steps are usually very small, the triangle strip becomes has a sufficient high spatial resolution such that it approximates the user defined cut path very well.

The elementary part of the cut path defines the interval $[t_i, t_{i+1}]$ of two particular points in time that directly follow one another. The cut path over one time step is decomposed into two triangles (see Figure 3.2).

On the data structure level, our algorithm stores the tool's head and tail coordinates at time steps t_i and t_{i+1} for intersection calculations. For testing purposes however, we arranged to read all coordinates for an entire cut. This way we are able to reproduce cuts by reading the data from a script.

3.4 Intersections

All tetrahedra that are possibly affected by the cutting tool have to be tested against intersections with the cut path. This requires iterating through all tetrahedra that possibly lie in the cut path. In case of progressive cutting, we deal with two types of

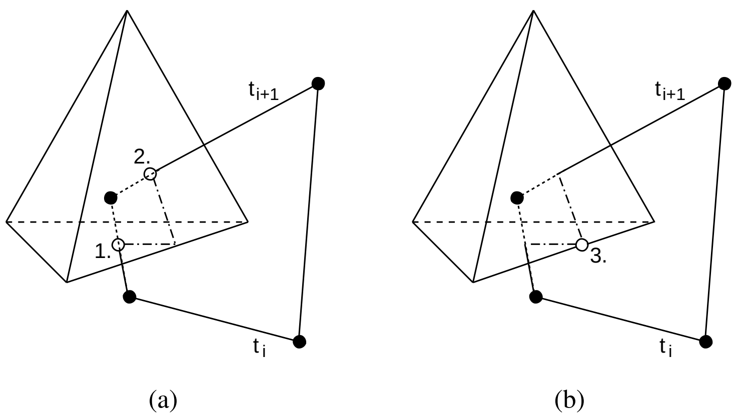


Figure 3.3: The different intersection types: (a) 1. shows a permanent face intersection and 2. a temporary face intersection, and (b) illustrates a permanent edge intersection.

intersections: Temporary face and edge intersections and those face intersections that are permanent. Both types are visualized in Figure 3.3.

3.4.1 Intersection Detection

The passage of the tool through the model can be tracked with the cut path. In order to determine if an element of the model, a tetrahedron, was cut by the tool, it has to be tested for intersections with the cut path in every time step.

We want to point out that intersections tests are relatively costly in computational resources. Obviously, testing every tetrahedron of the mesh in every simulation step on intersections would be far too costly. Therefore, we prefer to check against a small subset of the mesh that contains those tetrahedra that are likely to collide with the cutting tool. Tetrahedra which might be considered for intersection testing are predetermined in every time step via a rough collision detection between the model and the cutting tool. That way only a small subset of the model elements for which it is likely to be intersected have to be tested.

For all elements in this subset every of its 6 edges and 4 faces are tested for intersections against the cut path. Since the cut path is modeled as a strip of triangles and the tetrahedra have to be tested for intersection of edges (segments) and faces (triangles) it is sufficient to use segment-triangle intersection calculation only:

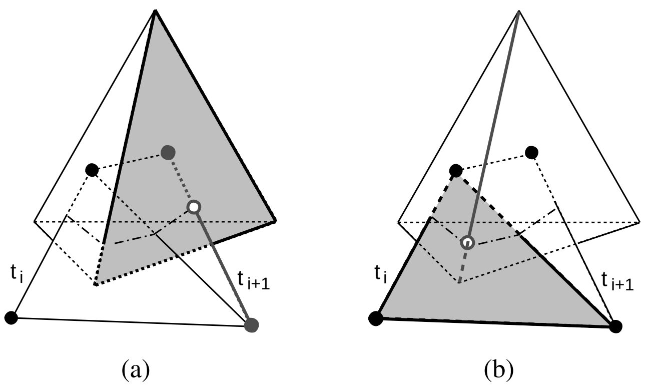


Figure 3.4: The specific segment-triangle intersection cases in our algorithms: (a) the cutting tool intersects a tetrahedral face, and (b) a triangle of the cut path, that is represented as a triangle strip, intersects a tetrahedral edge.

- Edge - Cut path: every edge is tested with triangle-segment intersection against the two triangles of the cut path in the current time interval.
- Face - Cut path: The face of an tetrahedra is an triangle. Every face is tested against every segment that spans the two triangles of the cut path.

The specific intersection testing cases are illustrated in Figure 3.4. The algorithm stores intersection information for all penetrated tetrahedra, like intersection coordinates, edge and face IDs, for further processing. There are more than three thousand possibilities to cut the edges and faces dependent on the point ID permutations and succession in which edges and faces are cut [4]. A formal description of the segment-triangle intersection follows.

3.4.2 Segment-Triangle Intersection

For a segment S that is bounded by two points $\vec{p}_0$ and $\vec{p}_1$ we define a corresponding line by its parametric equation:

$$L : \vec{x} = \vec{p}_0 + s(\vec{p}_1 - \vec{p}_0), s \in \mathbb{R} \tag{3.2}$$

For a triangle T with three vertices $\vec{v}_0$, $\vec{v}_1$, and $\vec{v}_2$ a plane P with $\vec{x}$ and the normal $\vec{n}$ of the plane

$$\vec{n} = (\vec{v}_1 - \vec{v}_0) \times (\vec{v}_2 - \vec{v}_0) \tag{3.3}$$

can be defined that includes the triangle T. For the determination if the segment S intersects the triangle T we first check for intersection of L and P. In case there is no intersection between line and plane there is consequently no intersection between the segment and the triangle.

The determination if there is an intersection at all can be easily done by checking if

$$\vec{n} \cdot \vec{d} = 0 \tag{3.4}$$

where $\vec{n}$ is the normal of the plane, $\vec{d} = \vec{p}_1 - \vec{p}_0$ is the direction vector of the line, and $\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i$ defines the dot product of two vectors $\vec{a}, \vec{b} \in \mathbb{R}^n$. That means if the dot product between $\vec{d}$ and $\vec{n}$ becomes 0, line and plane are located parallel to each other. Thus, either there is no intersection or the line totally lies in the plane. If L is contained in P this can be proven by inserting an arbitrarily chosen point of L into the equation

$$\vec{n} \cdot (\vec{p} - \vec{v}_0) = 0 \tag{3.5}$$

In case an intersection between L and P is present the intersection point $\vec{x}_0$ is calculated and it has to be investigated if the x_0 lies on P within the boundaries of the triangle T and on L within the section of the segment S.

The intersection point $\vec{x}_0$ between L and P is calculated by

$$\vec{x}_0 = \vec{p}_0 + s_0(\vec{p}_1 - \vec{p}_0) \tag{3.6}$$

We obtain s_0 by substituting the intersection $\vec{x}_0$ of the plane representation $P : \vec{n} \cdot \vec{x}_0 = d$ into the parametric equation of the line $L : \vec{x}_0 = \vec{p}_0 + s_0(\vec{p}_1 \cdot \vec{p}_0)$. Solving for s_0 we get

$$s_0 = \frac{d - \vec{p}_0 \cdot \vec{n}}{(\vec{p}_1 - \vec{p}_0) \cdot \vec{n}} \tag{3.7}$$

If the segment is defined from $\vec{p}_0$ to $\vec{p}_1$, then the intersection x_0 lies on the segment S if

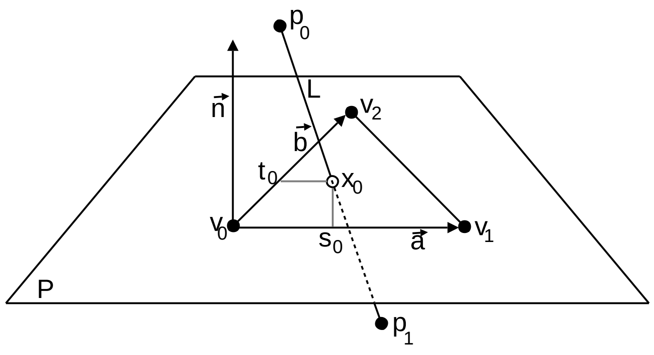


Figure 3.5: The triangle-segment intersection calculation.

$0 \leq s_0 \leq 1$ is true. The intersection point $\vec{x}_0$ then is calculated by solving:

$$\vec{x}_0 = \vec{p_0} + s_0(\vec{p_1} \cdot \vec{p_0}) \tag{3.8}$$

3.4.2.0.1 Triangle - Point Inclusion

Now it is still open if the intersection point x_0 is included within the bounds of the triangle.For inclusion calculations we use the parametric equation of the plane P with the corresponding vertices of T:

$$P : \vec{x} = \vec{v}_0 + s\vec{a} + t\vec{b} \tag{3.9}$$

with $\vec{a} = (\vec{v}_1 - \vec{v}_0)$, $\vec{b} = (\vec{v}_2 - \vec{v}_0)$ and $s, t \in \mathbb{R}$. $\vec{a}$ and $\vec{b}$ represent the edge vectors of T. The criteria for point inclusion are that $s \geq 0$, $t \geq 0$ and $s+t \leq 1$ are true. Furthermore, for $s = t = 0$ the point is identical to triangle vertex $\vec{v}_0$, if $s = 1$ and $t = 0$ then $\vec{x}_0 = \vec{v}_1$ applies. In case $s = 0$ and $t = 1$ the intersection point $\vec{x}_0$ corresponds to the vertex $\vec{v}_2$. The geometrical context can be observed in Figure 3.5.

Accordingly, s_0 and t_0 have to be found for the intersection x_0 and the inequations have to be verified. The "perp-dot" product by Hill[50] is used to solve for s_0 and t_0:

In order to solve for t_0 a perpendicular vector $\vec{a}^{\perp}$ to $\vec{a}$ is defined. $\vec{a}^{\perp}$ and $\vec{a}$ form a right angle and consequently $\vec{a} \cdot \vec{a}^{\perp} = 0$ applies. To ensure that the vector $\vec{a}^{\perp}$ lies in the plane P, $\vec{a}^{\perp}$ is obtained by $\vec{a}^{\perp} = \vec{n} \times \vec{a}$. The parametric equation of $P : \vec{x}_0 = \vec{v}_0 + s_0\vec{a} + t_0\vec{b}$ is then multiplied on both sides by $\vec{a}^{\perp}$ and solved for $\vec{x}_0$:

$$\vec{x}_0\vec{a}^{\perp} = \vec{v}_0\vec{a}^{\perp} + s_0\vec{a}\vec{a}^{\perp} + t_0\vec{b}\vec{a}^{\perp}$$

$$\vec{x}_0\vec{a}^{\perp} = \vec{v}_0\vec{a}^{\perp} + t_0\vec{b}\vec{a}^{\perp}, \vec{a} \cdot \vec{a}^{\perp} = 0$$

$$t_0 = \frac{(\vec{x} - \vec{v_0}) \cdot \vec{a}^{\perp}}{\vec{b} \cdot \vec{a}^{\perp}} = \frac{(\vec{x} - \vec{v_0}) \cdot (\vec{n} \times \vec{a})}{\vec{b} \cdot (\vec{n} \times \vec{a})} \tag{3.10}$$

In the same way for s_0 the perpendicular vector $\vec{b}^{\perp} = \vec{n} \cdot \vec{b}$, with $\vec{b} \cdot \vec{b}^{\perp} = 0$ it is possible to determine s_0:

$$s_0 = \frac{(\vec{x} - \vec{v_0}) \cdot \vec{b}^{\perp}}{\vec{a} \cdot \vec{b}^{\perp}} = \frac{(\vec{x} - \vec{v_0}) \cdot (\vec{n} \times \vec{b})}{\vec{a} \cdot (\vec{n} \times \vec{b})} \tag{3.11}$$

3.5 General Subdivision Operations

All three methods presented in this chapter have in common, that they use subdivision in order to model incisions along a defined cut path. The elementary operations to realize subdivision however, differ from method to method. Yet most operations follow the same simple principles. Cutting and especially the part of incision modeling can be seen as a geometrical problem. Thus, some linear algebra concepts are used in the following.

3.5.1 Edge Splitting

The procedure of splitting edges is the same in all methods in this chapter. If an edge intersection is determined, the edge E has to be split into two edges E_1 and E_2. Two new mass points are introduced near the intersection point. An edge $E(\vec{p_0}, \vec{p_1})$ is defined by two vertex points $\vec{p_0}$ and $\vec{p_1}$, the intersection point $\vec{p_{int}}$ lies on the segment between $\vec{p_0}$ and $\vec{p_1}$. The new edges are $E_1(\vec{p_0}, \vec{p_{int0}})$ and $E_2(\vec{p_{int1}}, \vec{p_1})$. A distance between $\vec{p_{int0}}$ and $\vec{p_{int1}}$ is required to model the incision gap. This distance is very small and corresponds to the thickness of the cutting tool. The two new point coordinates are obtained by linear interpolation:

First we calculate the direction vector $\vec{d} = (\vec{p_1} - \vec{p_0})$ directed from $\vec{p_0}$ to $\vec{p_1}$ and calculate the norm $\vec{dn}$ of $\vec{d}$:

$$\vec{dn} = \frac{(\vec{p_1} - \vec{p_0})}{||(\vec{p_1} - \vec{p_0})||} \tag{3.12}$$

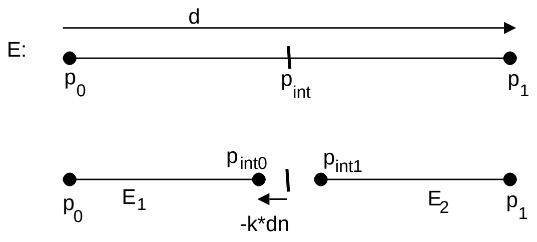


Figure 3.6: The procedure of splitting an edge: the coordinates of the two new points are determined by linear interpolation.

The new edge points $\vec{p_{int0}}$ of E_1 and $\vec{p_{int1}}$ of E_2 can be determined by

$$\vec{p_{int0}} = \vec{p_{int}} - k \cdot \vec{dn} \tag{3.13}$$

$$\vec{p_{int1}} = \vec{p_{int}} + k \cdot \vec{dn} \tag{3.14}$$

where k is a constant that influences the cut's gap width (see Figure 3.6).

3.5.2 Face Splitting

While the process of edge splitting always remains the same, the way we divide faces differs from algorithm to algorithm. Face splitting, in contrast to splitting edges, is implemented differently for the methods of this chapter. Faces which are split symmetrically organize the insertion of face midpoints. Minimal face triangulation use points that emerge due to edge intersections only. This reduces splitting faces to splitting the face's two intersected edges and assigning the sub-tetrahedral vertices properly. The face splitting techniques will be explained more in detail in the sections belonging to the different methods below.

3.6 State Machine Approach with Improved Shortcut Subdivision

The first algorithm implemented by the author of this thesis is an interactive progressive cutting method with subdivision by means of a combination of symmetric and minimal elements subdivision patterns. Transitions between temporary subdivisions for tetrahedra to be subdivided are managed via a state machine framework. Our algorithm differs from the original state machine framework by Bielser et. al [4]. In contrast to the original state machine approach, which keeps tetrahedra removal operation minimal, our algorithm is realized entirely without removal operations. A system for updating the tetrahedral substructure is used instead. Additionally, we extend their progressive tetrahedral mesh cutting technique to deformable models in context of virtual surgery. Furthermore, the algorithm is integrated into a surgery simulation system.

With the objective of keeping the mesh complexity low, minimal elements subdivision patterns are deployed wherever the application of temporary subdivision sequences is not necessary. This is achieved by additional shortcut procedures, which supplement the state machine framework. Moreover, we define a graph representation for possible state transitions of the state machines and optimize the state machine framework by precompiling possible transition sequences before the simulation starts. These transition sequences are stored within a matrix and can be easily accessed at run-time.

A conceptual overview of our state machine version, analog to the example in Figure 2.9 on page 21, is given in 3.7. For the fundamental concept of the state machine approach please refer to the previous chapter: Section 2.5.2.2 on page 19. In the following we present the main aspects of our implementation and dive into the issues stated above.

3.6.1 States

We assign an own state machine instance and a current state to each tetrahedron that is possibly in intersection with the cut path.

3.6.1.1 Representation

A state s is defined by the number and intersection occurrences at edges and faces of a tetrahedron. Both face and edge intersection occurrences are stored as 4-digit and 6-digit bit-codes respectively. Basically, each digit encodes for each face and edge whether a tetrahedron is in intersection with the cut path. As a nice side effect, the digits can be used as an edge and face index. Moreover, the coordinates for the actual intersections are stored as attributes of the state too. Fortunately, there is a finite set of states S for a state machine. The set of states correspond to the boxes in the conceptual view of the state machine in Figure 3.7 on the following page. Initial and end states are have a colored background.

