Accurate Real-Time Framework for Complex Pre-defined Cuts in Finite Element Modeling

By Tong Xin

This thesis is submitted for the degree of
Doctor of Philosophy

School of Computing
Newcastle University

November 2020

Abstract

Achieving detailed pre-defined cuts on deformable materials is vitally pivotal for many commercial applications, such as cutting scenes in games and vandalism effects in virtual movies. In these types of applications, the majority of resources are allocated to achieve high-fidelity representations of materials and the virtual environments. In the case of limited computing resources, it is challenging to achieve a convincing cutting effect. On the premise of sacrificing realism effects or computational cost, a considerable amount of research work has been carried out, but the best solution that can be compatible with both cases has not yet been identified.

This doctoral dissertation is dedicated to developing a unique framework for representing pre-defined cuts of deformable surface models, which can achieve real-time, detailed cutting while maintaining the realistic physical behaviours. In order to achieve this goal, we have made in-depth explorations from geometric and numerical perspectives. From a *geometric* perspective, we propose a robust subdivision mechanism that allows users to make arbitrary predetermined cuts on elastic surface models based on the finite element method (FEM). Specifically, after the user separates the elements in an arbitrary manner (i.e., linear or non-linear) on the topological mesh, we then optimise the resulting mesh by regenerating the triangulation within the element based on the Delaunay triangulation principle. The optimisation of regenerated triangles, as a process of refining the ill-shaped elements that have small *Aspect Ratio*, greatly improves the realism of physical behaviours and guarantees that the refinement process is balanced with real-time requirements.

The above subdivision mechanism can improve the visual effect of cutting, but it neglects the fact that elements cannot be perfectly cut through any pre-defined trajectories. The number of ill-shaped elements generated yield a significant impact on the optimisation time: a large number of ill-shaped elements will render the cutting slow or even collapse, and vice versa. Our idea is based on the core observation that the producing of ill-shaped elements is largely attributed to the condition number of the global stiffness matrix. Practically, for a stiffness matrix, a large condition number means that it is almost singular, and the calculation of its inverse or the solution of a system of linear equations are prone to large numerical errors and time-consuming. It motivates us to alleviate the impact of condition number of the global stiffness matrix from the *numerical* aspects. Specifically, we address this issue in a novel manner by converting the global stiffness matrix into the form of a covariance matrix, in which the number of conditions of the matrix can be reduced by exploiting appropriate matrix normalisation to the eigenvalues.

Furthermore, we investigated the efficiency of two different scenarios: an exact square-root normalisation and its approximation based on the Newton-Schulz iteration.

Experimental tests of the proposed framework demonstrate that it can successfully reproduce competitive visuals of detailed pre-defined cuts compared with the state-of-the-art method (Manteaux et al. 2015) while obtaining a significant improvement on the FPS, increasing up to 46.49 FPS and 21.93 FPS during and after the cuts, respectively. Also, the new refinement method can stably maintain the average *Aspect Ratio* of the model mesh after the cuts at less than 3 and the average *Area Ratio* around 3%. Besides, the proposed two matrix normalisation strategies, including ES-CGM and AS-CGM, have shown the superiority of time efficiency compared with the baseline method (Xin et al. 2018). Specifically, the ES-CGM and AS-CGM methods obtained 5 FPS and 10 FPS higher than the baseline method, respectively.These experimental results strongly support our conclusion which is that this new framework would provide significant benefits when implemented for achieving detailed pre-defined cuts at a real-time rate.

I would like to dedicate this thesis to my loving family,
especially to memory of my grandmother, BaoZhen Yu.

Declaration

I declare that this dissertation is an original report of my doctoral research except where specific reference is made to the work of others. This dissertation is written by me and have not been submitted in whole or in part for any previous degree.

By Tong Xin
November 2020

Acknowledgements

I would like to express my deep gratitude to the people who have contributed in various ways to the completion of this dissertation.

First and foremost, I thanks my supervisors, Dr Graham Morgan and Dr Gary Ushaw, for their patient guidance, enthusiastic encouragement and useful critiques of this research work. Their guidance helped me in all the time of research and writing of this dissertation.

Many thanks to my colleagues who have helped in various ways during my Ph.D studies. Thank you all for your accompany and encouragement in my many moments of crisis. Your friendship makes my life a wonderful experience.

I would like to thank my loving grandparents, Guilan Li and Hongshun Ren, for their love and encouragement in all my endeavours and for the value they placed on family. I also would like to thank my family for their endless and unconditional love, in particular, my parents, Xiaoqing Ren, Huadong Xin; my parents in law, Yuxian Li and Shuping Wang, and my little son, Zixiao Wang who is five years old now, as well as my husband, Shidong Wang and all of whom have made my accomplishments possible.

Table of contents

List of figures

List of tables

List of Symbols

E Young's modulus.

F The applied external forces.

$F_{internal}$ The global internal forces generated at all nodes in a mesh.

F_m The external forces applied on a point mass.

ΔL Deformation.

Ω The problem domain.

$\boldsymbol{F}$ The deformation gradient.

$\boldsymbol{U}$ Displacement field.

$\boldsymbol{X}$ Node position vector in an reference configuration (material coordinate).

$\boldsymbol{x}$ Node position vector in current configuration (spatial coordinate).

$\mathbf{C}$ The global damping matrix.

$\mathbf{K}$ The global stiffness matrix.

$\mathbf{K}_e$ The element stiffness matrix.

$\mathbf{M}$ The global mass matrix.

$\mathbf{R}$ The global rotation matrix.

σ Stress generated in a continuum body.

ε Strain caused by non-rigid deformations.

$\{x, y, z\}$ Cartesian coordinate system.

$f_{internal}$ The internal forces generated at all nodes of an element.

m The mass of a point mass.

Chapter 1. Introduction

Physically-based simulations that allow cutting bring significant benefits in the training of doctors to perform surgical operations and in representing damage scenes in commercial applications including video games and the computer animations in movies. For example a damage scene may correspond to sword cutting cloth or arms during a fight. This is due to the fact that it:

- reduces the cost and time for the training of surgical operations (McCloy & Stone 2001). Traditionally, performing a surgical operation is expensive especially when the doctors lack experiences. This inexperience causes longer operation time and increases the possibility of the doctor making a critical mistake. One can use other alternatives rather than real human bodies. However this still brings several disadvantages. Plastic models cannot reflect the mechanical properties of living organs and the use of animals can be expensive as well. In addition books and videos which describe and demonstrate surgical operations do not provide the necessary hands-on interactive experience. In such situations performing the surgical operation on physically-based virtual human bodies which are simulated by computers is necessary for the training of the surgical operations. Such technology is commonly termed *virtual surgery* (Gillio 1999). *Virtual surgery* allows the application of complicated scenarios on the simulated models without requiring expensive resources. In addition, such technology gives the trainees the space to make mistakes without any serious consequences.

- increases the visual realism of physically-based models which are commonly used in video games and 3D movies. A high-performance video game may allow the skin or cloth to be cut when characters are fighting, and similarly in the virtual movies should support cutting to achieve a high fidelity representation of the virtual environment. In such situations models that do not react appropriately when cuts are applied can significantly diminish the realism of the simulation and hence spoil the user experience. Therefore realistic cutting simulations are crucial in damage scenes in video games and 3D movies.

1.1 Motivations

A cutting technology that can be implemented in both virtual surgery and other commercial applications such as video games and 3D movies is difficult to develop. This is due to the fact

that these two areas have different requirements to meet. Traditionally the cutting technologies applied in virtual surgery need to interactively represent the cut trajectory which is drawn by users onto the simulated organ. In mesh-based modelling methods, such simulated organs are modelled by a finite number of geometric primitives, which are also called *elements*. For instance, the skin of a simulated organ is composed of triangular elements. In this situation a minimal set of sub-triangles of each intersected triangular element should be generated to reduce the subsequent computational cost but such a refinement is not adequate when the incisions caused by cuts have arbitrarily-shaped boundaries. In addition, most of the recent research in virtual surgery has focused on increasing the realism of the simulated deformable material models (Zhang et al. 2017). However models that do not represent the incisions accurately significantly diminish a simulation's realism.