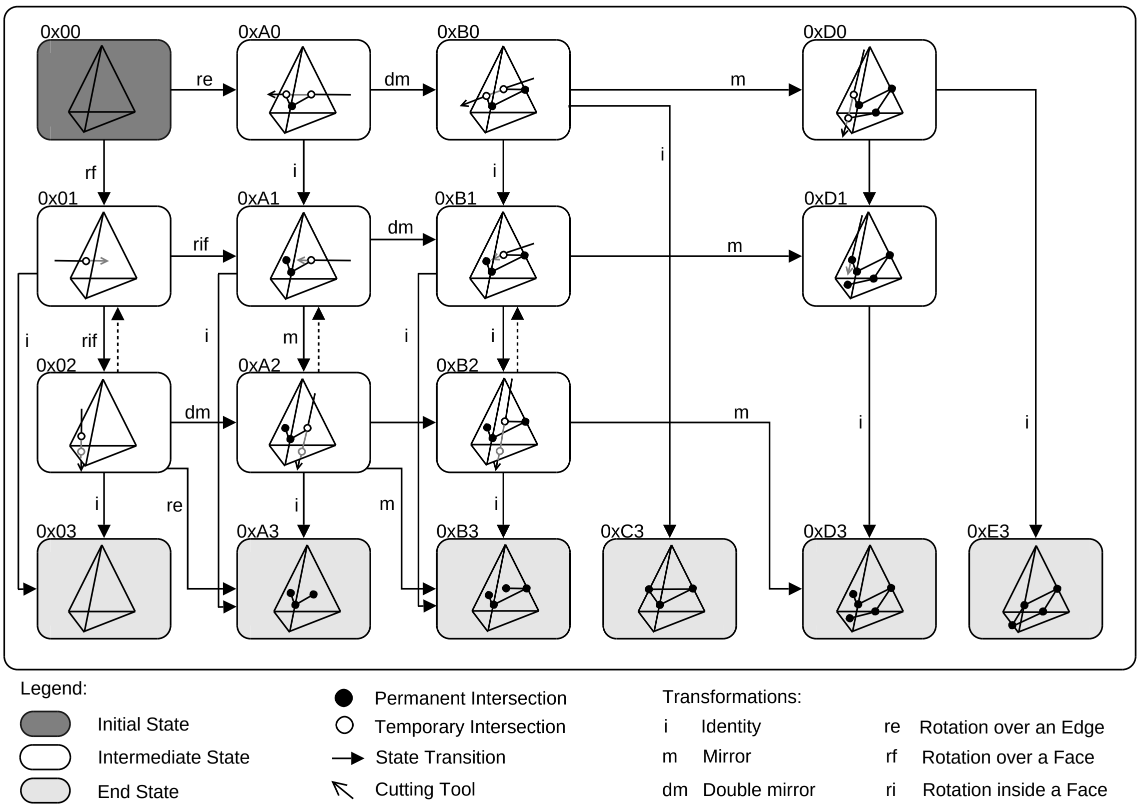


Figure 3.7: The conceptual overview of our state machine implementation.

3.6.1.2 State Determination

The storage of the digit codes for face and edge intersections has the purpose of an efficient state determination. Within every time step the properties of a state are checked for alterations. This is done by checking the digit codes. In case there is a change of the state at hand, a state transition with the appropriate set of subdivision operation has to be applied to the tetrahedron.

Since the means for identifying states are only given at the present time step and due to the fact that a state transition can only be determined if two states are known (start state and goal state), it is necessary that each tetrahedron stores its previous state as well. For this purpose we avail ourselves of the hexadecimal labeling scheme proposed by Bielser et al. Label 0x00 represents the initial state, labels of the form 0x0_, 0xA_, 0xB_, 0xC_, 0xD_, and 0xE_ depict states according to the topological different intersection distribution. 0xA3, 0xB3, 0xC3, 0xD3, and 0xE3 are end states and correspond to the five cut cases displayed in Figure 2.3 on page 13.

3.6.2 State Transitions without Removal Operations

A state transition t results in a set of subdivision operations that alter the substructure of a tetrahedron. For instance, if a tetrahedron passes from state 0xA0 to 0xB0 (see Figure 3.7 on the preceding page) the sub-tetrahedra count increases from six to ten elements. Only the two sub-tetrahedra sharing the two split edges of the 0xA0 tetrahedra can be kept as they have been inserted. The remaining four have to be substituted by eight tetrahedra with a different vertex configuration.

With our algorithm, the above example is realized without deletion of the four remaining sub-tetrahedra, but by updating the sub-tetrahedra vertex IDs. The other four tetrahedra needed are simply added to realize the state 0xB0. Updating is sufficient on account of a property of the symmetric subdivision patterns. With each temporary subdivision, no matter in which order intersections are found, the number of sub-tetrahedra is always increased for a parent tetrahedron, that is currently under subdivision. It is never required to delete an element from the initial state to an end state.

3.6.3 Transformations

Subdivision procedures are not realised for all symmetric and rotational different cases of a tetrahedron, because there are far too many. For the case that a single edge of a tetrahedron is cut, there are six possible edges that could be cut. However, a set of subdivision operations is only implemented for a single case for one specific edge, although there are six cut permutations. The purpose of transformations is to exploit the symmetry properties of a pattern. Transformations allow to use one generic procedure for all six permutations by mapping the available implementation to the actual present topological cut case. This is achieved by applying a series of transformation operations that consist of rotation and mirroring. The single implemented case is called pattern. The advantage of this procedure is that the number of operation sets is kept small for all state transitions in the state machine. The approach of using transformations has been used by several researchers before. Our implementation follows for the most part the procedure of [4].

Beside the information of the state we store for every tetrahedron Δ_i in the set of penetrated tetrahedra the current transformation information. There are five types of transformations: mirror, double mirror, rotate inside a face, rotate over an edge, and rotate over a face. Some transformation types can also be achieved by combination of others. For instance double mirror is nothing less than successively applying two mirror operations to an element. The advantage of the five transformation types is that it is possible to link exactly one transformation type to one specific state transition. This becomes important when applying transition sequences, as explained in the next section.

An important issue is that transformations do not occur in a geometrical sense, but on basis of local vertex IDs. The vertex identifications (v_0, v_1, v_2, v_3) of Δ_i will only be transformed and switched accordingly. Thus, a transformation can be interpreted as a mapping function $M : (v_0, v_1, v_2, v_3) \rightarrow (v'_0, v'_1, v'_2, v'_3)$, that maps the current vertex ID distribution to a new permutation of vertex IDs. The actual shape of Δ_i will not be mirrored or rotated. Every transformation is defined by the ID of a source primitive to the ID of a destination primitive. Primitives are either edges or faces. The source primitive is the primitive cut in the generic pattern. The destination primitive is the one intersected in the real case.

3.6.4 Transition Sequences

It is possible that a tetrahedron passes from a state to another state although both states are not directly connected by a single transition. For example, in case the user moves the cutting tool quickly, it is possible that a tetrahedron is cut at two edges within the same time interval. In this case, where the state machine does not provide for a direct state transition from the previous state to the current state a sequence of state transitions has to be applied to the element. A state transition sequence involves the successive appliance of state transitions to immediate state successors that connect the previous and current state and its transformations.

3.6.5 Transformation Sequences

If a sequence of state transition is applied to a tetrahedron, then the transformation of the element are build up incrementally. It is irrelevant if the transformation sequence has to be applied within one time interval or during more than one time intervals. The transformation between two intermediate states is only correct if it is applied to the absolute transformation that combines all incremental transformations made in state transitions beforehand. Therefore every tetrahedron has to store its absolute transformation in correspondence to vertex ID distribution of the initial state.

3.6.6 State Transition Graph

We improve the state machine approach of Bielser et. al [4] by optimizing the search for state transition sequence for two given states.

3.6.6.1 Finding the Optimal Path and Precalculation

When transition sequences are processed, the information of the start state at time t_i and the goal state at time t_{i+1} are available only. In order to carry out the subdivision correctly it is necessary to find the intermediate states between start and end state because the transitions between all states yield the required operation sets for the proper subdivision sequence.

The question is how to determine intermediate states of state transition sequences. Or in other words, we seek the shortest path between start and goal state. The state machine is easily transformed into a unweighted directed graph. We define a directed state machine graph Γ as a pair of states S and transitions T:

$$\Gamma := (S, T) \tag{3.15}$$

where transitions are defined as a set of ordered pairs of states

$$T = \{(x, y)|x, y \in S \wedge x \neq y\} \tag{3.16}$$

Finding the shortest path in a directed graph is solved by breadth-first search (BFS). BFS builds up a predecessor array, that holds the shortest path for a given state to every state in the graph. Then for the shortest path $(s_0, s_1, ..., s_n); s_i \in S$ the operations associated with each transition (s_i, s_{i+1}) can be applied successively, in order to reach the subdivision of the goal state.

It would be fairly ineffective to determine the shortest path of transitions every time it is requested. Since the state machine graph does not change during run-time, it is worthwhile to precalculate all possible shortest paths for the set $\{(x, y)|x, y \in S\}$ and store it in a predecessor matrix at program start. Then, for a given start and goal state the transition sequences can be looked up with the help of the predecessor matrix.

3.6.6.2 Transition Validity Check

Not every pair in $\{(x, y)|x, y \in S\}$ is connected by a valid transition sequence. Those pairs can be identified by an empty predecessor array in the matrix. In that spirit, state transitions and state transition sequences are valid if an non empty predecessor array exist. If an invalid transition sequence is requested, no subdivision is applied and the element keeps its previous state.

3.6.7 State Machine Realisation and Work Flow

Figure 3.8 on the next page outlines the the work flow of our state machine implementation and illustrates interaction of intersection detection and subdivision procedures of one time step. At first a subset of the model's tetrahedra, which are potentially in intersection with the the cut path, are identified via collision detection. This way the number of elements required to undergo time costly intersection tests can be limited. For all elements in the subset *collision tetrahedra*, the intersection detection procedures are executed. All occurrences and coordinates of face and edge intersections are stored for each tetrahedron. Intersection occurrences are stored as digit code. A tetrahedron is added to the set of *penetrated tetrahedra* if any face or edge intersections with the cut path of the current time step are found. The set of penetrated tetrahedra also holds those elements that are still under temporary subdivision. These are tetrahedra that have not reached an end state yet.

Hereafter, testing starts for changes in states for each tetrahedron Δ_i in the set of penetrated tetrahedra. If no intersections were found for a tetrahedron Δ_i, it is checked for state changes that occur not due to new intersection, but changes in the bearing of the tool. For instance, this is the case when the tool leaves Δ_i by passing from state 0xA1 to state 0xA3 (see Figure 3.7 on page 41). These kinds of state transitions do not require elementary cut operations like splitting edges or faces, yet they are necessary for tetrahedra to reach an end state.

If an intersection was found for a tetrahedron Δ_i then it has to be investigated if the intersection was dealt with before or if it is freshly found. The former case occurs when the tool sweeps through a face and the intersection point of the already intersected face has to be adapted to the the tools new position. This way the progressive follow-up of the cut to the position of the cutting is modeled. In the latter case it is likely that a state transition has to be applied. Then, the determination of the current state is conducted according to the recently found intersections. The new state is tested for validity. If the new state is valid Δ_i stores its current state and the state of the previous time step. If the state is not valid the current state of Δ_i is kept.

Now the best suited transition sequence $t = (s_0, s_1, ..., s_n); s_i \in S$ from the previous state s_0 to the the current state s_n is taken from the state transition graph Γ. For ev-

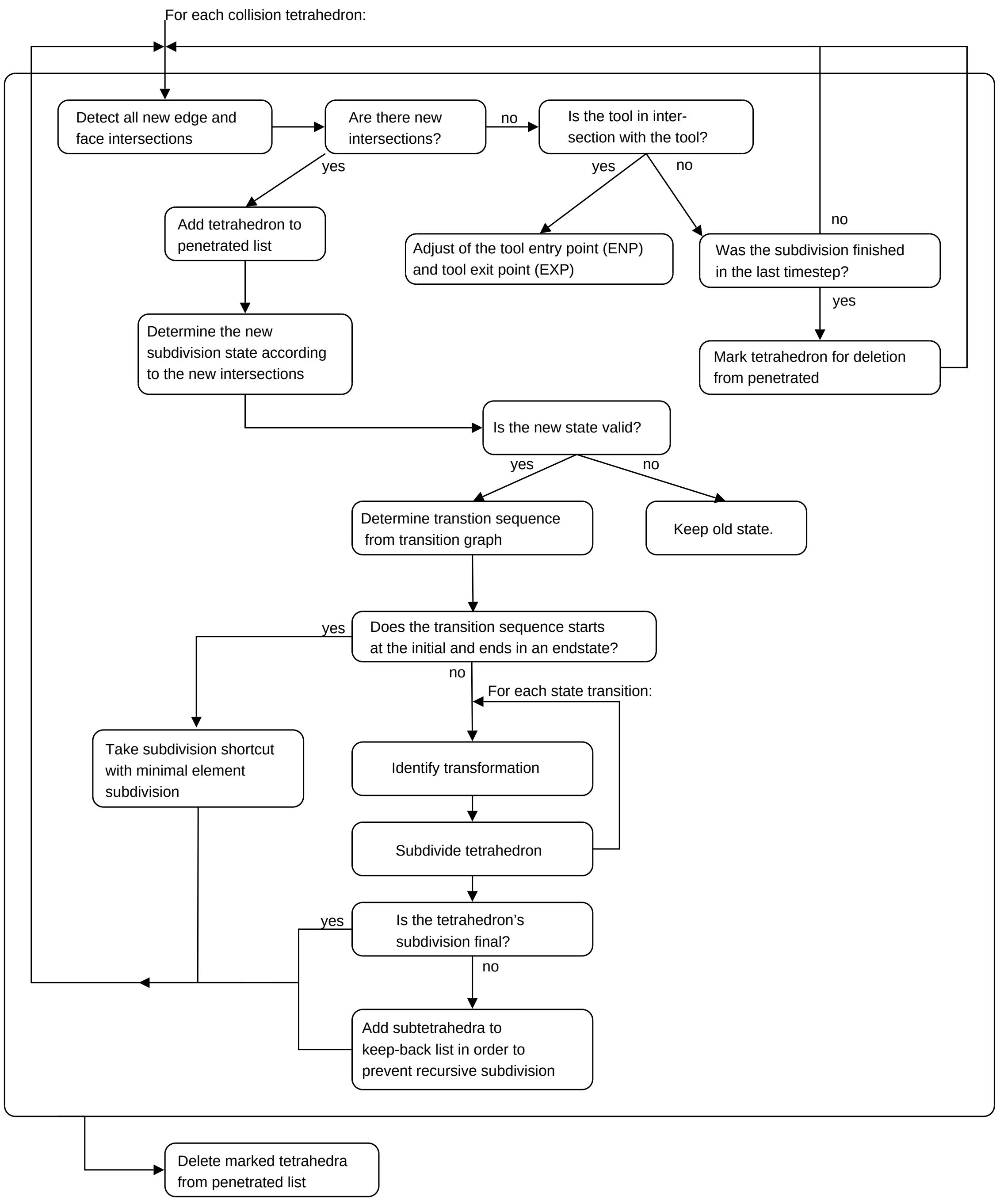

For each collision tetrahedron:
Detect all new edge and face intersections
Are there new intersections?
no
Is the tool in intersection with the tool?
yes
yes
no
Add tetrahedron to penetrated list
Adjust of the tool entry point (ENP) and tool exit point (EXP)
no
Was the subdivision finished in the last timestep?
yes
Determine the new subdivision state according to the new intersections
Mark tetrahedron for deletion from penetrated
Is the new state valid?
yes
no
Determine transtion sequence from transition graph
Keep old state.
yes
Does the transition sequence starts at the initial and ends in an endstate?
no
For each state transition:
Take subdivision shortcut with minimal element subdivision
Identify transformation
Subdivide tetrahedron
yes
Is the tetrahedron's subdivision final?
no
Add subtetrahedra to keep-back list in order to prevent recursive subdivision
Delete marked tetrahedra from penetrated list

Figure 3.8: The work flow diagram of the state machine algorithm.

ery transition in the time step t the necessary transformation is acquired in order to map the present intersection distribution to the particular implemented subdivision pattern. Then a set of subdivision operation that corresponds to the state transition (s_i, s_{i+1}) is applied to Δ_i. A special case of state transition is formed by transitions where a tetrahedron directly changes from the initial state into an end state within one time step. Since this case does not require handling of temporary subdivisions, the tetrahedron is subdivided with its final configuration of a minimal number of sub-tetrahedra.

In order to prevent the recursive subdivision of temporary sub-tetrahedra before the subdivision of the parent element is finished, we keep a list with the ids of the sub-tetrahedra. When the collision detection returns all possibly intersected tetrahedra of the mesh, those tetrahedra are filtered from the collision tetrahedra set and prevented from being intersection tested. When Δ_i in the set of penetrated tetrahedra reached an end state in the last time step then it will be removed from the list of penetrated tetrahedra.

Furthermore, the data that is necessary to update the mass-spring model is collected during each time step.

3.6.8 Subdivision

For subdivision we use two sets of subdivision patterns. One set are the symmetric subdivision patterns introduced in Figure 2.5 on page 14. We abandon the sixth subdivision pattern proposed by Bielser et al. in order to keep our state machine smaller and more incomplex.

For special cases, when transition shortcuts are applied, we use the minimal elements subdivision patterns as implemented by Mor. et al. (see Figure 3.15 on page 56). In what terms these special cases are applied is explained more in detail in Section 3.6.12 on page 54.

On account of speed performance we resign to use the LUT technique as it is so common to store sequences of subdivision operations for all operation needed to realize a state transition. The reason for this is simply that reading operation instructions from a script is time costly, although it provides for more modularity. For the sake of interactivity we forgo the advantage of modularity and simply implement a subdivision handler that consist of one procedure for every state transition, containing a subdivision operation set. In the following the implementation of the specific subdivision operation, face splitting, is explained. Edge splitting was presented in Section 3.5.1 on page 38.

3.6.8.1 Face Splitting

If a second edge of a face is intersected during a cut, then the face has to be split into two severed parts. The symmetric face subdivision patterns that we adapted in our algorithm suggests the insertion of two face midpoints m_1 and m_2 as displayed in Figure 3.6 on page 39(a). This section discusses how we acquire the coordinates for the two new midpoints to be inserted.

The first face midpoint m is inserted if a face's first edge is split and a incision is modeled into the tetrahedral face. This midpoint then follows the intersection point of the cutting tool until the next edge is cut. Then the face midpoint is set back to a specific, reasonable position within the face that has to be determined as explained in the following.

3.6.8.1.1 Midpoint Sampling In order to achieve a cut path approximation that approximates the cutting movement directed by the user more closely, the midpoint is calculated as the arithmetic mean out of all detected face-tool intersections as proposed by [4]. All detected n face intersections, starting from the first intersection where the cutting tool first intersects the face at time point t_i to the last intersection of the face at t_{i+n}, are summed up incrementally with each time step (see Figure 3.6 on page 39(b)). When the tool leaves the face the new averaged face midpoint position is calculated by multiplying the summed up vector with the reciprocal of the scalar n, where n is the number of face intersections. The midpoint m for the set of intersections can be observed in Figure 3.9 on the next page(c).

$$m = \frac{1}{n} \sum_{i=0}^{n} \vec{p_i} \tag{3.17}$$

3.6.8.1.2 Midpoints for Non-intersected Faces If a the tool sweeps through a face within one time step no face intersection for midpoint sampling can be obtained. The two edges of the face are consequently cut simultaneously within the time step. In this case the arithmetic mean of the two edge intersection points yield the averaged midpoint m.

3.6.8.1.3 Modeling the Incision Gap In order to model the cut's gap, the point m has to be split into two midpoints m_1 and m_2. Both points have to model a gap into

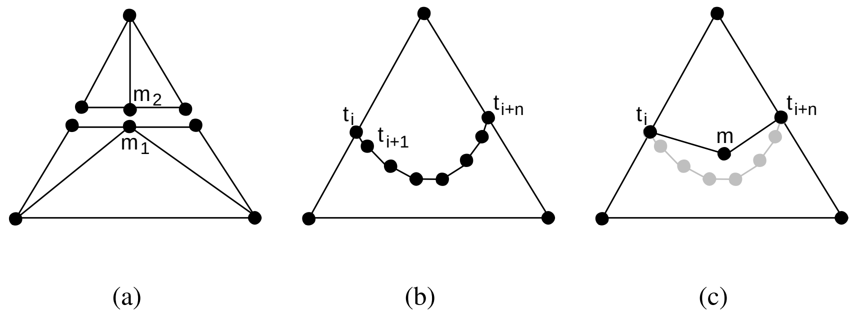


Figure 3.9: Finding the first averaged midpoint: (a) if two edges of a tetrahedral face are intersected the face is supposed to be split and two midpoints need to be inserted. For this the all recorded intersection point of the face are sampled (b) and the average point m is obtained (c).

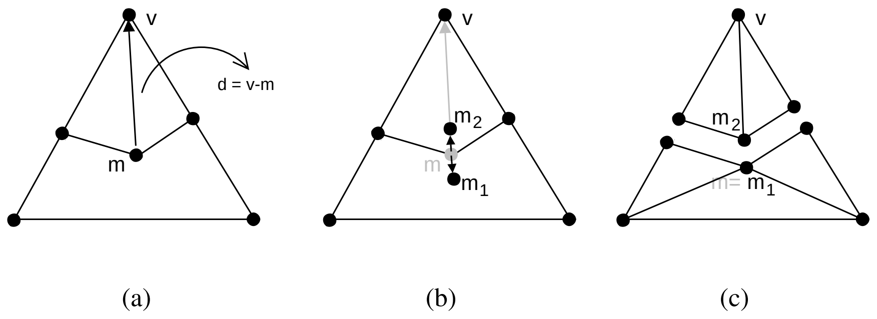


Figure 3.10: The insertion of the second midpoint: (a) $\vec{d}$ gives the axis on which direction the two midpoints m_1 and m_2 are shifted away from each other, (b) shows the inserted midpoints to the proper position, and (c) displays the triangulated face with m_1 and m_2 modeling the incision gap.

the face center. Thus, they are shifted along the direction vector $\vec{d}$ that is directed from the sampled midpoint m to the face vertex v, where v is the vertex that both cut edges have in common. m_1 and m_2 are calculated as follows:

$$m_1 = m - k \cdot \frac{\vec{d}}{||\vec{d}||} \tag{3.18}$$

$$m_2 = m + k \cdot \frac{\vec{d}}{||\vec{d}||} \tag{3.19}$$

where $\vec{d} = v - m$ and k is a scalar that defines the width of the cut's gap. The process of inserting the second midpoint and adjusting the midpoints positions is illustrated in Figure 3.10.

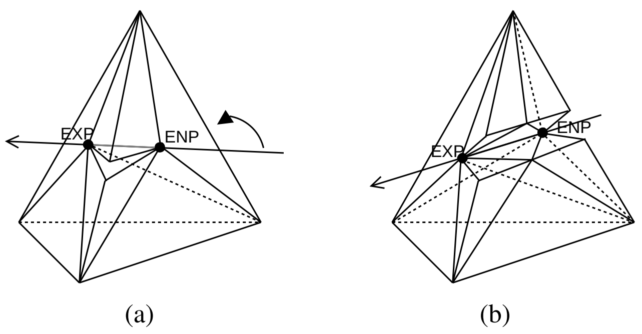


Figure 3.11: Example for a state transition illustrating ENP and EXP usage

3.6.9 Progressive Adjustments

The permanent pursuit of the modeled cut up to the cutting tool's position requires a continuous adjustment of the latest face intersection points. The points in question are the intersection points of the tool with the faces that are currently penetrated by the tool. When the tool moves further through a face these intersection points have to be adjusted to the coordinates of the cutting tools intersections. A tetrahedron can possess none, one or two of such kind of points. If a tetrahedron has one tool intersection point, that is, the tool is plugged into the tetrahedron but does not stick out of it, then there is one point referred to as tool entry point (ENP). In case the tetrahedron is penetrated at two faces, both a tool ENP and a tool exit point (EXP) exist. Figure 3.11 exemplifies the adjustment of the ENP during a change from state 0xA0 to 0xB0.

3.6.10 Temporal Node Assignment

There is one major complication with progressive cutting that deploys symmetric subdivision patterns. It was discovered with the original state machine approach by Bielser et al. The problem is that for a tetrahedron it can not be known in advance what the final cut configuration will be required when a temporary substructure is added. This problem arises due to the fact that it is not known in advance in which direction a cut will be continued or if it will be continued at all. The mesh has to be consistent at all times, so a tetrahedron which has been intersected at one edge will be subdivided into a set of sub-tetrahedra before the final end state is known. As soon as a tetrahedron has new intersections, it changes its state and the temporal substructure has to be altered. Then there is the fact that we forgo the procedure to delete all sub-tetrahedra of an old state just in order to insert the a sub-tetrahedra configuration of the a new state due to

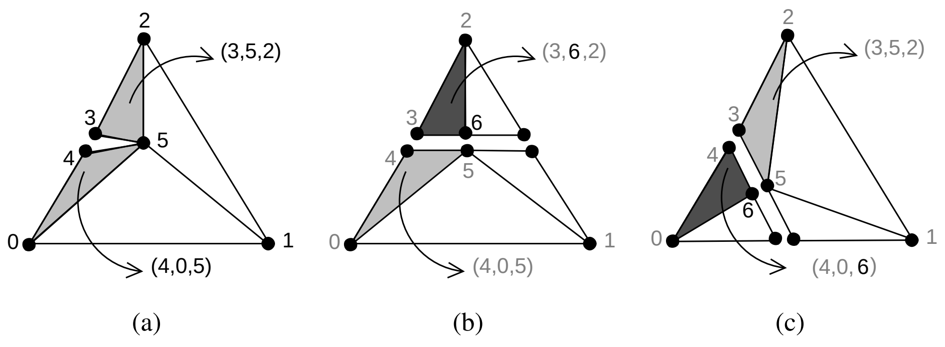


Figure 3.12: The complication with temporary subdivision is that with a further edge intersection, it can not be known in advance which edge, (0,1) or (1,2), will be cut. Dependent on which edge is cut the outcome of the point id distribution for the sub-triangle nodes at the midpoint are different. Thus, in case of a transition of (a) to (b) or (c) one mid point id of either triangle (3,5,2) or (4,0,5) has to be updated.

performance reasons. This is merely possible on account of the symmetric properties of the subdivision patterns. Already existing sub-tetrahedra have to be accommodated to the new situation. This has the effect that for inserted tetrahedra from previous state transitions the already assigned vertices of face intersection points have to be substituted by new points.

Figure 3.12 depicts a 2D example. For a face that has an incision at one edge there are three possibilities to continue the cut. Either the cutting tool leaves the face without cutting another one of the intact edges, or it cuts through one of the remaining edges. In the latter case the face has to be split. That requires the introduction of a second face midpoint and the update of one of the already inserted sub-triangles. It is computational expensive to determine with hindsight which point of an element has to be reassigned to the second midpoint.

Bielser et al. solve the problem by introducing an indirection on data structure level. It allows an automatic reassignment of the point ids. For this purpose they virtually presplit every tetrahedral face and hold three virtual midpoints corresponding to each face vertex. When a face is intersected the one real midpoint is created with the three virtual points referring to it. The inserted sub-element use the virtual points, not the real inserted midpoint. If another edge is split then a second midpoint is inserted and the three virtual midpoints are assigned to the two real midpoints accordingly. With the use of virtual nodes the there is no need to trace back point ids already inserted into the data structure of the tetrahedra objects. It is sufficient to update the references of the virtual midpoints to the real midpoint.

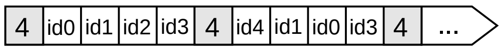


Figure 3.13: The storage of mesh elements: A continuous cell array holds the point IDs of the four vertex points for each tetrahedron in the mesh.

However, our data structure for the mesh geometry is predetermined by VTK (Visualization Toolkit)*; the visualisation system we use. This data structure compels us to use an alternative approach for solving the problem. In the following we explain necessary parts of the data structure used and report on our solution.

The tetrahedra of the mesh are held in the form of a continuous cell array that stores the four point IDs of the vertex points for each tetrahedron. The n-th point ID of a tetrahedron with the element ID k can be retrieved at the $k * (n + 1)$ position in the array (see Figure 3.13). This data structure is called vtkCellArray in VTK and belongs to the data object vtkUnstructuredGrid, which we use for storing geometry information and visualisation. With the cell array of VTK it is not possible to keep pointers to integer values instead of integer values. Thus, we had to provide for a different way of indirection.

For the storage of intersection occurrences, transformation and state information virtual tetrahedra instances with an ID corresponding to tetrahedra in the cell array are used. In order to solve the problem of updating point IDs of sub-tetrahedra we keep pointer into the cell array for those points that are possibly affected by a point ID update. These pointers are stored with the virtual tetrahedron according to its face ID and corresponding face point ID.

Now the update method is explained on a 2D example. If a face is incised at one edge as in Figure 3.14 on the following page(a) all sub-elements that are inserted with the face midpoint with ID=5 keep pointer into the cell array to the very same midpoint ID. If the face has to be split into two parts in a following time step then the point IDs of the affected sub-elements can be updated via the references of the midpoint pointers as in Figure 3.14 on the next page(b). The definition which point IDs of which sub-elements must be updated is given in the subdivision operation sets of the proper state transition. As a consequence, transformation are applied to the update pointer as well as to the subdivision operations.

*Visualization Toolkit: www.vtk.org

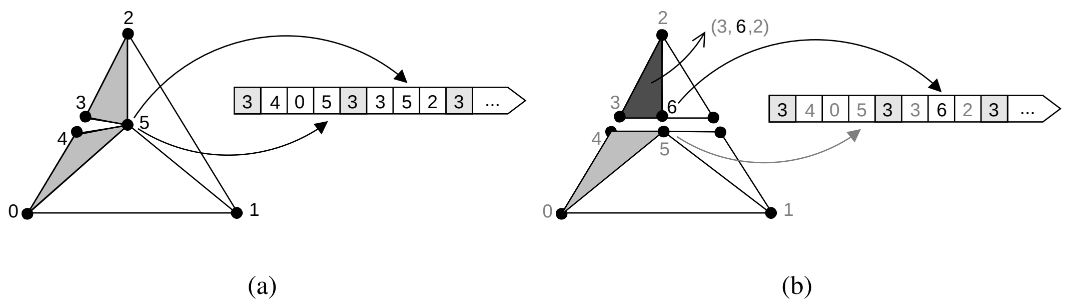


Figure 3.14: Our implementation of indirection: (a) When a face is split all sub-elements that share the midpoint ID keep pointer into the cell array to this midpoint ID. (b) If the face is split the cell array can be updated automatically with the help of the mid point pointers.

3.6.11 Maintaining Mesh Connectivity

In order to maintain the connectivity of the mesh after cutting adjacent edges and faces are split uniformly only once. This way it is possible to prevent the generation of jagged cut surfaces. In our implementation we solve the matter by centrally storing edge and face representations once per mesh primitive independent from the number of adjacent elements that share the primitive. Edges and faces are stored in connectivity maps that refer to potential new IDs of mass points due to the split of the primitive. The primitives can be identified by the point IDs of their vertices.

If a tetrahedron is intersected at an edge and the edge is supposed to be split into two then it is first checked if the edge was already split by the subdivision procedures of an adjacent tetrahedron. This is done with the help of a mapping function from edge ID to the new points e_0 and e_1 as outlined in Algorithm . If the two new points IDs e_0 and e_1 already exist in the edge connectivity map c these ids are used for further subdivision procedures and the splitting procedure for the edge is skipped. The corresponding algorithm is pointed out in Algorithm. The result that is obtained with the usage of the edge connectivity map is illustrated in Figure 2.10 on page 26(b).

All issues discussed above also apply if a face is cut and there is an adjacent tetrahedron. All face points should be added to the mesh only once for two tetrahedra that share a face. Consequently, there is a connectivity map for face midpoints too.

```
procedure determineSplitPointIDs($\Delta$, $p_0$, $p_1$)

1. access edge connectivity map $c$ of $\Delta$ with $key$ =($p_0$,$p_1$)
```

```
2. if($c$ maps to a valid pair $(e_0, e_1)$ ) then
       return $(e_0, e_1)$
   else
       retrieve $(e_0, e_1)$ with procedure splitEdge($\Delta$, $p_0$, $p_1$)
       insert $(e_0, e_1)$ into $c$
       return $(e_0, e_1)$
```

Splitting an edge $E(p_0, p_1)$ with vertex IDs p_0 and p_1 for a tetrahedron Δ that is shared by several adjacent tetrahedra

3.6.12 Shortcut Transitions

State transitions starting at the initial state and ending directly in one of the end states do not need to pass trough intermediate states. This is possible because all intersection are determined within one time step and the possible cut configuration and the corresponding final cut pattern can be determined immediately. This is also the case when non-progressive cutting is performed. The state machine is extended by what we refer to as shortcut transitions. A shortcut transition is implemented as an operation set that subdivides untouched tetrahedra with a permanent configuration. We realised shortcut transitions for state transitions from state 0x00 to 0xA3, 0xB3, 0xC3, 0xD3, and 0xE3 respectively. The operation sets of the shortcut transitions use a minimal set of tetrahedra for the subdivision. The subdivision patterns as shown in Figure 3.15 on page 56 are used. The purpose of the five shortcut operation sets is to avoid the realisation by costly and redundant state transformation sequences and to reduce element count without using deletion methods.