In contrast, commercial applications which involve cutting always aim to represent the arbitrarily-shaped cut incisions accurately in complex cutting scenarios (e.g., the ability to handle multiple cut incisions when cutting folded material models). Unfortunately, such effects are simply too expensive under a real-time requirement or at an interactive rate since it requires more resources to be dedicated to both the cutting and the material simulation. In this situation interactive cutting which allows complex cutting cannot be achieved without a noticeable delay.

Currently, achieving complex cuts defined by the users on physically accurate Finite Element Method models in real-time is challenging. In such a technique the cuts should look realistic and be reproduced at a real-time rate which is that the lowest Frame Per Second (FPS) during the cut should be at least 30 FPS (Gregory 2017). This is the problem that will be addressed in this thesis. Such cutting framework provides a real-time approach using accurate material models that can operate on the hardware which employs standard GPU techniques.

1.2 Challenges

Achieving a real-time cutting method that allows complex cutting using realistic deformable models is challenging as several difficult tasks need to be handled simultaneously. These tasks include the simulation of realistic deformable materials, the handling of complex cutting scenarios (including an arbitrarily-shaped cut, re-cut, cross cut, sectioning and overlapping cut), and satisfactory real-time performance. We will discuss each task in turn in this section and describe the appropriate assumptions made for our experimental work.

1.2.1 Physically-based Modelling

Producing a physically-based model which allows cutting is a challenging task. This is due to the fact that such physical behaviours increase the computational cost. Typically heuristic deformable models are used in cutting simulations to satisfy the real-time requirement (Meier et al. 2005, Cueto & Chinesta 2014). The deformations of such models are governed by manually changing the parameters of the heuristics by the developers rather than the motion

equation whose changes follows the law of physics. In this manner the realism of the deformations in heuristic deformable models always relies on the observation and analysis of the reference deformable materials in the real world. Moreover when forces are applied such heuristic deformable models cannot react appropriately unless the developers pre-describe the response. Therefore it cannot provide physically accurate response in a real-time interactive simulation.

The lack of realism in heuristic deformable models has motivated the development of physically-based deformable models (also known as mechanical models). The deformations of such models are described by the laws of physics and can be represented mathematically in the form of partial differential equations. Ideally such equations would have an analytical solution. However to find such an analytic solution the equations often need to be considerably simplified by making a number of assumptions which are usually not valid in the real world. Hence the partial differential equations are usually solved numerically. There are many techniques available to find such an approximation, e.g., Finite Element Method, Finite Element Difference, and Boundary Element Method. The general idea behind many of these methods is to discretise the problem domain into a *mesh* which consists a finite number of elements and apply a mechanical governing equation with specific assumptions to each element. Thus the solution for the whole problem domain can be approximately obtained by assembling the solutions of elements which are iteratively computed.

The FEM has been successfully applied in several physically-based models such as elasticity. This thesis first investigates cutting on FEM-based deformable surface model discretised by a finite number of triangular elements.

1.2.2 Complex Cutting

A realistic representation of cuts is very challenging to achieve as a variety of cutting scenarios need to be handled. Imaging that the users apply a cut to a surface material model using a simulated blade. This blade can reach the model anywhere and give an arbitrarily-shaped cut trajectory. Moreover the simulated material will change over time so that the blade may intersect with a curved boundary or multiple places of the material. The simulated material may also need to allow several cuts even at the same place without losing volume or numerical stability. Therefore a reliable cut algorithm is required to handle complex cutting scenarios.

For FEM-based models, representing accurate cuts in all of the cutting scenarios mentioned above does require elements to be altered or introduced. If a cut is to be accurately presented exactly where a user needs it then the elements around this area must be remeshed. Usually this means that new elements need to be introduced to represent the desired shape of the incision caused by the cut. If this was not the case and the approach was restricted to simply removing elements then this would have the effect, in FEM, of removing material mass (undesired) while restricting cuts along existing element edges (not where intended). Therefore accurate cutting in FEM means that the first step has to determine which elements are needed to be remeshed. If the

shape of newly generated elements near the incision changes significantly, their physical behaviors will become inaccurate and affect the numerical stability of the simulation. Such elements are also known as *ill-shaped elements*. The intention is to use an increased number of smaller similarly shaped elements to better reflect a cut's position while still maintaining the nearby physical properties.

Handling complex cutting scenarios while avoiding ill-shaped elements is non-trivial in cutting simulations using FEM models. In the Finite Element Method, the governing equations of the simulated material are reformatted into matrix form. The global stiffness matrix stores the coefficient of each node in the model, which reflects the degree of force in all directions. The condition of the global stiffness matrix reflects how a function changes according to small changes in its inputs. The condition number can be calculated by the ratio of the magnitude of the largest and smallest eigenvalue.

In our simulation, ill-shaped elements are those triangular elements that significantly make the condition number of the global stiffness matrix high and therefore decrease the time efficiency and numerical stability of the simulation. The reason why this is important relates to the way FEM works. The solution of the whole problem domain is transferred into solving a system of equations which are determined by assembling each element equation using a numerical approach. Such numerical approaches usually adopt an iterative method to obtain the optimal solution of the system of equations in the form of $A\mathbf{x} = \mathbf{b}$; however ill-shaped elements can make the condition number of the global stiffness matrix A high (Shewchuk 2002). Solving the linear system with such an ill-conditioned coefficient matrix is slow due to a low convergence speed caused by numerical instability. (*i.e.* for a system of linear equations the independence between each equation decides the robustness of the numerical calculation. This is why avoiding ill-shaped elements when re-meshing the simulated material is one of the contributions of this thesis.

1.2.3 Real-time Simulation

Graphical simulations should not only provide scenarios that look realistic but also act and react realistically to user control. After applying a cut all calculations due to that cut have to be finished within milliseconds to avoid visual delays. For instance, the generally accepted lowest FPS is 30 FPS in real-time simulations which means only 33.33 ms can be used to simulate a single frame. In this case, all the calculations need to finish within 33.33 ms. Therefore a framework which can achieve the cuts in real-time while ensuring physically-based accurate behaviours is required to produce a realistic cutting simulation.

High fidelity cutting tends to require more resources to be dedicated to the cutting and the object simulation themselves, usually at the expense of other simulated activities or via dedicated/costly hardware which is unaffordable in many practical applications. Research into cutting for real-time simulation usually makes a compromise between the realism (does the cut look as expected and the model respond appropriately) and the performance (can the cut be

achieved when requested without noticeable delay). Therefore, an active area of research is the ability to allow advanced object models to be accurately cut while lowering the demand for computational resources. This is another significant contribution of the thesis, introducing a new framework of cutting which can reproduce competitive visuals of detailed pre-defined cuts while operating five times faster than the state-of-the-art method (Manteaux et al. 2015) as well as 1.7 times faster than the baseline method (Xin et al. 2018).

In real-time simulations the equation needs to be solved in a given time step. A time step which is too small would provide insufficient time to achieve the desired mathematical accuracy and ultimately would cause the material model to fail. The more easily deformable a material model is regarding cutting the more this compounds the difficulty of ensuring real-time and physical accuracy. That is certainly true when allowing re-cutting as mesh elements which have already been altered or are newly introduced must be re-altered and managed while still maintaining real-time behavioural properties. As the number of mesh elements grows (more cuttings) the computational expense increases and the possibility of ill-shaped elements also increases. This is when existing approaches tend to have their greatest difficulty and many do not allow re-cutting especially in real-time scenarios.