3.6.13 Hierarchical Subdivision

Up to now it was assumed that edges of tetrahedra can be cut only once. This is only true as long as the cutting tool is in intersection with a tetrahedron. The temporal substructure is prevented from being subdivided recursively until a finial subdivision is found. The tetrahedra of the final subdivision then are released from a keep back list. Afterwards, hierarchical subdivision can take place. Consequently, sub-tetrahedra can be cut again after the tool left the hierarchical higher tetrahedron. This way loop cutting can be realised where the cut path of the tool intersects itself. Thus, a tetrahedron's edges are restricted to be cut only once for one state machine pass through. An already cut edge can then be cut again by cutting the inner, hierarchical deeper lying tetrahedra.

3.7 Minimal Element Subdivision within State Machine Framework

In our second cutting method, we deploy minimal elements subdivision patterns (as Mor et al.) within a state machine framework. As experiments will show in the next chapter the increase in mesh quality In contrast to the original state machine framework proposed by Bielser et al. this keeps the mesh complexity lower. Symmetric subdivision patterns have the advantage that progressive cutting gets by with minimal removal operations, but they increase the element count rapidly.

The method proposed by Mor et al. [5] however, deletes elements of temporary subdivisions completely, just to replace them with elements of the next temporary subdivision. This is computationally expensive. Our approach forgoes removal operation when possible and uses update procedures instead. This is the case whenever the element count within a tetrahedron increases or remains the same during transitions between temporal subdivisions. The present sub-tetrahedra arrangement is updated and missing tetrahedra are added. Updating works via a pointer indirection into the continuous cell array of the model's mesh representation (as Figure 3.13 on page 52) and is quite similar to the one explained in Section 3.6.10 on page 50. When the spatial arrangement of the sub-tetrahedra within a temporary subdivision changes, the affected point IDs of those sub-tetrahedra are adjusted accordingly.

This method uses the same state machine frame work as outlined in the previous main section for the shortcut improved symmetric approach. Therefore, we only point out some crucial differences.

3.7.1 Minimal Tetrahedral Subdivision

The minimal element patterns as they are also used by Mor et al. for progressive cutting via subdivision are depicted in Figure 3.15 on the following page. The pattern in Figure 3.15 on the next page(a) corresponds to the set of designation 0xA_, the one displayed in Figure 3.15 on the following page(b) to 0xB_ and so on. In the following we go into the differences of this subdivision approach to the state machine framework with symmetric subdivision.

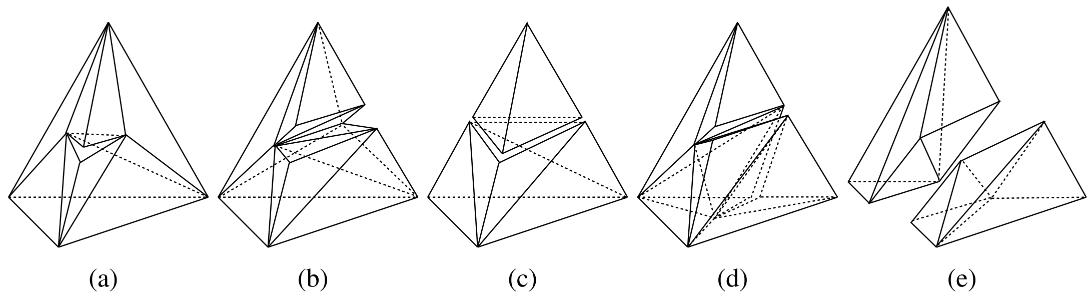


Figure 3.15: The five minimal elements subdivision patterns used.

3.7.2 Elementary Subdivision Procedures

Since only vertices that are introduced due to permanent face intersections and permanent edge intersection (see Section 3.4 on page 33) are used for the subdivision, the face splitting procedures differ from those of the symmetric approach. Minimal element subdivision forgoes the insertion of face points in the case a face is split into two parts. This renders the usage of an indirection data structure for the management of temporary subdivisions as pointed out in Section 3.6.10 on page 50 unnecessary. For this reason, the face splitting procedure is basically simplified to the split of the two intersected edges. Consequently, the determination of face points for non-intersected tetrahedral faces as in Section 3.6.8 on page 47 is also redundant.

The edge splitting technique works analog to the one of the symmetric subdivision. Moreover, the face point adjustment for the progressive follow-up of the modeled incision up to the current position of the tool remains the same. Midpoint sampling (see Section 3.6.8 on page 47) is only used for permanent face intersections, since midpoint for symmetry reasons aren't needed anymore. Additionally, due to the prevention of midpoints the maintenance of the mesh connectivity differs in two aspects. Firstly, where the symmetric approach has to keep a face connectivity map with reference to two points for every split face and every incised face, the minimal element state machine approach keeps only reference to one value and only for incised faces. The second aspect is linked to the fact that deletion of tetrahedra are necessary in a few state transition cases. The connectivity map for incised faces must be updated in such a case. Shortcut transition are also applied to avoid unnecessary state transition sequences. For those shortcut transition consistently minimal elements patterns are applied that remain a face persistent mesh topology.

3.8 Hybrid Cutting Algorithm

This section introduces the third algorithm, a novel hybrid cutting approach that combines progressive and non-progressive cutting.

3.8.1 Progressive vs. Non-Progressive Cutting

As already stated in Section 2.5 on page 17 there are two ways to conduct subdivision dependent on the time when subdivision is performed for an element. With non-progressive cutting a cut is not visualised for the duration of the tool's first entry into the element until the tool's exit. The idea is to detect all intersection points between cut path and element before the subdivision is applied. A noticeable lag between the actual tool's position and the cut occurs. This affects the realism of the simulation, especially for low refined meshes. Progressive cutting on the other hand, subdivides elements while the cutting tool is still in intersection with them. The cut is modeled up to the cutting tool's position at all times. Consequently, progressive cutting prevents a lag-behind effect and yields a visually more satisfactory result. However, progressive cutting is more complex in matters of time and storage for managing the temporary subdivision than non-progressive cutting, since not all intersections of an element with the cutting tools trajectory have been determined when the subdivision is applied. Temporary intersection configurations and current subdivision topologies have to be stored. While the cutting tool moves further into the model more edges or faces of an element are intersected. Then it is necessary to substitute the currently present temporary subdivision by a new one that corresponds to the new intersection situation. Additionally, progressive cutting poses more instability problems to the soft tissue simulation, since for short time periods very small and degenerated tetrahedra can be generated due to temporary subdivision. Degeneracies are tetrahedra that have short edges or a flattened shape. For instance, they are created if in the moment the cutting tool just sweeps into an element and the first temporary subdivision occurs (see Figure 2.11 on page 27(b)). Degeneracies are devastating for the simulation stability, since the numerical solvers for the soft tissue simulation can not cope very well with these inhomogeneous parts of the mesh. Non-progressive cutting methods have the advantage that they neither have to deal with the stability problems due to temporal subdivisions nor do they require the storage of additional state and temporary subdivision information.

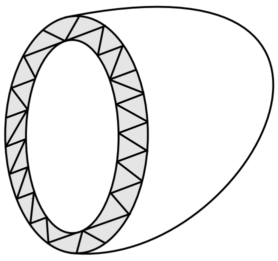


Figure 3.16: Cross section of a volume mesh: The surface tetrahedra of the outer hull in the model are cut progressively (gray). For the core part of the model non-progressive subdivision is applied.

3.8.2 Concept

In contrast to the former two algorithms of this chapter, which use progressive subdivision only, our hybrid approach combines both advantages of the visual appealing progressive cutting and the benefits of the in performance and stability superior non-progressive method. The crucial concept of the method is, firstly, to cut an outer margin of tetrahedra along the outer hull of the organ progressively (see Figure 3.16), and secondly, to apply non-progressive cutting to the inner core of the soft tissue model. This generates the impression of a smooth cut, where the user has the main visual access to the model and at the same time a more efficient algorithm for the model core can be used. The outer margin of the model's hull is given by tetrahedra that share a point with the surface.

Progressive cutting of the outer hull is realized via the state machine algorithm as it is outlined in Section 3.7 on page 55. We deploy minimal elements subdivision pattern within the state machine framework, in order to keep the mesh complexity low.

The inner core of the model is non-progressively subdivided, consistently with the same subdivision patterns as the outer mesh part. Those patterns correspond to the final states of a state machine. Degenerated tetrahedra also emerge when cutting is performed close to a mass point (see Section 2.7.1 on page 26). For stability increase, snapping was implemented for non-progressive cutting part of the algorithm. .

3.8.3 Identifying Surface Tetrahedra

Since our algorithm applies progressive and non-progressive cutting adaptively according to an elements position, surface tetrahedra need to be identified and updated during the cutting procedure. The initial set of tetrahedra is determined prior simulation start. Firstly, all tetrahedra are checked for faces that belong to the model's surface. This is realized with the help of the connectivity maps described in Section 3.6.11 on page 53. For each face the map is searched for an adjacent partner. In case no such partner can be found the face's vertex IDs are stored in a surface point IDs set. Hereafter, tetrahedra sharing one vertex with the surface point IDs set are identified and their reference is stored in a surface tetrahedra set.

In order to support consistent hierarchical subdivision it is necessary to add references of emerging surface elements to the set of surface tetrahedra. Obviously, such elements are created during cutting due to new cut surfaces along the incision. With minimal elements patterns only those tetrahedral vertices are used that emerge on account of edge or face intersections. Consequently, all sub-tetrahedra that fill the volume of the tetrahedra undergoing subdivision share at least a point with the emerging cut surface. For this reason, the surface tetrahedra set is constantly updated with elements that are inserted during the cutting process. For a tetrahedron that is subdivided progressively the addition of its sub-elements to set is conducted after the it has reached its end state. Beforehand the sub-elements are kept from being subdivided recursively anyway.

3.8.4 Snapping Realisation

In this section we outline our implementation of the snapping concept. In contrast to the work of Steinemann et al. [8], snapping is also applied to cut topologies that do not lead to the dissection of an element. In other words, we apply snapping not only for the patterns corresponding the end states 0xC3 and 0xE3, but also for 0xA3, 0xB3 and 0xD3. The set of minimal subdivision patterns was extended by six additional snapping patterns, as illustrated in Figure 3.17 on the next page.

Snapping patterns can be regarded as substitutions for the standard minimal element patterns. They replace the a minimal element patterns on certain conditions, which are the amount and the distribution of snapped vertices in a tetrahedron. One snapping pattern can be applied to to different topological cut cases depending on how many vertices fulfill the requirement to be be snapped onto the cut path. For instance, snapping pattern 0xF1 can be applied to a cut topology corresponding the pattern 0xA3 if one vertex has to be snapped. It can be also applied if for pattern 0xB3 exactly one edge has to be snapped and the both of the edge intersections lie within the distance of the snapping threshold. Consequently, there are certain conditions on which a specific snapping pattern substituted. We distinguish between snapping cases where either a

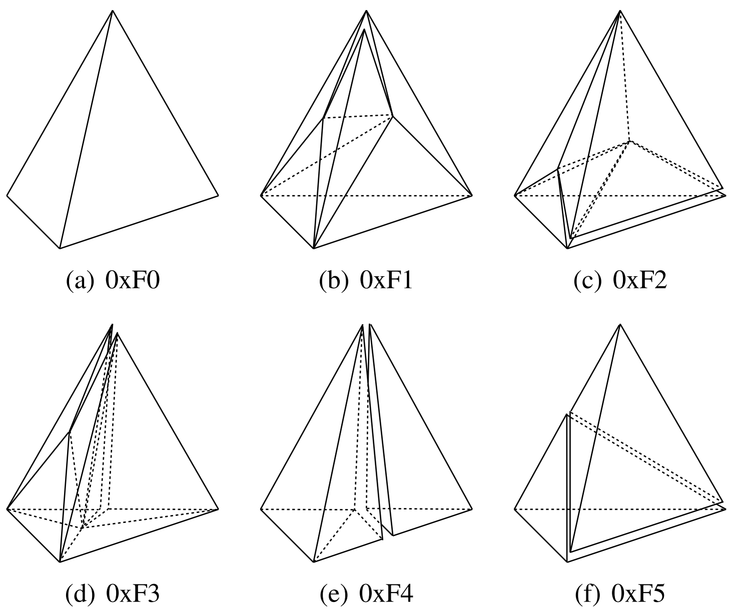


Figure 3.17: The six subdivision patterns with their corresponding state designations that extend the set of minimal element subdivision patterns in Figure 3.15 on page 56.

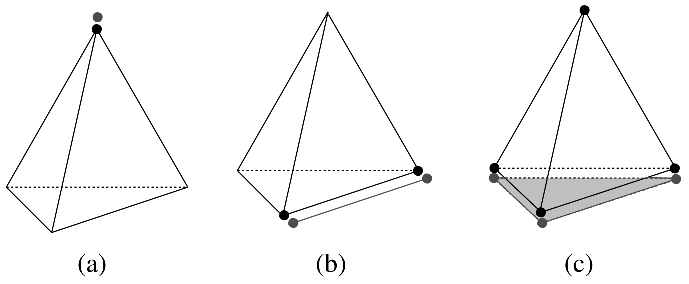


Figure 3.18: The three different versions of the snapping pattern of state 0xF0. Duplication of (a) of a vertex, (b) an edge, and a face.

vertex, an edge or face has to be duplicated. For example, snapping pattern 0xF0 can be applied in all three cases (see Figure 3.18).

Table 3.1 on the next page lists the different conditions under which a snapping pattern is deployed for subdivision instead of a normal minimal elements pattern. For each snapping pattern there are several application cases. The second column of the table specifies if a vertex, edge or face should be duplicated in order to model the incision gap. In the third column the state corresponding to the minimal elements pattern that would be applied for subdivision if no snapping was used is pointed out. Finally, the last column states on what condition the snapping pattern is applied.

Table 3.1: Application conditions for snapping subdivision patterns

Snapping Case	Dupli-cation	Unsnap. Case	# Snap Vertices	Conditions
$0xF0$	Vertex	$0xC3$	1	The three cut edges share the snapped vertex
	Edge	$0xC3$	2	Two cut edges share one snapped node; another cut edge shares the second snapped vertex
	Edge	$0xE3$	2	Two cut edges share a snapped vertex respectively
	Face	$0xC3$	3	Three snapped vertices are not shared by the cut edges
	Face	$0xE3$	3	Two of four cut edges share a one snapped vertex
$0xF1$	Vertex	$0xA3$	1	Snapped vertex is member of cut edge
	Vertex	$0xB3$	1	The two cut edges share the snapped vertex
$0xF2$	Edge	$0xB3$	2	
	Edge	$0xD3$	2	Two adjacent cut edges are cut close to two different snapped vertices
$0xF3$	Vertex	$0xD3$	1	
$0xF4$	Vertex	$0xC3$	1	One of the cut edge intersection is mapped to the snapped vertex only
	Vertex	$0xE3$	1	Two adjacent cut edges share the snapped vertex
$0xF5$	Edge	$0xC3$	2	
	Edge	$0xE3$	2	Three of the four cut edge intersections are mapped to the two snapped vertices and two adjacent cut edges share one of the snapped vertices

Chapter 4

Experiments and Results

4.1 Experiment Setup

In this chapter, we first introduce the setup of our experiments. A description of our data set and evaluation procedure follows. The results and interpretation of the experiments are presented.

4.1.1 Hardware and Software Setup

The tests of our the cutting procedures were conducted on a standard PC running Ubuntu Linux as operating system. It is equipped with 1.25GB RAM, AMD Athlon 64bit and a 3000+ with a Nvidia Geforce4 440 Go graphics card with 64 MB video memory. All cutting methods were entirely implemented in C++. For the visualisation of the model and the cutting tool, we employed the open source graphics library VTK [*]. The assembly of the system components, user interface and graphics pipeline was done with Python [†].The system was compiled with gcc version 4.0.2.

4.1.2 Data Sets

Different models are used in the experiments. There is a simple cube model in several resolutions used, that is homogeneous meshed. Since the cutting procedures are developed for surgery simulations, we used a second model that is anatomically shaped in form of a kidney. We tetrahedralized this model with varying numbers of tetrahedra. A kidney model of 4922 tetrahedra is used throughout this chapter for most of the experiments. The meshing algorithm used for the initial topology of the model before we apply cutting it is provided by the work of [48].

[*] Visualization Toolkit: www.vtk.org
[†] Python: www.python.org

4.1.3 Method Comparison

The following experiments compare the results of the three different cutting algorithms implemented by the author. Using the different cutting algorithms within the same simulation setup is possible on account of the modular system architecture that allows to replace the modules interchangeably. This way we ensure the cutting algorithms are tested in the same system environment. Specifically, we are comparing the following algorithms:

- the implementation of progressive cutting via a state machine framework using symmetric subdivision patterns as in the work of Bielser et al. [4],
- our shortcut improved state machine approach as presented in Section 3.6 on page 39,
- our hybrid cutting approach that combines progressive and non-progressive cutting and uses minimal element subdivision patterns as introduced in Section 3.8 on page 57,
- an implementation of the standard non-progressive minimal element subdivision method (MS), and
- two versions of the hybrid approach and the MS approach respectively, with to levels of snapping (as outlined in Section 3.8.4 on page 59) with thresholds of 5% and 15%.