1.3 Research Assumptions

This research is built on the following assumptions:

- The numerical stability of a system of finite element equations can be maintained through improving the shape and size of ill-shaped elements.
- The time efficiency of solving a system of linear finite element equations can be improved by reducing the condition number of the global stiffness matrix of the system.

1.4 Thesis Contribution

This thesis develops a framework for achieving detailed cuts defined by users in real-time on Finite Element Method (FEM) based deformable surface models. We propose a general cutting technique applicable to a high-performance material model that can be simulated in real-time. Our method is capable of handling complex cuts which include smooth arbitrarily-shaped cuts, re-cuts, cross cuts, partial cuts, cut out a part of the material model and overlapping cuts in a 3D environment. This is achieved while still maintaining the physical accuracy of the deformable surface material model during its dynamic deformations.

The contributions of this thesis can be summarised as:

- A real-time cutting method which can accurately represent cuts pre-defined by users at a real-time rate which means that the lowest FPS during the cut is on average higher than 30 FPS.

- A new refinement method that keeps the average *Aspect Ratio* of the elements at higher than 3 and the average *Area Ratio* steadily around 3% after applying cuts. Therefore the deformable material model is physically accurate in the following dynamic deformations.
- Two matrix optimisation strategies that utilise square root matrix normalisation methods to keep the condition number of the global stiffness matrix low (a.k.a., well-conditioned) after the cuts to improve the time efficiency of the physics solver used to solve the system of finite element equations.

1.5 Publication

Accurate Real-Time Complex Cutting in Finite Element Modeling Xin T, Marris P, Mihut A, Ushaw G, Morgan G. 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP 2018 (Pages 183-190).

A paper is ready to submit to *Graphical Model* journal, mainly based on the technology which optimises the global stiffness matrix after the cuts to improve the time efficiency of physics solver. This technology will be presented in Section 3.5.

1.6 Thesis Outline

This thesis is organised into six chapters. **Chapter 1** describes the motivations behind this doctoral research and the research hypotheses. **Chapter 2** first introduces the physics and mathematical background corresponding to the field of physically-based simulations. It also provides related research and applications to contextualise the contribution of this thesis. **Chapter 3** presents a cutting method that is capable of handling complex cutting on physically accurate material models in real-time. **Chapter 4** provides the experiment plan and setting for the proposed framework. **Chapter 5** tests the time efficiency of the proposed framework in a variety of cutting scenarios and shows the visuals of reproduced complex cuts. Finally **Chapter 6** discusses the contribution of this thesis to extending the existing knowledge base and discusses potential areas for further work.

Chapter 2. Background and Related Works

This dissertation aims to provide a new technique which is capable of representing accurate cuts on physically accurate deformable surface material models. So this chapter will focus on introducing the representation of the physical behaviours of the deformable materials in the real world and the cutting techniques in computer graphics. The definition and the classification of deformable solid materials are first introduced (Section 2.1). This chapter then explains how the deformations of deformable materials are described and simulated in computer graphics (Section 2.2). Afterwards the significant physically-based modelling techniques in previous research will be reviewed and compared so that we are clear that the Finite Element Method can be used to reproduce the most accurate physically-based deformable behaviours (Section 2.3).

The second half of this chapter will introduce and discuss the cutting techniques in computer graphics. The mesh of a material model should be established before applying cuts. Such a mesh represents the topological structure of the material model. The meshes that are used to represent different types of the materials in the real world will be introduced in Section 2.4.

We then compare the surface models with the volumetric models aiming to identify the more suitable models to be used to simulate the tissue removal of surface materials in the real world (Section 2.5). Furthermore we introduce the steps of the cutting of 2D surface material models and discuss the challenges (Section 2.6). Finally we review the most significant related works to discuss how these challenges have been considered or resolved.

2.1 Deformation

Deformable solid materials may change their shape or size when any external force or temperature change is applied. Under an external force deformable solid objects can experience changes to their shapes such as extension, compression, shearing, bending, or twisting. When the deformation goes beyond a threshold it can result in material failure (also known as fracture). Temperature changes will mostly influence the physical state of the deformable solid objects. For example, a marshmallow can melt under a heat transform.

This thesis is only concerned with the simulation of deformable solid surface objects and the cuts applied to them. We would like to remind the reader that we do not discuss deformation at an atomic level. Instead, we will introduce the theories and technical backgrounds about

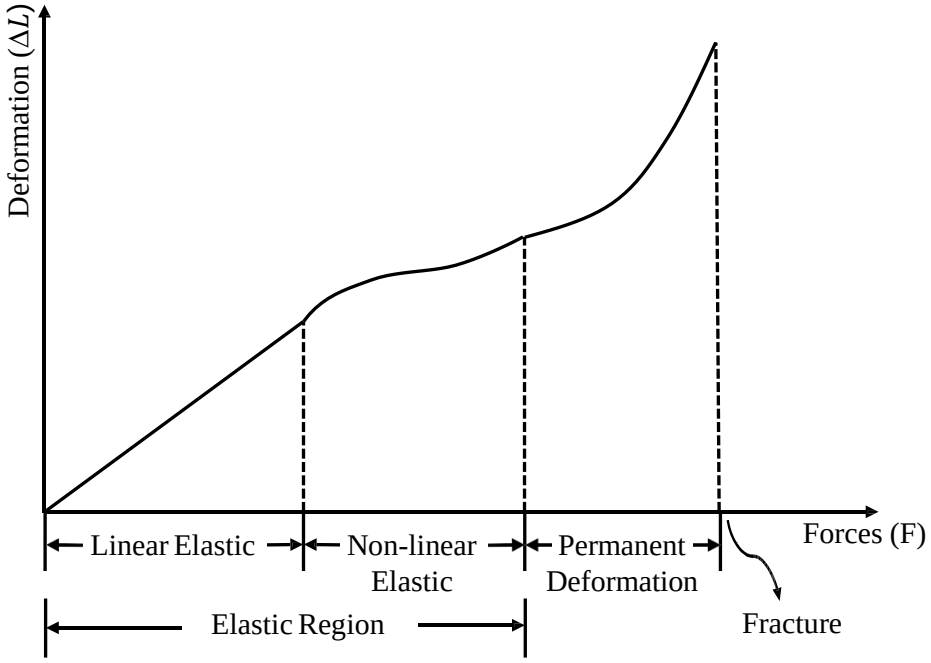


Fig. 2.1 A variety of deformations resulting from the applied forces to rubber (Sokolnikoff et al. 1956).

deformation from the field of continuum mechanics where the simulated objects (solid or fluid) are assumed as a continuous body rather than as discrete particles.

2.1.1 Definition

Deformation describes the change in the deformation of a continuous body. This means that a curve drawn in the initial body placement will normally change its lengths and position when the body is displaced. Deformations can be classified as shearing, extension, compression, rotation and rigid body translation. In particular a deformation is called a rigid body motion when it only involves rotation and rigid body translation.

2.1.2 Classification

Different levels of forces applied to an object can result in a variety of deformations depending on the material properties of the object. Figure 2.1 depicts the relationship between the deformation ΔL and the force F when it is applied to the object. When the force is not significant the internal forces that resist the applied force will arise. If these are strong enough to resist the applied force this allows the object to return to its original state after the applied force is removed. When the deformation is a reversible process the object is termed elastic. Compared with elastic deformation, where the energy expended in deformation is temporarily stored in the elastic strain and can entirely recover after the external load is removed, plastic deformation describes the permanent deformation that dissipates energy during such an irreversible process. Furthermore, at the end of the plasticity range, adding more applied forces can cause structural

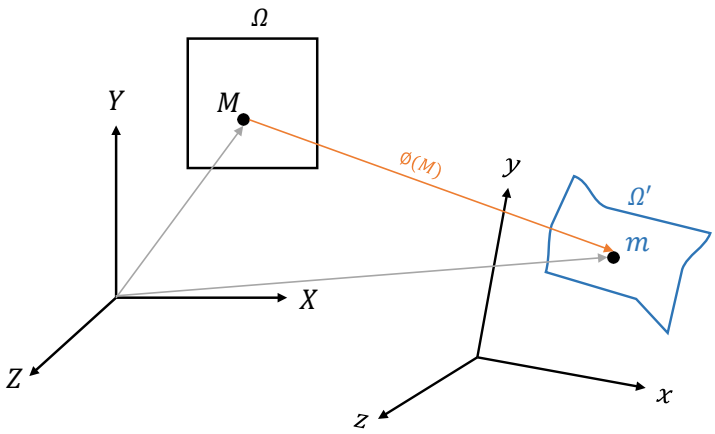


Fig. 2.2 In continuum mechanics, the deformation map $\phi(M)$ reflects the changes of a particle in a reference configuration to a current configuration. $\boldsymbol{M}$ and $\boldsymbol{m}$ represent the position of the particle in a reference configuration and a current configuration respectively.

failure (also known as fracture). After the fracture the material does not remain continuous anymore and fracture mechanics is needed to describe the dynamic behaviours, i.e., the dynamic of cracks propagation after the material failure (Anderson 2017).

In contrast to deformable objects, rigid objects cannot change their shapes under any applied external forces. Only translation and rotation will influence the dynamics of the rigid body when applying external forces. However, for deformable objects it is not just translation and rotation that need to be considered but the deformation in the body itself also needs to be taken into account when describing the dynamic behaviours.

2.2 Physically-based Deformation

2.2.1 Continuum Mechanics

Continuum mechanics construct a mathematical model of an object in the real world to describe its complex physical behaviours. This theory analyses an object at a macroscopic level, i.e. the object is treated as a continuous body rather than a group of discrete particles. A set of partial differential equations (PDEs) are employed to describe the macroscopic physical quantities, e.g. position, velocity. It is called solid mechanics when using the continuum mechanics theory to describe the physical behaviour of a solid object. In this section we will explain how the deformation of solid objects is described using continuum mechanics which is commonly used in computer graphics applications (Nealen et al. 2006).

Deformation in Continuum Mechanics is the transformation of a continuum body from a reference configuration to its current configuration (Liu 2013). A configuration (in a 2D or 3D coordinate system) assembles the positional information of all particles of the solid object. As shown in figure 2.2, when analysing deformation, the reference configuration describes the domain Ω in an undeformed state, in which the vector M represents the reference coordinates

(also known as *material* coordinates) of a particle in the continuum body. The current configuration is treated as a deformed configuration, in which m is called the spatial coordinate with respect to the current configuration.

The displacement field $\boldsymbol{U}$ reflects any transformation of the continuum body from the reference configuration to the current configuration, and it is calculated as:

$$\boldsymbol{m} - \boldsymbol{M} \tag{2.1}$$

2.2.2 Strains and Stresses

The displacement field $\boldsymbol{U}$ has two components which include the rigid-body displacement and the non-rigid deformation. The term rigid-body displacement implies that the displacement only concerns where the object moves to but does not result in any stretching or deforming inside the continuum body. However, non-rigid body deformations can refer to more complex consequences such as material bending, twisting and stretching. Such kinds of structural changes are recognised as strains which causes stresses that are generated in the continuum body.

Constitutive Laws, also called material laws are the mathematical characterisation of the behaviours of the material. Here we only discuss elastically deformable materials that are isotropic. The material properties of such kind of material are independent of the direction meaning that the material properties measured in all directions are the same. The well-known constitutive law that is used in computer graphics applications to describe the relationship between strain and stress is Hooke's Linear Law, which states

$$\sigma = E\varepsilon \tag{2.2}$$

σ is the *stress* and $\varepsilon = \Delta l/l$ is the *strain* resulting from the non-rigid deformation while E is known as the elastic modulus or Young's modulus and indicates the stiffness of the material. Hooke's Linear Law assumes the applied forces are not significant or do not cause bending or shearing, e.g., as exhibited by an extended or compressed spring. However, when the applied forces are either sufficiently large or act in arbitrary directions, then Hooke's Linear Law will be invalid. In such a situation, Hooke's Linear Law needs to be generalised to higher dimensions to represent the deformations in all the directions.

Figure 2.3 shows how the stress σ in the equation 2.2 is generalised in a 3D environment to quantify the strains at all directions. As shown in the figure, the stress of a particle in the material can be represented by the Cauchy stress tensor (Fung 1977) which is a second order tensor:

$$[\sigma] = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{2.3}$$

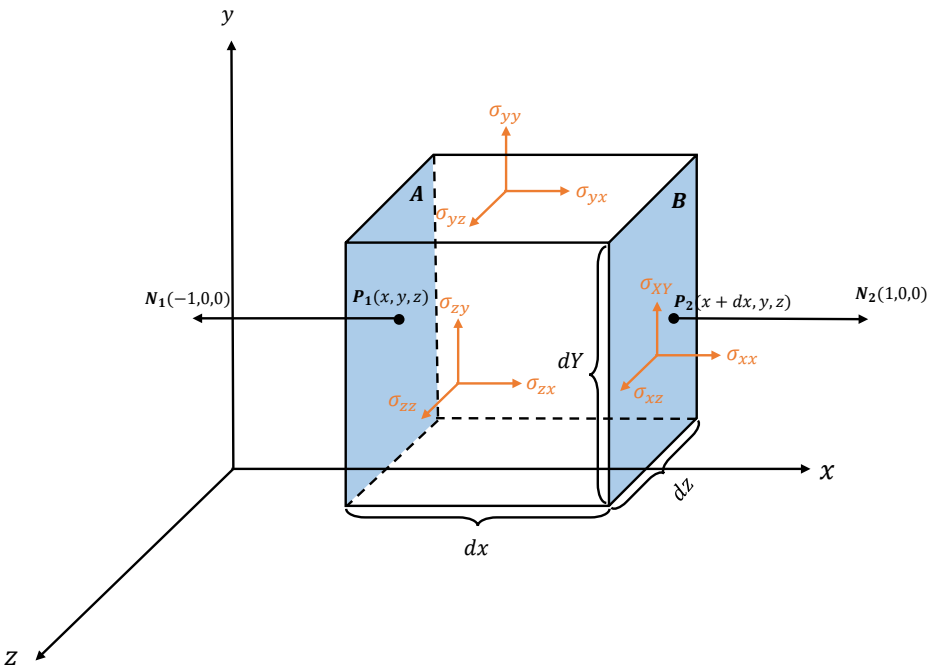


Fig. 2.3 Visual representation of generalised Hooke's law in 3D, the cube presents a particle of the material model. The stress σ has been generalised to describe deformations at all directions.

containing nine components represented by σ_{ij}. The first index i indicates the direction in which the stress component distributes, and the second index j denotes which face the stress acts on. The Cauchy stress tensor assumes the strains occurring on a particle are small enough so that they cannot cause shearing or twisting. Those materials whose deformation can be quantified by the Cauchy stress tensor are called linear elastic materials.