4.1.4 Representational Cut Procedure

In order to ensure the same experiment setup we provided for reproduction of a cut path by scripting the cutting tools head and tail coordinates for each time step of a cut. This way a cut can be reproduced and the same cut path is applied to the same model using different cutting algorithms. We are conducting our experiments with a representational cut that dissects the kidney model (4922 tetrahedra) with a large area cut over 30 time steps. We chose the cut to be a common cut during virtual surgery simulation.

Table 4.1: The number of sub-elements created with five different patterns for symmetric subdivision and minimal element subdivision.

Cut Topology	Symmetric Subdiv.	Minimal Element Subdiv.
Figure 2.3 on page 13(a)	6	6
Figure 2.3 on page 13(b)	10	8
Figure 2.3 on page 13(c)	11	4
Figure 2.3 on page 13(d)	14	9
Figure 2.3 on page 13(e)	16	6

4.2 Mesh Complexity

Cutting that uses subdivision for incision modeling increases the number of elements within the mesh, since a cut element is split into several sub-elements. The magnitude of the element increase depends on the kind of subdivision method used. Table 4.1 compares the number of elements that are created with the five different cut topologies using symmetric subdivision or minimal element subdivision. The table illustrates that minimal elements subdivision generates less tetrahedra than the symmetric subdivision for all cut topologies, except for the first topology, where the number of sub-tetrahedra is identical .

A high increase of the element count raises the complexity of the model. A growing mesh complexity slows down the deformation simulation. Consequently, a good cutting algorithm would leave the model's complexity as low as possible without compromising the mesh quality or the result of the cut. Cutting inserts elements in a specific confined region and alters the mesh's topology. The element count is higher after a cut at local areas where the cut occurred. This causes the simulation to slow down. The concentration of mass points becomes more dense and the number of springs attached to one mass point increases the more sub-element are added to a subdivision. In comparison to the unmodified parts of the mesh, this locally comes with an alteration of the deformation behaviour and can lead to simulation instabilities. In the following we compare our methods with regard to the element increase.

Both Figure 4.1 on the next page and 4.2 show the element increase in our kidney model (4922 initial tetrahedra) over the duration of the cut. The abscissas of the diagrams indicate the time step. The ordinates display the overall number of elements in the mesh. The graphs plot the overall number of tetrahedra at each time step of the cut applied to the kidney model.

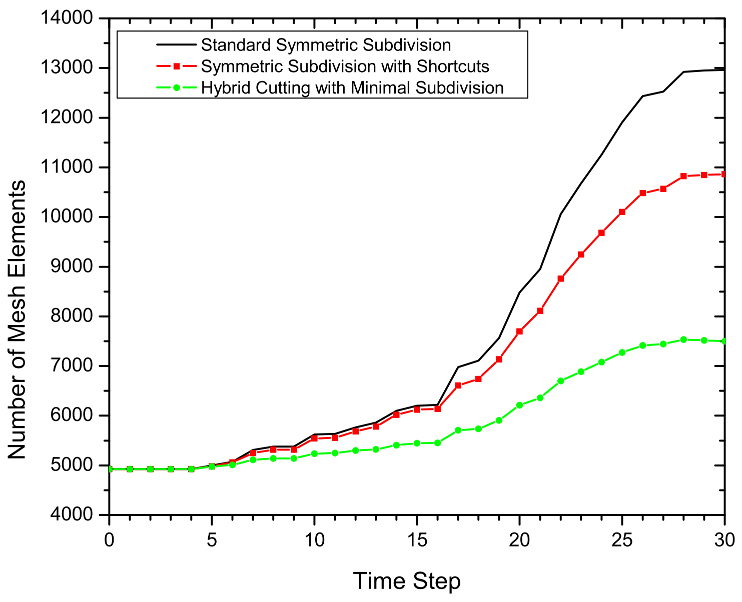


Figure 4.1: The overall element increase in the kidney model over the period of the cut with symmetric and minimal element subdivision implementations. Hybrid cutting generates much less mesh elements than the ordinary symmetric subdivision approach. Although the short cut improved algorithm clearly leads to an improvement.

The first diagram in Figure 4.1 compares the element increase of the ordinary symmetric subdivision, with our shortcut improved subdivision approach (Section 3.6 on page 39) and our hybrid cutting approach (Section 3.8 on page 57) that uses minimal subdivision patterns. The standard symmetric subdivision almost triples the initial number of tetrahedra during the cut. The shortcut improved symmetric subdivision leads to an improvement but still doubles the initial number of tetrahedra. The best performance has the hybrid cutting that generates 1.5 times as much elements as the initial number of elements.

Hybrid cutting keeps the mesh complexity lower than the ordinary symmetric subdivision approach and consequently promotes performance and speed of the soft tissue simulation. The shortcut improved algorithm clearly leads to an improvement, although it has still a worse result than the hybrid cutting approach.

In Figure 4.2 on the next page we compare two cutting approaches that use snapping. These are the hybrid cutting approach that applies snapping to the non-progressively cut core tetrahedra and non-progressive minimal subdivision cutting

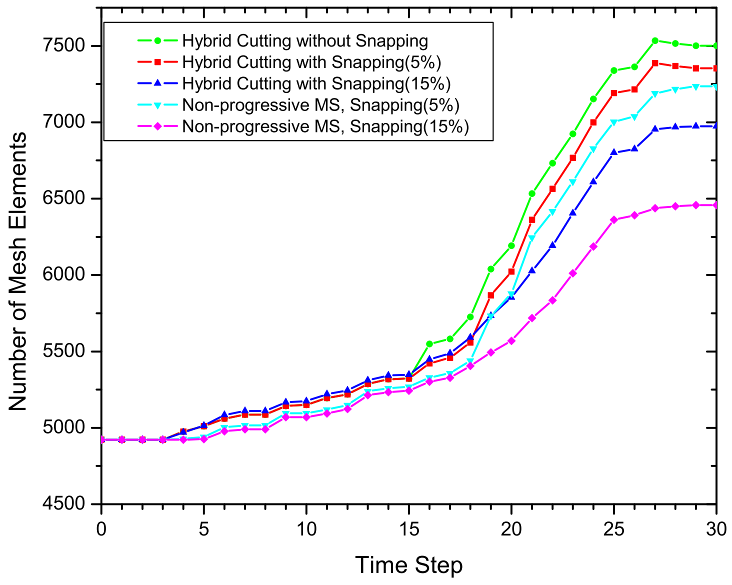


Figure 4.2: The overall element increase in the kidney model over the period of the cut with unsnapped hybrid cutting, snapped hybrid cutting and snapped non-progressive minimal element subdivision. It can be reasoned that snapping helps to lower the mesh complexity. Furthermore, hybrid cutting, which generates the impression of a continuous and smooth cut, also benefits from snapping and has a remotely worse result than the visually unappealing non-progressive method.

(MS) that is able to employ snapping on every element in the model. The curve of the hybrid cutting without snapping (circle line) from Figure 4.1 on the preceding page is plotted in the second diagram again for reference. There are two graphs for each cutting approach with two different snapping thresholds given in percent. The threshold value specifies when a tetrahedron vertex must be snapped. A vertex has to be snapped if an adjacent edge is cut within a distance of n% away from its position. The ascent and end results of the element count is highest for hybrid cutting without snapping, followed by the hybrid cutting with a 5% threshold for snapping, the non-progressive minimal subdivision (snapping threshold: 5%), the hybrid cutting with (snapping threshold: 15%), and the non-progressive minimal subdivision (snapping threshold: 15%).

It can be reasoned that snapping helps to lower the mesh complexity. Furthermore, hybrid cutting, which generates the impression of a continuous and smooth cut, also benefits from snapping and produces a remotely worse result than the visually unappealing non-progressive method.

4.3 Mesh Quality

Cutting alters the mesh topology and decreases the quality of the mesh. For a realistic and stable soft tissue simulation the mesh should be maintained as homogeneous as possible. Cutting with subdivision methods generate inhomogeneous mesh parts by:

- insertion of tetrahedra with different volumes that are smaller in comparison to the tetrahedra they replace
- creation of flattened tetrahedra with edges that differ significantly in length and have a bad edge aspect ratio

We define a local and a global mesh quality criterion in order to be able to measure the quality decrease that has been inflicted upon the mesh by the different cutting algorithms. The global criterion arises from the relation of all tetrahedral volume sizes to each other. This provides evidence about how much the tetrahedra are divergent in size. With this the amount of degenerated tetrahedra can be identified that come from too small volume instead from unbalanced edge lengths in one tetrahedron. In a mesh where volumes of tetrahedra have extreme variations the edge lengths of all tetrahedra in the mesh are unbalanced as well.

In contrast to this, the edge aspect ratio of one tetrahedron reveals information of the quality of the shape of a single element. Thus, the maximal edge aspect ratio, defined in Section 4.3.2 on page 71, yields a local mesh quality criterion. We will analyse the mesh according to those two mesh quality criteria in the following two sections.

4.3.1 Tetrahedra Volume

In order to visualise the relation of the volumes of all tetrahedra in the mesh, we measured the distribution of tetrahedra with different volume sizes in the model. The diagrams in Figure 4.3 on the following page and 4.4 show the volume distributions of all tetrahedra over the different volume sizes in the mesh before and after cutting. The tetrahedral volume is plotted on the x-axis, while the y-axis indicates the number of mesh elements. The number of tetrahedra occurrences with a specific volume be can extracted from the curves. A distribution curve was obtained by counting the tetrahedra that fall into a certain volume range. We distinguish into 100 different ranges for the

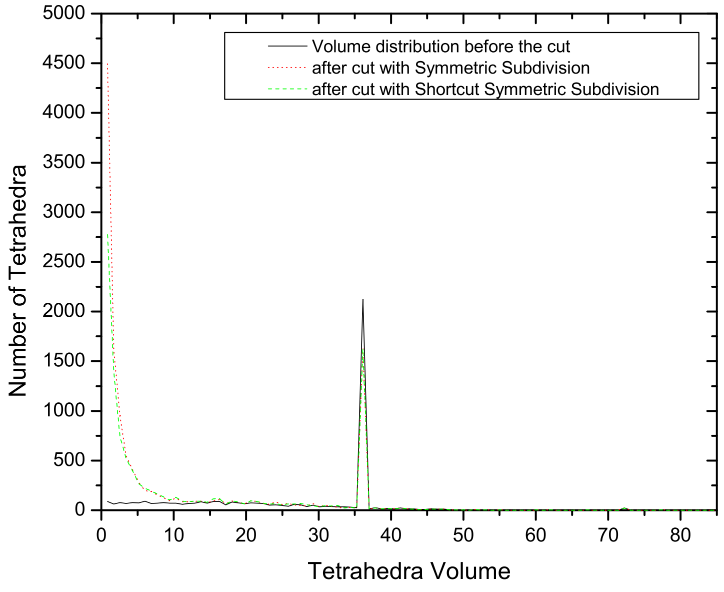


Figure 4.3: The volume distribution of tetrahedral volumes in the mesh before and after the cut with progressive cutting via symmetric subdivision approaches. With ordinary progressive symmetric subdivision more degenerated tetrahedra due to small volume are generated than with the shortcut improved cutting.

distribution measurement. The volume V of a tetrahedron falls into a range if the criteria:

$$V > \frac{V_{diff}}{100} n + V_{min} \tag{4.1}$$

and

$$V < \frac{V_{diff}}{100} (n+1) + V_{min} \tag{4.2}$$

are fulfilled. V_{diff} is the difference of the maximal volume V_{max} and the minimal volume V_{min} of all tetrahedra in the model and $n \in \{0, ..., 99\}$.

The solid curve represents the initial volume distribution of the kidney model (4922 tetrahedra) before cutting. In Figure 4.3 the dashed curve and the dotted curve show the change of the distribution after the completion of the same representational cut with the state machine algorithm that we presented in Section 3.6 on page 39. The dashed curve shows the modified volume distribution after progressive cutting via sym-

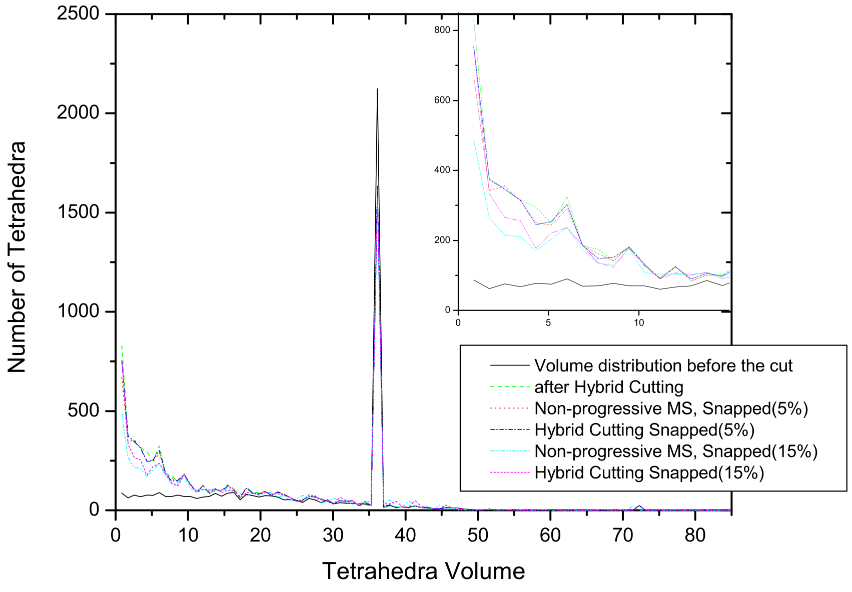


Figure 4.4: The volume distribution of tetrahedral volumes in the mesh before and after the cut with unsnapped hybrid cutting, snapped hybrid cutting and snapped non-progressive minimal element subdivision. In comparison to the symmetric subdivision approaches the distribution of all plots is more balanced in the lower volume region. The number of tetrahedra with extreme small volume is considerably smaller, especially when snapping is applied.

metric subdivision without shortcut transitions. The dotted plot uses the same algorithm with the improved shortcut subdivision transitions.

The initial tetrahedral volume distribution indicates a regular meshed model that mostly consists of elements with a volume of approximately 35 volume units and a few tetrahedra with smaller volume. Both curves indicate the generation of a considerable large amount of undersized tetrahedra after cutting. For symmetric cutting with improved shortcut subdivision the increase in small tetrahedra is slightly lower. Especially the amount of degenerated tetrahedra with a volume in the range of V_{min} to 0.85 is considerably lower (2774 tetrahedra) compared to the ordinary symmetric subdivision (4494 tetrahedra). Consequently the shortcut improvement leads to a slight improvement in mesh quality.

All the curves in Figure 4.4 for the hybrid cutting and the non-progressive minimal element subdivision (MS) show a better distribution after cutting than the symmetric subdivisions. The sub-figure within the diagram shows a magnified extract from the lower left main diagram.

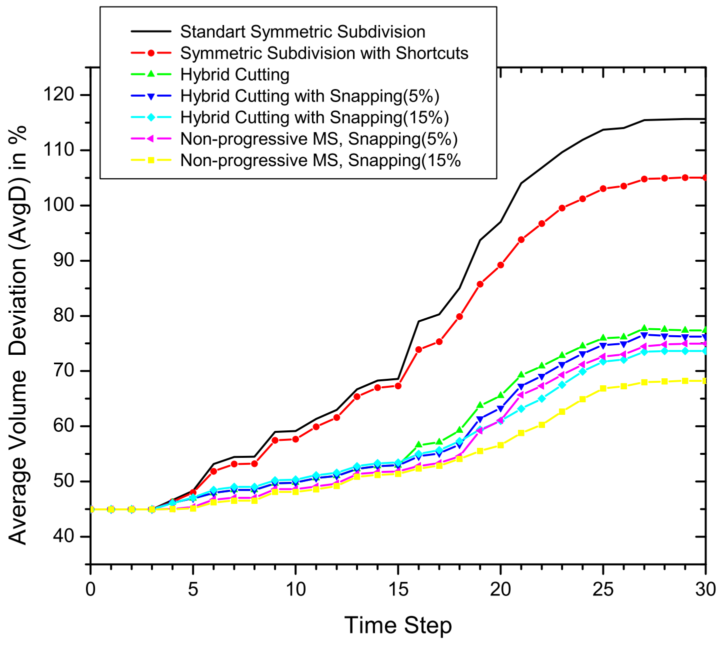


Figure 4.5: The average volume deviation for all methods over the duration of the cut.

Where the shortcut improved symmetric subdivision creates 2774 tetrahedra at the smallest range, the hybrid cutting generates only 827 tetrahedra. Furthermore, more tetrahedra with larger volume are created. The best volume distribution in the low volume range is achieved by the non-progressive MS algorithm with a snapping threshold of 15%. It is closely followed by the performance of the hybrid technique with the same snapping threshold. The result with 5% snapping threshold for both algorithms yield a slightly worse result. Here again non-progressive MS is shown to have a remotely better performance.