2.2.3 Deformation Gradient

To measure the deformations, the deformation gradient $\boldsymbol{F}$ which is a matrix field, is used to indicate the rates of change. For any particle in a continuum body, this can be calculated as the derivative of the current position vector $\boldsymbol{x}$ with respect to the reference position vector $\boldsymbol{X}$. As $\boldsymbol{U} = \boldsymbol{x} - \boldsymbol{X}$ then

$$\boldsymbol{x} = \boldsymbol{X} + \boldsymbol{U} \tag{2.4}$$

and hence

$$\boldsymbol{F} = \frac{\partial}{\partial \boldsymbol{X}}(\boldsymbol{X} + \boldsymbol{U}) = \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{X}} + \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{X}} = I + \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{X}} \tag{2.5}$$

For a rigid body translation the relative position between each of the particles in the continuum body do not change during the displacement. In this case the partial derivative $\frac{\partial \boldsymbol{U}}{\partial \boldsymbol{X}}$ is zero so that the deformation gradient $\boldsymbol{F} = I$, where I is an identity matrix. This says that the rigid body translation does not cause any stress.

Similarly a rigid body rotation does not change the shape of a continuum body therefore generates no internal stress. This is to say the value of gradient of the deformation which only

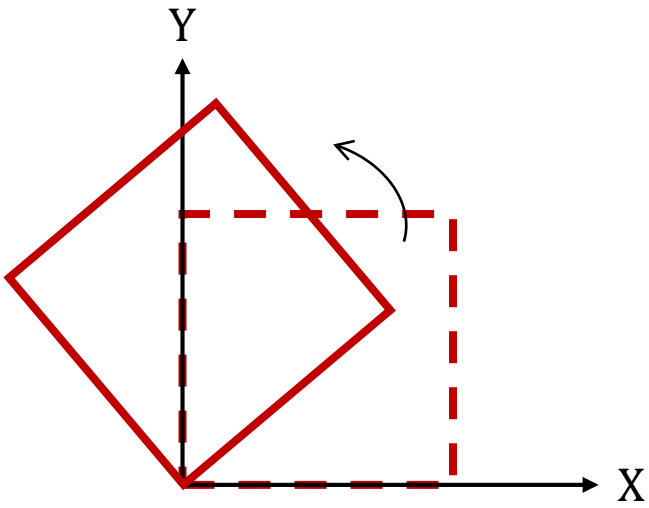


Fig. 2.4 An example of the rigid-body rotation. Such rotation should not cause internal stress as it does not change the shape of the continuum body.

includes the rigid body rotations should be equal to I. For an example, if an object is subjected to a pure rotation in two dimensions as shown in figure 2.4, in which

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{X}\cos\theta - \boldsymbol{Y}\sin\theta \\ \boldsymbol{y} &= \boldsymbol{X}\sin\theta + \boldsymbol{Y}\cos\theta \end{aligned} \tag{2.6}$$

then the rotation matrix is

$$\boldsymbol{F} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.7}$$

where θ is the rotation angle.

In this case, the deformation gradient is no longer equal to I after calculations only involving the rigid rotation matrix. This result indicates that the rigid body rotation is mistaken as the non-rigid body deformation. Therefore such rotation should be calculated and analysed separately avoiding calculation errors.

Consequently the challenge is to separate the rigid-body rotation and the pure deformation so that we can measure both of the displacements. Polar Decomposition is a technique popularly used in continuum mechanics to solve this problem. As shown by Chadwick (2012), the deformation gradient $\boldsymbol{F}$ can be decomposed into the product of two matrices

$$\boldsymbol{F} = RU = VR \tag{2.8}$$

where R is the rotation matrix, and U and V are symmetric matrices describing the deformations that cause strains and stresses.

2.2.4 Origins of the Physically-based Deformable Models

Kinematic models are traditionally used in computer graphics applications. Such models only study the mathematics of motion without considering the forces that affect the motion. Imagining a scene in the game in which a man which is modelled by the kinematic method is running with a consistent velocity, if you push him at the back, he will not change his velocity or direction, i.e. no response to any applied external forces. Moreover, the shape of such models cannot change. This is an efficient way to represent rigid materials whose shape cannot change even under applied external forces. Unfortunately, they are passive models that do not interact with other objects or react due to any applied external forces unless pre-described changes on shapes are provided. Physically-based methods have been widely used in computer graphic applications to simulate realistic physical behaviours of deformable materials in the real world (Zhang et al. 2017).

Space and time-dependent physics problems can usually be expressed by partial differential equations (PDEs). For an infinitesimal point which is at position $\mathbf{p}$ of an elastic material, the dynamic motion of the infinitesimal point can be described by:

$$\rho \ddot{x} = F_{internal} + F_{external}, \tag{2.9}$$

where ρ is the density of the material and $F_{external}$ represents the total external force applied on the object. Treating the infinitesimal point as a cube as shown in figure 2.3, $F_{internal}$ be calculated by adding up all forces applied on the six faces of the cube, i.e.,

$$\begin{aligned} F_{internal} &= F^x + F^y + F^z \\ &= \begin{bmatrix} \frac{\sigma_{xx}}{dx} + \frac{\sigma_{xy}}{dy} + \frac{\sigma_{xz}}{dz} \\ \frac{\sigma_{yx}}{dx} + \frac{\sigma_{yy}}{dy} + \frac{\sigma_{yz}}{dz} \\ \frac{\sigma_{zx}}{dx} + \frac{\sigma_{zy}}{dy} + \frac{\sigma_{zz}}{dz} \end{bmatrix} \\ &= \nabla \cdot \sigma. \end{aligned} \tag{2.10}$$

Terzopoulos et al. (1987) employ elasticity theory into kinematic models to construct differential equations which describe the physical behaviours of deformable objects composed of rubber, cloth, and paper. The authors also consider other general inelastic behaviours such as viscoelasticity, plasticity, and fracture. They represent the dynamic motion for the particle in the deformable model by rewriting the differential equation 2.9 in Lagrange's form as:

$$\frac{\partial}{\partial t}(\mu_{(a)} \frac{\partial_{r_{(a,t)}}}{\partial t}) + \gamma_{(a)} \frac{\partial r}{\partial t} + \frac{\delta_{\varepsilon_{(r)}}}{\delta r} = f(r,t) \tag{2.11}$$

where $f(a,t)$ is the total external forces applied on the deformable model. The left side of the equation calculates the force produced due to the external forces. The first term on the left hand side represents the inertial force from the model's distributed mass where $r(a,t)$ is the position

of the particle a at time t and $\mu(a)$ is the mass density. The second term describes the damping force due to dissipation. For an infinitesimal element the damping is divided by its volume $dx \cdot dy \cdot dz$. Therefore, $\gamma(a)$ represents the damping capacity per unit volume at a which is so called as *damping density*. The third term is the elastic force that result from the deformation of the model away from its rest shape, and the $\varepsilon(r)$ represent the potential energy of the resulting deformation. In a subsequent paper the authors also implemented such techniques into fracture simulation (Terzopoulos & Fleischer 1988*a*,*b*).

The integration of a purely geometric model and the corresponding dynamic differential equations allows the shape of the model to change over time. In this manner, the deformable models can react with more realistic animations involving the intersection with other models with various external applied forces in the simulated environment. It also opens the research area for specifically aimed applications such as virtual surgery, where the models are expected to response with a realistic animation when colliding with the virtual scalpel.

2.3 Physically Based Deformable Models and Applications

To be able to solve such PDEs as given by equation 2.9 analytically the equation normally has to be simplified by making assumptions and hypotheses which are so far away from being realistic in the real world. Hence the PDEs which represent the majority of physically based problems have to be solved numerically to get a solution which accurately represents the true behaviour. Numerical solutions based on discretisation are therefore used to approximate the solution of PDEs. In this section, we will introduce the well-known, Finite Element Method (FEM) that is used to solve partial differential equations in computational mechanics and other methods including the Boundary Element Method (BEM), Mass-spring system and Point-based Animation.

2.3.1 Finite Element Method

The Finite Element Method (FEM) is a powerful approach used to provide an approximate solution of PDEs in computational mechanics. This method can be applied on an arbitrarily-shaped simulated continuum body with specific boundary conditions and can lead to a large-scale PDE when the size of the problem domain is large. The principle of the FEM is to convert the solution of a PDE into the solution of a set of algebraic equations that can then be solved numerically. To this end, a simulated continuum body is represented by nodes and a finite number of disjoint elements (see figure 2.5) in which the behaviours of each element is governed by an algebraic equation. It is worth noticing that the FEM only results in a set of linear equations if it is applied to a linear PDE. As this dissertation primarily concerns 2D surface materials the FEM will only be applied to 2D surface models.

We assume a 2D problem domain is discretised into a mesh with triangular elements. Therefore the problem of solving the PDE governing the behaviours of the whole problem domain is

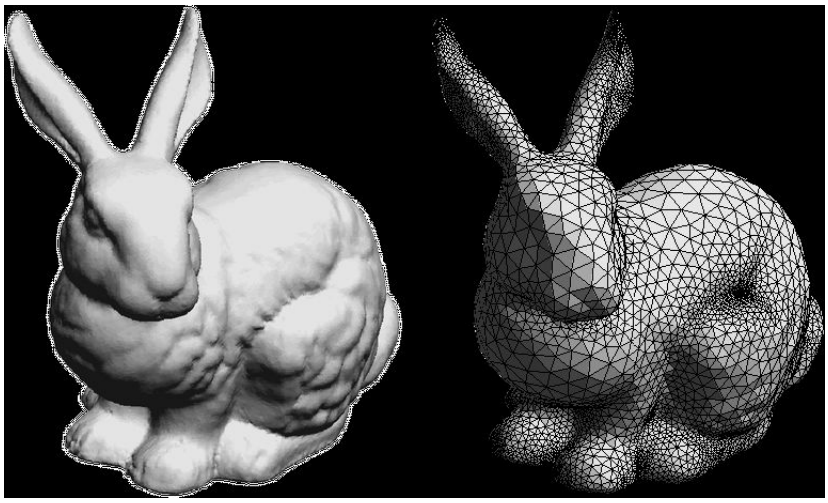


Fig. 2.5 Example of a FEM-based model. The surface of the bunny is represented by nodes and triangular elements. Retrieved in http://mech.fsv.cvut.cz/~dr/papers/ECT02/node15.html.

transferred to solving a set of algebraic equations that describes the behaviour of each element and assemble the solutions to approximate the true solution of the original PDE. Each node in the mesh has Degrees of freedom (DoFs) in x and y directions. For each triangular element, the displacement field $\boldsymbol{U}_e$ is calculated by the three nodal displacements obtained on each vertex with the index i, j, k in single element, i.e.

$$\{\boldsymbol{U}_e\} = \begin{Bmatrix} \boldsymbol{U}_i \\ \boldsymbol{U}_j \\ \boldsymbol{U}_k \end{Bmatrix} \tag{2.12}$$

in particular, the displacement at any position inside the element has to be interpolated involving the three nodal displacements. To this end, a shape function is constructed on each element to obtain the value of any position within the element.

Both linear and non-linear elements can be employed in the FEM. The linear elements, e.g. triangle, rectangle only have DoFs at the vertices of the primitive. Non-linear elements (also known as higher-order elements) can be achieved by adding nodes at the middle of each edge of the linear element. In particular, for elastic FEM-based models, the linear elasticity is only valid when employing the linear elements. Figure 2.6 gives an example of a linear triangular elements which has three control nodes and non-linear triangular element in which the middle of each edge is added one node. As shown in the figure, when applying forces to the linear and non-linear triangular element the non-linear triangular allows more forms of deformations while the linear elasticity is less accurate in such a situation (Solin et al. 2003).

After selecting the type of element and the degrees of freedom within each element, the next step is to define the strain-stress rule to link the internal forces $f_{internal}$ within each element with the displacements. The relationship can be described as

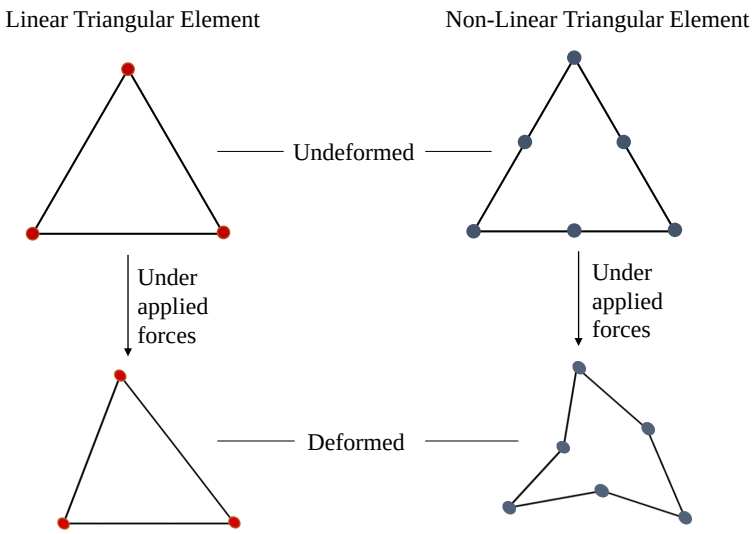


Fig. 2.6 Example of a linear triangular element and non-linear triangular element. The linear triangular element (left) has three control points while the higher-order triangular element (right) has six control points. When applying forces to an element more control nodes can provide more detailed deformation, but more complicated governing equations need to be constructed.

$$f_{internal} = f(\boldsymbol{U}_e) \tag{2.13}$$

Employing linear elastic theory this can be rewritten in a matrix form

$$f_{internal} = \mathbf{K}_e \cdot (\boldsymbol{x}_e - \boldsymbol{X}_e) \tag{2.14}$$

now here where $\mathbf{K}_e$ is the Jacobian matrix $\partial f_{internal}/\partial \boldsymbol{x}_e$ of $f_{internal}$, known as the element stiffness matrix (Hughes 2012). $\boldsymbol{x}_e$ and $\boldsymbol{X}_e$ include the *spatial* coordinates in the current configuration and material coordinates in the reference configuration of all nodes in a element. In this manner, each structural primitive can be treated as an individual computation. Consequently the global equations for the whole mesh can be obtained by assembling the element equations. The global stiffness matrix $\mathbf{K}$ for the whole mesh can be obtained by

$$F_{internal} = \mathbf{K} \cdot (\boldsymbol{x} - \boldsymbol{X}) \tag{2.15}$$

where $F_{internal}$ assembles the internal forces generated within each element in the mesh, $\boldsymbol{x}$ and $\boldsymbol{X}$ include the all *spatial* coordinates and material coordinates of all nodes in the mesh respectively.

In early methods, the problem domain is discretised into a finite number of regular spatial grids (Terzopoulos et al. 1987). The dynamic elastical motion equation can be discretised using finite differences. This is often easy to implement and calculate. However, it may prove difficult when trying to handle the complex boundary of an arbitrarily-shaped object with elements that have the same size and shape. In addition, when applying special effects such as cutting or fracture on such mesh, it is impossible to incorporate local adaptations to satisfy the accurate details.