The display format of the volume distribution of all elements in the mesh can not serve as a relative comparison of the different methods for the relations of all volume sizes in the mesh. The volume distribution depends on the overall element number in the mesh. Since the different algorithms create a different amount of mesh elements during a cut, we additionally compare the average volume mean deviation of the average tetrahedra volume ($AvgD$) in percentage. This value gives evidence about the relative volume quality of the entire mesh, independently of the mesh size. The average volume deviation $AvgD$ is calculated as follows. An average volume V_{avg} is simply calculated by

$$V_{avg} = \frac{1}{n} \sum_{i=0}^{n} V_i \tag{4.3}$$

where n is the number of tetrahedra and V_i the volume of each tetrahedron. Then the $AvgD$ is determined by

$$AvgD = 100 \cdot \frac{|V_{avg} - V|}{V_{avg}} \tag{4.4}$$

Figure 4.5 on the preceding page plots the values of $AvgD$ during the cut. The number of time steps are shown on the abscissa. The values of $AvgD$ are plotted on the ordinate. The diagram illustrates that the cut with symmetric subdivisions results in a high ascent of the $AvgD$ value and indicates that volumes in the mesh have a high deviation after cutting. Hereby the shortcut improved algorithm has a slighter slope and finishes the cut with an average deviation of 115.7$\%$ to the average volume. The ordinary symmetric subdivision has an final value of 105$\%$. Compared to this, hybrid cutting achieves a much better $AvgD$ at all times. It ends at an average deviation of 77.4$\%$.

From the comparison of our snapped hybrid method and the snapped non-progressive MS method follows that snapping results in a more gently inclined slope of the $AvgD$ curve. The best result achieves the 15$\%$ snapped non-progressive MS method with a final result of 68.2$\%$ average deviation, followed by hybrid cutting with 15$\%$ snapping threshold (73.6$\%$), 5$\%$ snapped non-progressive MS (74.9$\%$) and hybrid cutting with 15$\%$ snapping threshold (73.7$\%$).

4.3.2 Edge Aspect Ratios

Since the volume distribution is not the only criterion for a good quality we also conduct measurements that can be used to derive conclusions about the shapes of the mesh's elements. For this we define a local mesh quality criteria, the maximal edge aspect ratio.

The diagrams in Figure 4.6 on page 73 and 4.7 illustrate the distribution of all tetrahedra in the mesh according to their maximal edge aspect ratio ($maxAR$) before and after cutting. The abscissa indicates $maxAR$, which is a value between 0.0 and 1.0. The ordinate gives the number of mesh elements. Thus, the plots show how much tetrahedra with a specific $maxAR$ exist in the model. The solid graph in both diagrams show the initial distribution of the $maxAR$ before the mesh modification. All other graphs provide information about the $maxAR$ distribution after cutting.

The $maxAR$ is a local mesh quality criterion that defines how well balanced the lengths of the six tetrahedral edges are. With $maxAR$ we can draw a conclusion about the shape of a tetrahedron. It is calculated by the quotient of the length of the shortest edge l_{min} and the length of the longest edge l_{max}:

$$maxAR = \frac{lmin}{lmax} \tag{4.5}$$

Consequently, a $maxAR = 1.0$ corresponds to a extremely well shaped tetrahedron, which edge length are all the same. A value close to 0 corresponds to a badly shaped, degenerated tetrahedron, that would cause simulation instabilities. We classify $maxAR$ of a tetrahedron according to 100 discrete values between 0.0 to 1.0. The number of tetrahedra with a rounded off $maxAR$ value after the second decimal digit behind the point are counted.

The initial $maxAR$ distribution (solid line) indicates that the major parts of the mesh have values around 0.58 and 0.70, which conforms to good local quality of the these element's shapes.

In Figure 4.6 on the following page the distribution of all tetrahedra in the mesh according to their $maxAR$ value for the two symmetric subdivision methods and the hybrid cutting after the cut is depicted. As it can be observed from the dotted curve, our hybrid cutting generates much less badly shaped tetrahedra in comparison the ordinary symmetric subdivision generates. Especially in the range of 0.0 to 0.15 the difference is high. Furthermore, here again our shortcut improved symmetric subdivision method (dashed-dotted line) has a better result than the ordinary approach. Consequently, the hybrid approach leaves the soft tissue model in a more stable state than the other methods.

The diagram in Figure 4.7 on page 74 shows the same aspect as Figure 4.6 on the following page for hybrid cutting and non-progressive minimal element subdivision with snap-thresholds of 5% and 15% respectively. Note that the distribution of the methods is petty similar. The sub-diagrams magnify the lower left part of the diagram. The upper sub-diagram compares hybrid cutting and non-progressive minimal element cutting with a snap-threshold of 5%, the lower sub-diagram shows a comparison with 15% snapping threshold. Both the upper diagram and the lower diagram show a slightly better aspect distribution for non-progressive minimal element subdivision than for the

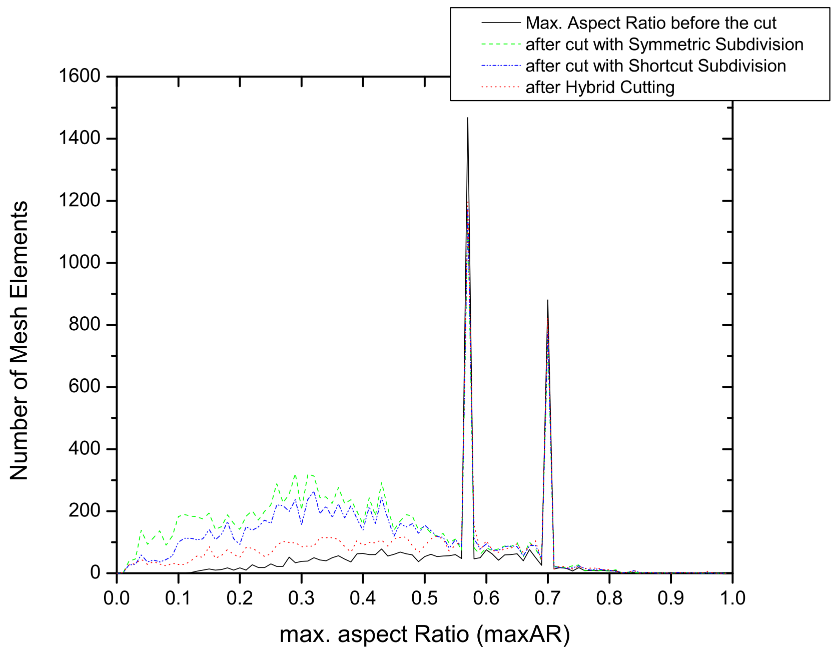


Figure 4.6: The distribution of the $maxAR$ values of all tetrahedra in the model before and after the cut with the ordinary symmetric subdivision, shortcut improved subdivision and hybrid cutting.

hybrid approach. However, considering that with hybrid cutting snapping can be applied to a part of the mesh only, it is noteworthy that the hybrid cutting approach is only minimally inferior to the non-progressive minimal element subdivision.

In addition it can be generally reasoned that snapping with a higher threshold also results in a better $maxAR$ distribution in the mesh.

However, the plots of the maximal edge aspect ratio distribution can not yield a relative comparison of the quality criterion, because the different algorithms generate a different amount of tetrahedra. For this reason we additionally compare a maximal aspect ratio quality quotient that is independent from the number of mesh elements and that yields evidence about the relative aspect ratio quality of the entire mesh. This maximal aspect ratio quotient Q_{AR} is determined by calculating the average mean of the $maxAR$ values over all tetrahedra in the mesh:

$$Q_{AR} = \frac{1}{n} \sum_{i=0}^{n} \frac{l_{min}}{l_{max}} \tag{4.6}$$

where n is the number of tetrahedra. The values of Q_{AR} over the duration of the sam-

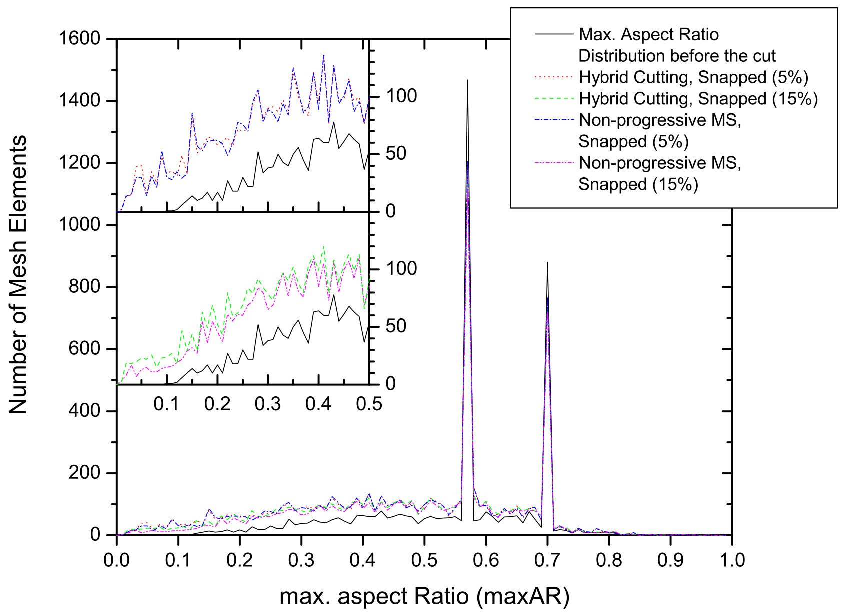


Figure 4.7: The $maxAR$ distribution of all tetrahedra in the model before and after the cut with snapped hybrid cutting and snapped non-progressive minimal element subdivision.

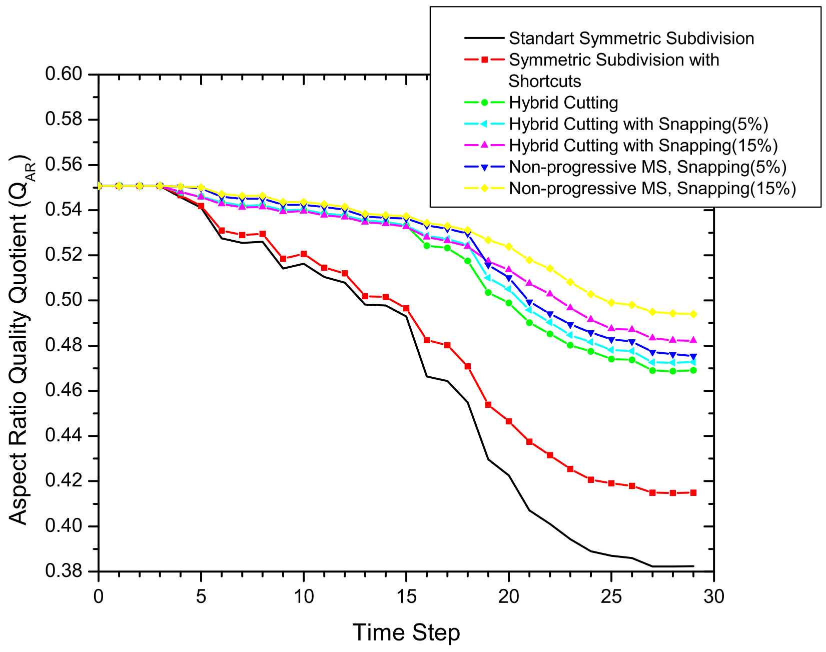


Figure 4.8: The averaged maximal edge aspect ratio of all tetrahedra in the mesh over the duration of the cut.

ple cut are plotted for all algorithms in Figure 4.8 on the previous page. The number of time steps are plotted on the x-axis and the values of Q_{AR} on the y-axis. Both curves of the symmetric subdivision approaches have a strong decay of the Q_{AR} value. The local aspect ratio quality for all tetrahedra is 0.38 after the standard progressive subdivision and 41.5 after the sample cut with shortcut improved symmetric subdivision. The decay for hybrid cutting is considerably slighter. The Q_{AR} is at 0.469 after the cut and indicates much better shaped tetrahedra. An improvement of this result is achieved by using snapping with our hybrid approach. With a snapping threshold of 5% the final value of Q_{AR} is 0.472, whereas with a threshold of 15% it is 0.482. In comparison to the standard non-progressive minimal element subdivision method (MS) with snapping, the snapped hybrid cutting performs only slightly weaker. The final average $maxAR$ value for 5% snapped MS is 0.476. For 15% snapping threshold it lies at 0.495.

Considering that the hybrid approach uses snapping only for a part of the mesh but provides the user with a progressive and more realistic visual feedback, it is worthwhile to put up with the relative small difference in the quality performance. The diagram also shows that, although symmetric subdivision gives the user a higher degree of freedom to specify a cut path, the loss on aspect ratio quality is very high.

4.3.3 Impact of Progressive Cutting on the Mesh Quality

Extreme shapes occur in temporary subdivisions for sub-tetrahedra every time the cutting tool is about to cut or just has cut an edge. These temporary degeneracies are characterized by very uneven edge aspect ratios and volumes. They lead to temporary instabilities of the mass spring simulation. This aspect is illustrated in Table 4.2 on the following page. It shows the aspect ratio and volume behaviour of the sub-tetrahedra for the dissection of a single tetrahedron by successively cutting three of its edges. In the state machine this corresponds to the transition sequence of one tetrahedron from initial state 0x00 over 0xA0 and 0xB0 to 0xC3. The same cut used for a progressive cut, corresponds to a direct transition from 0x00 to 0xC3. In the table, the last row represents the sub-tetrahedra created during the non-progressive cut. For reference, the maximal aspect ratio of the volume of every sub-tetrahedron after each transition is provided. The table states those values of the first time step directly after an edge was cut. The number of sub-tetrahedra is altered for progressive cutting after each edge cut. The table indicates that directly after the first and second edge cut two of the sub-tetrahedra have extremely small values. For non-progressive cutting only those

Table 4.2: The impact of progressive cutting on tetrahedral aspect ratio and volume.

State Transition	Max. Aspect Ratio	Volume in %
0x00 to	0.0063	0.006
0xA0	0.0131	0.004
	0.4689	41.9
	0.4606	1.2
	0.7324	55.5
	0.6273	1.4
0xA0 to	0.3152	7.4
0xB0	0.3561	5.4
	0.5684	38.1
	0.5087	9.1
	0.4751	23.2
	0.0055	0.006
	0.0046	0.005
	0.4586	16.08
0xB0 to 0xC3	0.8008	24.1
(progressive)	0.4926	17.7
0x00 to 0xC3	0.8944	28.0
(non-progressive)	0.4921	30.2

tetrahedra, which are generated due to the third and final intersection, are inserted into the mesh, leaving the aspect ratios more balanced for the duration of the cut.

This issue is also demonstrated in Figure 4.9 on the next page for a more complex mesh. The diagram illustrates the mesh quality decrease over the duration of a sample cut into a regular meshed cube of 3072 tetrahedra. 1776 of those tetrahedra are attached to the model's surface. The quality is measured by the average mean of the maximal edge aspect ratio over all elements. Here again the maximal aspect ratio is obtained by the quotient of the shortest and longest edge of a tetrahedron. The diagram shows that for progressive cutting the overall mesh quality is lower during the cut than for non-progressive cutting. Additionally, the progressive cutting graph possesses abrupt fall-offs into local minimal (cubed line). These originate from extremely shaped temporary sub-tetrahedra. With hybrid cutting less of these temporary degeneracies occur, since the non-progressive subdivided tetrahedra leave the aspect ratios more balanced.

Consequently, keeping the soft tissue simulation stable is simpler than with progressive cutting. Moreover, additional patterns for snapping can be easily integrated into non-progressive cutting method, while it is highly complex to integrate it for progressive cutting. It would also be necessary to apply snapping to temporary subdivision.

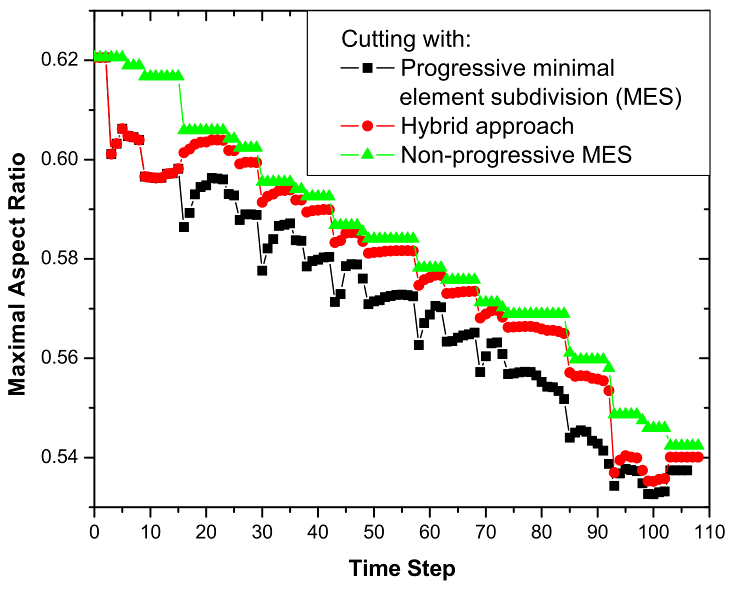


Figure 4.9: The decrease of the mesh quality during cutting with progressive and non-progressive minimal element subdivision, and hybrid cutting over the duration of a cut into a regular meshed cube of 3072 tetrahedra (1776 tetrahedra attached to the model's surface). Hybrid cutting helps to reduce the impact of temporary subdivisions and increases the simulation stability.

This would require much more resources and the state transition graph would become very complex, especially when attempting to prevent the impact of degeneracies due to temporary subdivisions. Thus, it is beneficial to avoid temporary subdivisions for a subset of the mesh, without the drawback of major visual drawbacks. In conclusion, the hybrid cutting approach not only assures an improved mesh quality and simulation stability after the cutting procedure, but also during cutting.

4.4 Algorithms efficiency

The hybrid method is best suited for voluminous models, since they have a small surface area compared to the overall volume. Another factor for the effectiveness of the method is the mesh quality. Although a sphere would be the ideal shape, the hybrid cutting would have no effect at all if the mesh consisted exclusively of surface points with only one point in the sphere's centre. All elements would be attached to the surface.

Figure 4.10 on the following page depicts the relation between mesh quality and

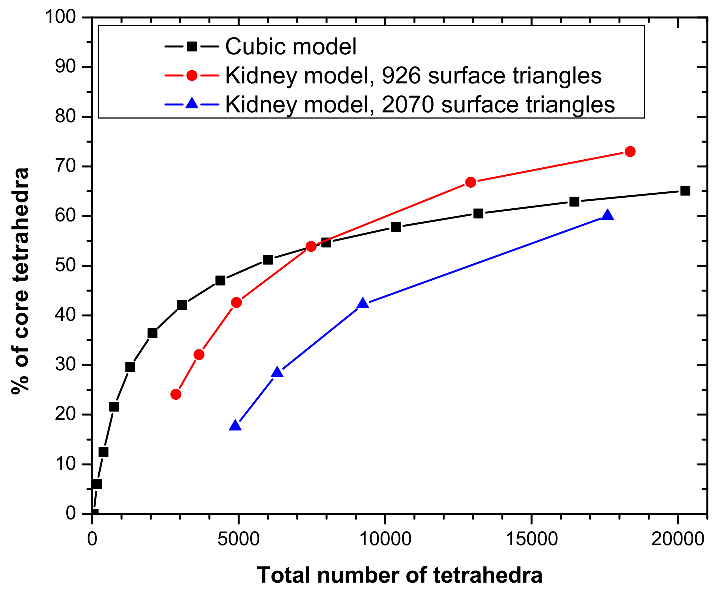


Figure 4.10: The efficiency of the hybrid cutting approach in relation to the element count. The efficiency depends on the relation of core tetrahedra and tetrahedra that share a point with the surface. Consequently, the hybrid method is best suited for voluminous models that are homogeneous meshed.

efficiency of the hybrid algorithm. Three models, one cubic and two kidney models are shown with different refinement. The cubic model's surface grows proportional to the increased tetrahedra count in the mesh. The graphs for the kidneys show the same organ with two different numbers of surface triangles. The element increase here is obtained by narrowing the mass point spacing within the model. Figure 4.10 shows that raising the ratio of mesh elements that do not share a point with the models surface leads to a gain in efficiency.

In the following we have a look on the efficiency in term of operation types that are used for the management of temporary subdivisions. The column diagram in Figure 4.11 on the following page shows the comparison of the overall number of used delete operations and update operations for four different algorithms over the interval of the cut. The number of operations used is plotted on the y-axis. The x-axis shows two types of operations used for the realisation of temporary subdivision replacements. We compare the standard progressive cutting approach via minimal element subdivision suggested by Mor et al. [5] to our implementation of minimal

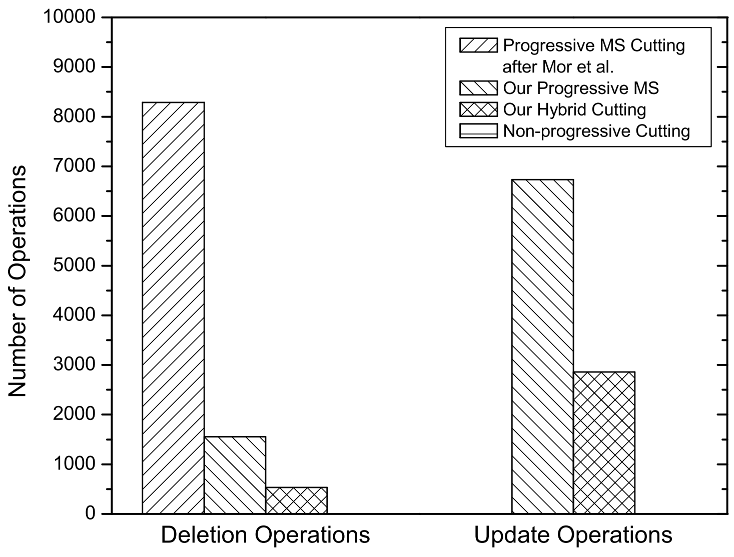


Figure 4.11: The comparison of the number of two types of operations used for the realisation of temporary subdivision replacements with different methods for the sample cut is shown. It is possible to avoid 4 out of 5 deletion operations with our implementation of MS via the state framework on account of a indirection on data structure level. Those deletion operations are substituted by computationally more inexpensive update operations. Hybrid cutting is twice as efficient in terms of temporary replacement handling as the MS approach.

element cutting and to our hybrid cutting approach.

Mor et al. delete sub-elements of temporary subdivisions when they need to be replaced. The deleted temporary subdivision is then replaced by a new temporary subdivision constellation. Deletion methods are costly, since elements need to be removed from the mesh data structure and from the representation of the deformable model, in our case continuous arrays. In our implementation we substitute the majority of the deletion operations by update operations with the help of an update pointer structure (as explained in Section 3.6.10 on page 50). The diagram illustrates that Mor's algorithm uses approximately 8000 deletion operations for the cut. The second column shows our progressive minimal subdivision approach. We are able to replace 4 out of 5 operations with update operation instead of using deletion operations. Since non-progressive cutting does not have to replace temporary subdivisions, none of the discussed operations need to be applied with non-progressive cutting. Our hybrid cutting that combines progressive and non-progressive cutting, avoids even more deletion and update operations

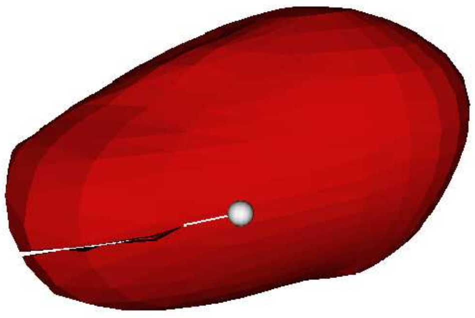


Figure 4.12: An incision into a model of a kidney.

while having a similar visual result like the two progressive minimal subdivision algorithms compared in the diagram. Thus, the hybrid approach is twice as efficient in terms of temporary subdivision replacements operations as our implementation of the progressive minimal subdivision approach.

4.5 Visual Performance

Figure 4.12 shows a flat shaded sample cut into a kidney with symmetric subdivision. In Figure 4.13 on the next page cutting into a highly refined kidney model with the hybrid cutting approach is illustrated . A comparison of the visual result of the very same cut with non-progressive minimal element subdivision and with our hybrid approach can be found in Figure 4.14 on the following page. Note the lack-behind effect that occurs with non-progressive cutting. The incision is not modeled up to the current tool's position and this creates uncontinuities while the cut is performed.

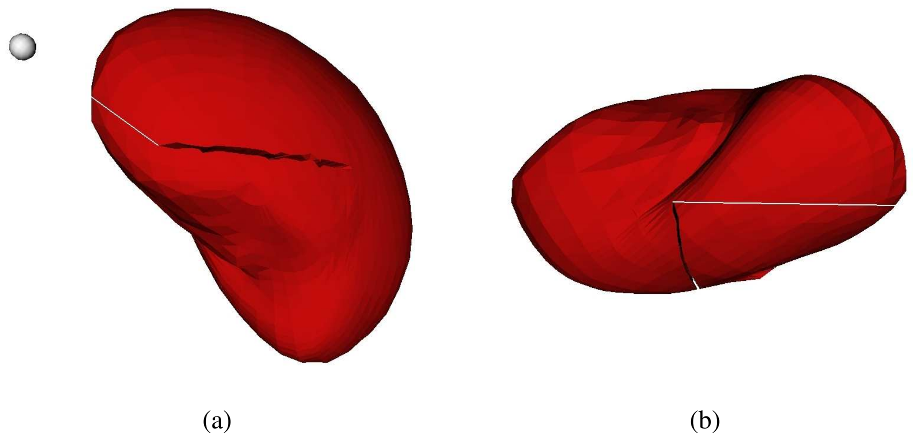


Figure 4.13: A cut to the kidney model with 2070 surface triangles and 7467 tetrahedra with the hybrid approach.

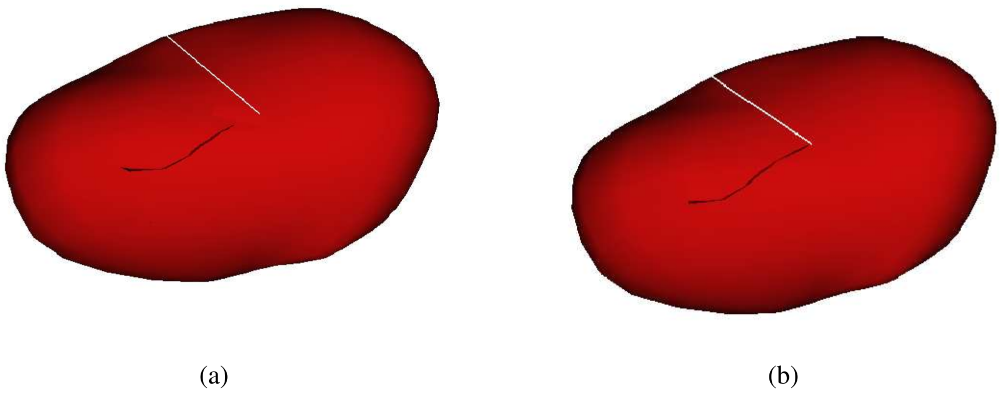


Figure 4.14: The same cut into a kidney with (a) non-progressive cutting and (b) hybrid cutting. The incision with non-progressive cutting is lacking behind the cutting tool's position.

Chapter 5

Conclusions

5.1 Summary

In line with this thesis three methods for cutting into soft tissue models for virtual surgery have been presented and prototypes have been implemented. A new hybrid cutting method for volumetric tetrahedral meshed mass-spring models was introduced. The technique combines two cutting methods, the progressive subdivision and non-progressive subdivision, into one technique. Moreover, an implementation of snapping was introduced for the non-progressive cutting method. This was integrated with the non-progressive cutting part of the hybrid approach. For snapping the set of minimal element subdivision patterns have been extended with six additional snapping patterns. The standard non-progressive minimal element subdivision method and the hybrid method with different snapping threshold where compared and evaluated against the other methods. Furthermore, the state machine framework of Bielser et al. [4] for progressive cutting via symmetric subdivision was implemented and enhanced by shortcut transitions, precalculation of optimal state transition sequences, and optimization of temporary subdivision handling. In the previous chapter the enhanced state machine framework was tested against the original one. Due to the high amount of tetrahedra that are created with the state machine that uses symmetric subdivision, a state machine framework was integrated with minimal element subdivision patterns, which was also used for the progressive cutting part of the hybrid cutting.

We evaluated the methods described in Chapter 3 on page 29 on the basis of the fundamental question how the different cutting algorithms influence the mesh complexity, the mesh quality, the stability of the soft tissue simulation, their effectiveness, and how the methods perform visually. Since the mesh complexity grows proportionally with the amount of elements in the mesh, the number of elements yield a criterion for complexity. The mesh quality, which is altered during cutting, was measured according to a local and a global criterion. As local criterion the shape of a single tetrahedron was evaluated by the maximal edge aspect ratio quotient ($maxAR$) of the shortest and longest edge of a tetrahedra. The distribution of the $maxAR$ values before and after a cut was compared for all methods. Since the different methods create a different amount of tetrahedra during cutting, a value independent from the element count was introduced, in order to get a relative and competitive result for the different results. For this reason, an experiment that determines the average mean of the $maxAR$ for all tetrahedra in the model was conducted. The global mesh criterion was defined as the relation of the tetrahedral volume sizes. Measurements were conducted on basis of the absolute distribution of all elements in the model according to their volumes. A relative difference between the volume sizes was determined too. For this, the average volume deviation of $avgD$ over all tetrahedra in relation to the average volume size was defined. The $avgD$ is also independent from the number of elements in the model.

5.2 Complexity

The experiment concerning the mesh complexity generally shows that cutting with subdivision methods leads to a rapid increase of elements in the model. Hybrid cutting in comparison to the ordinary symmetric subdivision approach, reduces the model's complexity severely. This clearly lies in the nature of the applied subdivision patterns in both methods. The shortcut improvement for the symmetric approach leads to an advancement of the ordinary method. However, it is a small improvement compared to the result of the hybrid method. Moreover, the the rate on which shortcut transition are applied, depends on the cut's velocity and how coarse or fine the mesh refinement is. Thus, the margin of improvement tends to vary with a change in those parameter. Another aspect of snapping is revealed. Snapping does not only promotes the simulation stability, it also decreases the mesh complexity. Furthermore, hybrid cutting, which generates the impression of a continuous and smooth cut, also benefits from snapping and has a remotely worse results than the visually unappealing non-progressive method. Nevertheless, one has to keep in mind that snapping alters the shape of adjacent tetrahedra. This comes from the adjustment of the position of a vertex. Altering the position of a tetrahedrons vertex also changes the shape of all adjacent tetrahedra. This in turn may create new degenerated tetrahedra. Thus, the threshold at which an vertex is snapped to an edge intersection must be chosen carefully. In conclusion, it can be reasoned that hybrid cutting beneficial for performance gain, because it it results in less complexity than the other progressive cutting methods. The small forfeit in complexity in comparison to the standard minimal element approach, in the cases where snapping is applied, is acceptable. It is trade- off for a much more realistic visual feedback, since the lack-behind effect of non-progressive cutting remains hidden concealed to the user.

5.3 Mesh Quality and Simulation Stability

From the experiments of the mesh quality, regarding both the local and global mesh criterion, follows that all cutting methods with minimal element subdivision, including our hybrid technique, retain a better mesh quality than both symmetric approaches. Here again the shortcut improvement shows a slightly better result than the ordinary method. Consequently, the hybrid approach and the implementation of progressive minimal element cutting via the state machine framework are much more suited for cutting in mass spring systems than the symmetric approaches, since a good mesh quality is essential for a stable deformation simulation.

As expected and already shown by other researchers [13][8], snapping improves the mesh quality, because it contributes to the prevention of degenerated elements. Snapped non-progressive minimal element cutting has the best result concerning the volume distribution, average volume deviation, $maxAR$ distribution, and the averaged $maxAR$. However, the result is closely followed by the outcomes of the hybrid cutting. Here as well the more realistic feedback compensates for this small difference.

In matters of simulation stability, we did investigations on the impact of progressive cutting on the mass spring model in respect to the temporary mesh quality during a cut. The non-progressive subdivision, the progressive subdivision and the hybrid approach were compared according to their average values of the $maxAR$ of all tetrahedra over the duration of a cut. When solely progressive cutting is applied abrupt fall-offs of the averaged $maxAR$ value into local minima occur, due to extreme shaped temporary subdivisions. Non-progressive cutting has a steady stepwise fall-off. The measurement pinpoints that the hybrid method reduces extremely shaped elements due to temporary subdivisions. Consequently, hybrid cutting, that has a similar result as progressive cutting in terms of the constant follow-up of the modeled incision to the tool's position, can avoid the simulation instabilities inflicted by degeneracies due to temporary subdivisions for the core part of the mesh. This way the overall stability of the mesh is increased.

5.4 Efficiency

Hybrid cutting proves to be most efficient when applied voluminous and homogeneous meshed models, since the ratio of core tetrahedra and tetrahedra that share a point with the model's surface is the crucial factor. Being able cut fractions of the mesh without storing temporary subdivision information, updating and deleting elements promotes performance and thus, improves interactivity of the simulation.
In conclusion, the hybrid approach shows quite promising results. In comparison to progressive cutting it leads to an increase in the stability in the mass spring simulation, especially because it is possible to avoid the destabilisation effect of temporary subdivisions. Moreover the interactivity is increased, by reducing the cut complexity.