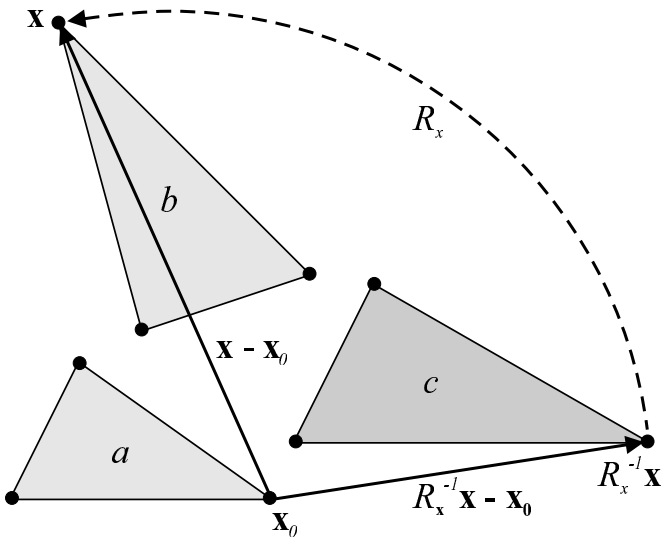


Fig. 2.7 The Co-rotational FEM. Original resource from Müller et.al's work (Müller et al. 2002). The undeformed model (a) in a reference configuration transferred into current configuration with a deformed form (b). The co-rotational FEM transferred the deformed model back to the reference configuration with the pure rotation part R_x and then process the linearised transformation (c).

Overcoming this obstacle in the standard FEM, O'brien & Hodgins (1999) constructed irregular-sized elements to simulate the deformable objects. They introduced a simple form of the FEM which computes the nodal forces locally from the adjacent elements instead of solving the equations associated with the unknown nodal *spatial* coordinates. In such a method, the force due to the distributed mass and the internal forces due to the deformation and the applied external forces on nodes of the deformable model are not calculated separately. The authors introduced this method in the modelling and animation of the brittle fracture of elastoplastic materials. They employed a finite number of tetrahedra to discretise the deformable model to describe the plastic objects. By determining the stress generated over each element of the model, the model can then decide where to initiate the crack and in what directions the crack should propagate subsequently.

As mentioned in the section 2.2.2, the linear elastic materials are only valid under small deformations. Hence elastic FEM-based models which obey the linear elasticity can only handle small deformations otherwise the solution can result in unrealistic or unstable effects when large (non-linear) deformations are involved. To resolve this issue, Müller et al. (2002) introduced the Co-rotational FEM (also known as *warped stiffness* FEM), in which large deformations are handled by extracting the rotations before solving the system using a linear elasticity approach. In this manner, the problem is transferred into solving linear (or small) deformations under a rotated coordinate system. The principle of this method is to remove the rotation part in the transformation from $\boldsymbol{X}$ to $\boldsymbol{x}$ in Equation 2.15.

Therefore Equation 2.15 can be rewritten in the form

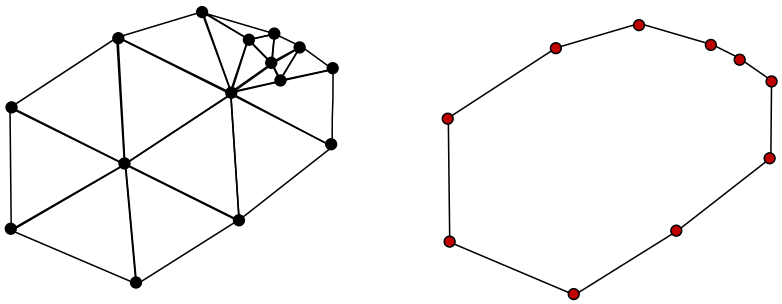


Fig. 2.8 Take a 2D problem domain as an example, the FEM (left) discretes the domain into a finite number of elements (e.g. triangles in the figure), and the black dots are the interior nodes. However, the BEM (right) only discretise the boundary of the domain, only on which generate the internal particles (shown as the red dots).

$$\begin{aligned} F_{internal} &= \mathbf{RK} \cdot (\mathbf{R}^{-1}\boldsymbol{x} - \boldsymbol{X}) \\ &= \mathbf{RKR}^{-1}\boldsymbol{x} - \mathbf{RK}\boldsymbol{X} \end{aligned} \tag{2.16}$$

where **R** represents the pure rotation matrix and then the linear elastic deformation is computed under a rotated configuration by multiplying the global stiffness matrix **K** by **R**. Later Müller and his colleague explore the approach in a real-time fracture simulation for a wide range of materials (Müller & Gross 2004). They represent the fracture locations and orientations by determining the stress, and a non-uniform subdivision strategy enables accurate representation of the fracture edge on the employed tetrahedral mesh which is commonly used in modelling volumetric models.

The Co-rotational FEM avoids the unrealistic effects caused by linear distortions that can result in incorrect forces under large deformations. In this manner it provides realistic results efficiently which are adequate for accurate interactive simulations.

Co-rotational FEM provides a simplified geometrically nonlinear formulation to represent nonlinear deformation of 3D models. We embrace it with our novel cutting technology to provide a high-performance physically-based model allowing complex cutting in real-time.

2.3.2 Boundary Element Method

In contrast to the FEM in which all computations are distributed throughout the element the Boundary Element Method (BEM) discretises only the surface of the problem domain (see right hand side of figure 2.8). In this manner, it significantly reduces the computational overhead as it results in a substantial reduction in the dimensionality e.g. for a two-dimension model only the line-boundary of the domain need to be discretised; while for the volumetric model it is the two-dimensional surface.

James & Pai (1999) were the first to integrate the BEM in deformable object modelling and animation. They presented the boundary integral equation formulation of static linear elasticity with the BEM discretisation technique. As no discretisation is needed inside the problem

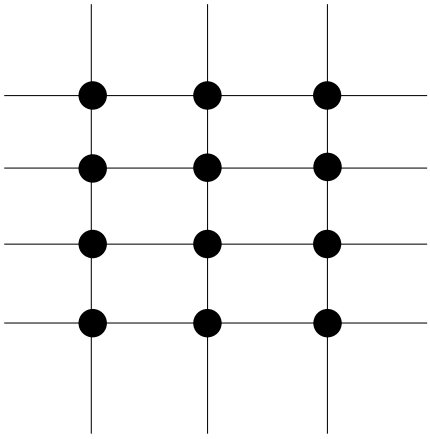


Fig. 2.9 The Mass-Spring System introduced by Breen et al. (1994) to simulate the rigid deformation of cloth. The system consists of particles and massless springs that are connected in the direction of weft and warp.

domain they successfully provide a fast, physically accurate real-time simulation for virtual deformable objects.

The BEM provides an effective alternative to the FEM when modelling deformable objects. However it is not suitable for modelling complex structural models. The Boundary Element Method treats the interior of the model as continuous without discretisation. In this manner it is difficult to handle the discontinuities in the mesh to satisfy particular effects such as cutting or fracture.

2.3.3 Mass-spring System

The Mass-Spring System (MSS) is considered as the simplest and most intuitive method to construct and implement. Instead of constructing a continuous model with a Partial Differential Equation (PDE) as mentioned in section 2.3.1, the MSS directly establishes a discrete model. In such a method the problem domain is simply constructed with a set of point masses connected by massless linear springs with fixed connectivity. For a given time t, the forces $\mathbf{F}_m$ applied on single mass are calculated from its spring connections with its neighbouring masses and also with any external forces such as gravity, etc. The governing equation of motion is derived from Newton's second law $F_m = m\ddot{\boldsymbol{x}}_m$, in which $\boldsymbol{x}_m$ is defined as the spatial coordinate of the masses $i = 1, \ldots, n$. Therefore, the forces for the entire model can be calculated as:

$$\mathbf{M}\ddot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{v}), \tag{2.17}$$

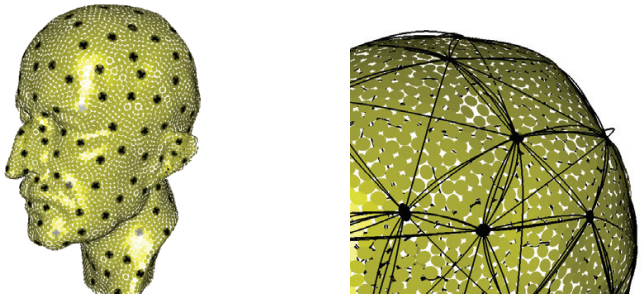
where $\mathbf{M}$ is the $3n \times 3n$ diagonal mass matrix, and $\mathbf{v}$ represent the instant velocities of the masses. The dynamic behaviour of the model can be solved with the numerical integration of Ordinary Differential Equations (ODEs).