5.5 Future work

With the hybrid approach we presented a promising cutting technique that enables a user to define a cut path freely and interactively into a soft tissue model. The method emphasis on maintaining soft tissue simulation stability without compromising the realism of a steady-going cut simulation. However, there are still areas within this topic, especially regarding the realism and performance of the cutting procedure, which need to be improved and developed further.

5.5.1 Stability

Snapping proved to be quite valuable for maintaining soft tissue simulation stability. The stability of progressive cutting can be further improved by integrating snapping into the state machine for the progressive cutting part of the hybrid approach too. This would raise the complexity of the state machine graph, but would in turn support the simulation stability greatly. Moreover, it might be possible to avoid degeneracies due to temporary subdivision by introducing the stability criteria of Section **??** on page ?? to the state machine. A tetrahedron with a temporary subdivision would be permitted to perform a state transition only if the next states subdivision is stable. In other words, progressive subdivision is restrained as long the sub-tetrahedra, which are supposed to be inserted, would be smaller than a predefined stability threshold. This would work for the case where the tool has just cut an edge or face in the time step shortly before-

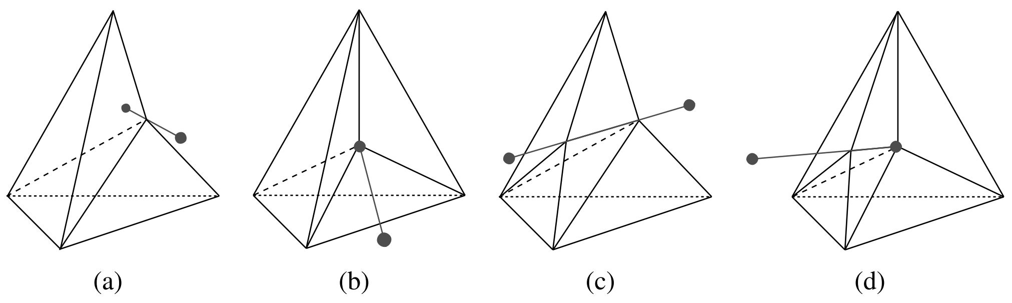


Figure 5.1: Deformation patterns: geometrical deformation of (a) one edge , (b) one face, (c) two edges, and one face and on edge(d)

hand. For cases where the tool is about to cut another primitive in the element another solution must be found. One possibility is to restrain the progressive adjustments of the tool's ENP and EXP (entry point and exit point) until the sub-elements fulfill the quality requirements.

In addition, node snapping could be enhanced by the usage of a dynamic threshold, which takes the edge length of adjacent tetrahedra into account. This would avoid the creation of degeneracies by altering the shape of adjacent tetrahedra due to snapping.

5.5.2 Deformation Patterns

In order to simulate the physical behaviour of the tool's and soft tissue model's interaction, it is necessary to simulate the temporary topological changes that are applied to the model by deformation. Since a surgery tool is not indefinitely sharp the user should be required to exert a certain level of force with the cutting tool to the model until it can be penetrated by the tool. It is required to generate a collision response and to propagate the applied forces properly into the model. For this reason, we suggest four deformation patterns that create new vertices were the tool comes in contact with the model. The new points are displaced with the movement of the tool and they serve for the centre of the deformation. The deformation patterns are depicted in Figure 5.1.

Moreover, we recommend to integrate these four deformation patterns into the state machine as explained in the following. The state machine should be extended by the additional deformation states in order to adapt the state machine approach for cutting with deformation response. One possibility is outlined in Figure 5.2 on the following page. Several state transitions are suggested that model the change-over from the geometrical modeling of the deformation to incision modeling. Deformation states

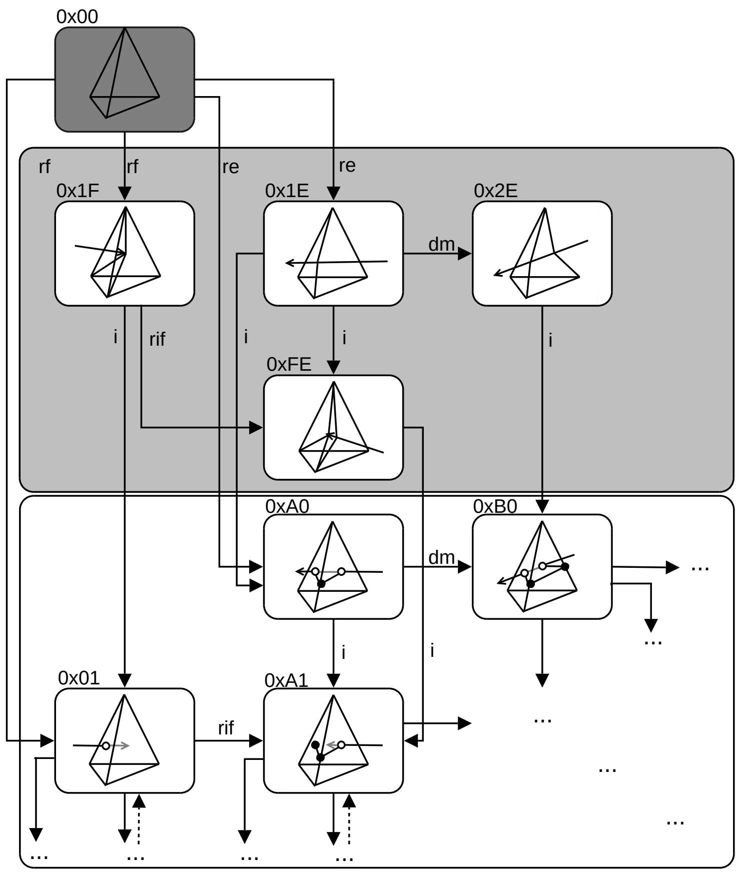


Figure 5.2: The extended conceptual view of the state machine with the four deformation patterns.

(light gray background) are used in order to model local deformations in the environment, where the cutting tool exerts force to the model without exceeding a certain force threshold. Exceeding the threshold would causes a tetrahedra to change from a deformation state into a state that models the incision of a cut (white background). Consequently, the determination which state is assigned to a tetrahedron would not only depend on the number of intersection occurrences and their distribution, but also on the force that is applied to the vertices by the cutting tool .

5.5.3 Performance

In order to enhance the performance of the cutting algorithm the rough collision detection between cutting tool and model has to be replaced by a more sophisticated one. Since the intersection detection is one of the computational most expensive parts a subset of tetrahedra that is as small as possible should be tested for it within every time step. This will speed up the performance of the cutting module within the simulator considerably.

Bibliography

[1] D. Bielser, V. Maiwald, and M. Gross. Interactive cuts through 3-dimensional soft tissue, 1999.

[2] D. Bielser and M. Gross. Interactive simulation of surgical cuts. In *PG '00: Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*, page 116, Washington, DC, USA, 2000. IEEE Computer Society.

[3] F.P. Brooks. What's real about virtual reality? *IEEE Comput. Graph. Appl.*, 19(6):16–27, November 1999.

[4] D. Bielser, P. Glardon, M. Teschner, and M. Gross. A state machine for real-time cutting of tetrahedral meshes. *Graph. Models*, 66(6):398–417, 2004.

[5] A. Mor and T. Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2000.*, volume 1935, pages 598–607. Springer-Verlag, October 2000.

[6] A. Mazura, S. Seifert, U. Rembold, and R. Dillmann. Virtual cutting in medical data, in proc. medicine meets virtual reality, 1997.

[7] Fabio Ganovelli, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. A multiresolution model for soft objects supporting interactive cuts and lacerations. *Comput. Graph. Forum*, 19(3), 2000.

[8] D. Steinemann, M. Harders, M. Gross, and G. Szekely. Hybrid cutting of deformable solids. In *VR '06: Proceedings of the IEEE Virtual Reality Conference (VR 2006)*, page 5, Washington, DC, USA, 2006. IEEE Computer Society.

[9] M. Nakao, T. Kuroda, H. Oyama, M. Komori, T. Matsuda, and T. Takahashi. Combining volumetric soft tissue cuts for interventional surgery simulation,

miccai '02: Proceedings of the 5th international conference on medical image computing and computer-assisted intervention-part ii. pages 178–185, London, UK, 2002. Springer-Verlag.

[10] C. Bruyns, S. Senger, A. Menon, K. Montgomery, S. Wildermuth, and R. Boyle. A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *Journal of Visualization and Computer Animation*, 13(1):21–42, 2002.

[11] D. Bielser and M. Gross. Open surgery simulation, 2002.

[12] Wen Wu and Peng Ann Heng. An improved scheme of an interactive finite element model for 3d soft-tissue cutting and deformations, springer-verlag, 1998.

[13] H.-W. Nienhuys. Cutting in deformable objects. phd thesis, utrecht university, 2003.

[14] H.-W. Nienhuys and A.F. van der Stappen. A Delaunay approach to interactive cutting in triangulated surfaces. In *Fifth International Workshop on Algorithmic Foundations of Robotics*, Advanced Robotics, page ? Springer Verlag, 2003.

[15] D. Serby, M. Harders, and G. Szekely. A new approach to cutting into finite element models. In *MICCAI '01: Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 425–433, London, UK, 2001. Springer-Verlag.

[16] F. Ganovelli and C. O'Sullivan. Animating cuts with on-the-fly re-meshing, 2001.

[17] Y.-J. Lim and S. De. On the use of meshfree methods and a geometry based surgical cutting in multimodal medical simulations. *haptics*, 00:295–301, 2004.

[18] S. Suzuki, N. Suzuki, A. Hattori, A. Uchiyama, and S. Kobayashi. Sphere-filled organ model for virtual surgery system, ieee transactions on medical imaging, volume 23, number 6, june 2004, pp. 714-722, 2004.

[19] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer*, 16(7):437–452, 2000.

[20] M. LeDuc, S. Payandeh, and J. Dill. Toward modeling of a suturing task, 2003.

[21] J. Brown, K. Montgomery, J.-C. Latombe, and M. Stephanides. A microsurgery simulation system. *Lecture Notes in Computer Science*, 2208, 2001.

[22] H. Zhang, S. Payandeh, and J. Dill. On cutting and dissection of virtual deformable objects, robotics and automation, 2004. proceedings. icra '04. 2004 ieee international conference, p.3908- 3913 vol.4. In *MICCAI '02: Proceedings of the 5th International Conference on Medical Image Computing and Computer-Assisted Intervention-Part II*, pages 178–185, London, UK, 2004. Springer-Verlag.

[23] H. W. Nienhuys and A. F. van der Stappen. Supporting cuts and finite element deformation in interactive surgery simulation, 2001.

[24] A. Al-khalifah and D. J. Roberts. Survey of modeling approaches for medical simulators, international conference series on disability, virtual reality and associated technologies, oxford, uk, pp. 321-329, 2004.

[25] F. Mazzella, J.-C. Latombe, and K. Montgomery. Auto aquisation of elastic properties of soft tissues for surgical simulation, standford-nasa biocomputation center, standford, 1999.

[26] C. Bruyns and K. Montgomery. Generalized interactions using virtual tools within the spring framework: Cutting, 2002.

[27] H. Zhang, S. Payandeh, and J. Dill. Simulation of progressive cutting on surface mesh model.

[28] Kevin Chun-Ho Wong, Yu-Hang Siu Siu, Pheng-Ann Heng, and Hanqiu Sun. Interactive volume cutting. In *Graphics Interface*, pages 99–106, 1998.

[29] R. Dillmann T. Salb, T. Weyrich. Preoperative planning and training simulation for risk reducing surgery. international training and simulation conference (itec), the hague, the netherlands. 1999.

[30] E. Keeve, S. Girod, and B. Girod. Computer-aided craniofacial surgery, computer assisted radiology car, 1996.

[31] G. Song and N. Reddy. Towards virtual reality of cutting: A feasibility study, proceedings of 16th annual international conference of the ieee engineering in medicine and biology society, baltimore, md, 1994.

[32] C. Bruyns and S. Senger. Interactive cutting of 3d surface meshes. *Computers & Graphics*, 25(4):635–642, 2001.

[33] C. Bruyns and K. Montgomery. Generalized interactions using virtual tools within the spring framework: Probing, 2002.

[34] C. Forest, H. Delingette, and N. Ayache. Removing tetrahedra from a manifold mesh, in proc. ieee computer animation, 2002.

[35] C. Forest, H. Delingette, and N. Ayache. Cutting simulation of manifold volumetric meshes, 2002.

[36] C. Basdogan, C. Ho, and M. Srinivasan. Simulation of tissue cutting and bleeding for laparoscopic surgery using auxiliary surfaces, 1999.

[37] H.-W. Nienhuys and A.F. van der Stappen. Combining finite element deformation with cutting for surgery simulations. In A. de Sousa and J.C. Torres, editors, *EuroGraphics Short Presentations*, pages 43–52, 2000.

[38] H.-W. Nienhuys and A.F. van der Stappen. A surgery simulation supporting cuts and finite element deformation. In *MICCAI '01: Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 145–152, London, UK, 2001. Springer-Verlag.

[39] A. Mor. Progressive cutting with minimal new element creation of soft tissue models for interactive surgical simulation. phd thesis, robotics institute, carnegie mellon university, pittsburgh, pa, 2001.

[40] F. Ganovelli, P. Cignoni, C. Montani, and R. Scopigno. Enabling cuts on multiresolution representation. *The Visual Computer*, 17(5):274–286, 2001.

[41] M. Bro-Nielsen, D. Helfrick, B. Glass, X. Zheng, and H. Connacher. Vr simulation of abdominal trauma surgery. *Medicine Meets Virtual Reality 6*, pages 117–123, January 1998.

[42] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, December 2006.

[43] H. Delignette. Towards realistic soft tissue modeling in medical simulation, ieee proceedings: Special issue on surgery simulation, pp. 512-523. 1998.

[44] M. Koch, S. H. M. Roth, M. H. Gross, A. P. Zimmermann, and H. F. Sailer. A framework for facial surgery simulation. Technical Report 326, 1999.

[45] S.F. Gibson. 3d chainmail: A fast algorithm for deforming volumetric objects. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 149–ff., New York, NY, USA, 1997. ACM Press.

[46] S.F. Gibson, J. Samosky, A. Mor, C. Fyock, W.E.L. Grimson, T. Kanade, R. Kikinis, H.C. Lauer, N. McKenzie, S. Nakajima, T. Ohkami, R. Osborne, and A. Sawada. Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback. In *CVRMed*, pages 369–378, 1997.

[47] S.F. Gibson, C. Fyock, E. Grimson, T. Kanade, R. Kikinis, H. Lauer, N. McKenzie, A. Mor, S. Nakajima, H. Ohkami, R. Osborne, J. Samosky, and A. Sawada. Simulating surgery using volumetric object representations, real-time volume rendering and haptic feedback.

[48] J. Boehm. Models and methods for mass spring systems in surgery simulation. Master's thesis, 2006.

[49] M. Eitz. Realtime soft tissue simulation employing constraint based particle systems and hierarchical spatial hashing. Master's thesis, 2006.

[50] F.S.Jr. Hill. The pleasures of 'perp dot' products, ch. ii.5 in graphics gems iv (ed. p. s. heckbert). san diego: Academic press, pp. 138-148, 1994.

致　谢

First of all, I would like to express my gratitude to my supervisor Prof. Gu Lixu for constantly supporting my work and helping me to improve it. Moreover, he gave me the great opportunity to work in the resourceful environment of the Digital Image Processing Laboratory, which has been inspiring and fruitful. Among the members of the lab, who own my gratitude, I would like to especially mention Zhang Jingsi, Zhang Shaoting and Huang Pengfei for always being at my side as team mates. They made the months in the lab a wonderful time and this work much more easier. Also, I would like to thank my German team mates, Jan Boehm and Matthias Eitz, who helped me on countless occasions and supported me with valuable feedback. My stay in China which has been an extraordinary opportunity was possible due to the efforts of Prof. Sheng Huanye and Prof. Hommel who founded the Double Degree Program in which I participated in. Moreover, I would like to thank the DAAD which supported me for a year by means of a scholarship. In addition, I would like to thank my family and friends for their appreciation and help. In the process of writing this thesis I would like to especially mention my mother Marika Dworzak and my brothers Stefan and Matthias for their support. Matthias granted me asylum in his office, when I couldn't concentrate at home and assisted me with his constant encouragement and feedback. I am also very grateful for the excellent feedback and motivation family Schaback provided me with. Last, but not least, I would like to thank Johannes for his great support and for putting up with me all that time.

在学期间的研究成果及发表的论文

在国际和国内学术会议上发表的论文

[1] J. Dworzak, L. Gu, Combining progressive and non-progressive cutting for soft tissue surgery simulations, *in Proceedings of the 21th Computer Assisted Radiology and Surgery (CARS' 07)*, Berlin, Germany, June 27, 2007

Index