Breen et al. (1994) introduced the Mass-spring System by integrating the laws of elasticity to reproduce the drape of woven fabrics with a wide variety of material properties. Compared to the FEM which employs continuum mechanics to approximate the true solution this approach establishes a discrete system in which a set of interacted particles are connected by massless springs with different natural length (see figure 2.9). The authors derived the model's energy functions based on the data that was obtained from the real cloth sample testing. In this manner, this technology is capable of characterising the draping behaviour of clothes with different material properties. They evaluate the visualisation effects of the simulation by comparing with the photographs of the drape of the actual cloth.

The spring-based approach provides relatively easy construction and convincing behaviour for a real-time simulation of simple cloth. Mass-spring systems can handle topological changes at a low computational cost by merely removing or changing the connection of springs. Unfortunately the model relies heavily on the regular pattern associated with the connectivity of the springs to produce a convincing simulation. Once the pattern is altered (e.g., via a cut or tear), a spring has to be removed entirely rather than divided into two as there is no continuous interpolation defined on springs. Holes created by cutting can be re-meshed by introducing new springs to align with the cutting trajectory in any direction. However, the new spring no longer describes the weft, warp, shear or bending stiffness of the material, because the force calculation along the spring is inconsistent with the original direction.

Cotin et al. (2000) presents a method to provide an interactive cutting simulation for virtual surgery in real time on the volumetric deformable object model with complex anatomical structure. They introduce a hybrid model, in which the domain is discretised into a tetrahedral mesh grid while the dynamics are computed based on the so-called tensor-mass system (also known as explicit FEM). In such a system, the problem domain is discretised by a finite number of elements (e.g. triangle, tetrahedra); however, the nodal forces are calculated from the adjacent nodes rather than solving a system of element equations as the standard FEM-based modelling.

Employing linear elastic theory enables the representation of real-time cutting and tearing even on deformable models with a sizeable anatomical structure (more than 8,000 tetrahedra). However this theory is not valid when the deformation is larger than 10% of the material's size (Fung 1977). Similar to the mass-spring system the tensor-mass model is efficient in terms of computational complexity as it discretises the model into a set of massed particles and springs while it also represents more realistic material behaviours since it defines continuous spring interpolation functions. It is based on continuous mechanics which means that when springs are removed or added the system can still maintain a convincing behaviour. It is a good alternative for the mass-spring system and finite element method. However, in a tensor-mass model no interpolation function is constructed within an element. Therefore, the collision with such a model can only happen on the vertices and springs but not the space within each element.



(a) (b)

Fig. 2.10 The source from Wick et al.'s work (Wicke et al. 2005).In their work, fewer sampled points are treated as the computational points when simulating the deformations of the underlying geometrical model which has more geometry points (a). The fibre system including sampled points and parametric curves mapped onto the underlying geometrical model (b).

2.3.4 Point-based Animation

Wicke et al. (2005) present a mesh-free animation, which employs a point-based animation to simulate the topological changes of thin shell models efficiently. As shown in figure 2.10, this method discretises the thin shell computational equation with so-called fibres. The system of fibres is composed of point samples connecting with local parametric curves rather than geometric primitives (e.g. triangle, rectangle) which have consistent connectivity. All computations for the dynamics and topological changes are executed by using such fibre system in order to achieve robust and fast updating. This approach allows the generation of fast animations as the high-resolution geometrical model is mapped on to such a low-resolution fibre system.

This approach provides a mesh-free method which has no limitation on the geometrical structure of the underlying model. It achieves fast updating of deformation as the displacement field interpolation is only defined at the sampled points, which are treated as the computational nodes to transfer the computed deformation back to the high-resolution mesh surface. However the resampling strategy can be significantly expensive in order to make sure the surface model is adequately sampled under large deformations. The high-resolution mesh is always mapped onto a coarse fibre network using the sampled points so that there is no guaranteed way to sample arbitrary edges after any topological changes. Therefore it would be a challenge to integrate such a technique into simulations which need to represent precise cut incisions such as virtual surgery.

Steinemann et al. (2006) introduce a method to quickly represent a model discontinuity that is caused by the cutting or fracture of deformable solid models. They discretise the deformation field based on a mesh-free method with a so-called visibility graph which stores the node connectivity of the model. The crack surface is synthesised as triangular meshes by the sampled points (deformation nodes). The pre-computed visibility graph estimates the node-to-node distance along visible paths. Upon cutting or fracture, the edges intersecting with the cut path are removed from the visibility graph and the connections between deformation nodes are updated by calculating the new shortest route around the cut.

This method cannot model highly-detailed cut incisions, primarily because the sampled points may be insufficient for complex arbitrarily-shaped cut edges. It may be acceptable in applications that purely require an efficient simulation but it is not adequate for applications such as surgery simulations. The point-based animations reduced the computational time for deformation and model continuity updating with the coarse embedded crack surface. However the so-called visibility graph needs to be updated and computed in each simulation step even when no topologically changes occur thus requiring an additional data search strategy to maintain and track the node adjacency during the overall simulation process.

2.4 Mesh Representation in Cutting Simulations

The mesh representation is crucial for a realistic cutting simulation in computer graphics applications. The choice of topology and discretisation is mainly influenced by the geometrical structure of the simulated objects. For a surface-based object such as cloth, the 2D mesh is a more straightforward approach than employing 3D meshes. However 3D meshes have the dominant position in the field of volumetric objects. Furthermore multi-layer meshes are required when the simulated object is made of composite materials or structures such as skin. As there is no general solution for mesh representation for multi-purpose application it is important to choose appropriate topological meshes for different types of cutting simulations.

2.4.1 2D Object Representation

A shell can be defined as a thin shell if it has an extremely small thickness compared to it's other dimensions and the deformations are not significant compared to thickness. 2D meshes are ideal to represent thin shells in the real world. In computer graphics both 2-dimensional cloth and skin simulations are crucial for a wide range of applications.

Adaptive re-meshing in cloth dynamics. Narain et al. (2012) present an approach for providing dynamic high-detail wrinkles in cloth simulation. They employ triangular finite element based meshes for the geometrical construction of simulated cloth. During the dynamics of the cloth model, they automatically increase the number of elements where the curvature of wrinkles becomes high while reducing the numbers of elements when not necessary. The anisotropic meshes produced during such operations allow it to satisfy the geometric details of the simulated cloth. By employing strain limitation to bound the permitted strain on each linear edge, the authors achieved stable and fast stiff spring modelling.

The adaptive re-meshing technique reduces the substantial computational overhead resulting from the uniformly high-resolution original meshes. This technique allows us to employ local optimisation in the cutting simulation. However, the need to maintain the quality of the generated elements during the topological changes is not taken into account. In addition the method is not designed for real-time applications primarily because the strain limiting based on constraint projections needs to operate in each time step and it can be time-consuming.

Skin simulation. A 2-dimensional mesh is also suitable for representing real human skin which is only a few millimetres thick. Using the advanced rendering techniques available in computer graphics we can reproduce visually realistic skin. However being able to model the movement of skin and react like real skin is still an open research field. Saito & Yuen (2017) present an point-based animation for simulating skin sliding caused by stretching other skin regions or any deformation from the underlying geometrical model. They employ an optimisation technique called the Alternating Direction Method of Multipliers (ADMM) to integrate the elastic thin-shell simulation and the collision handling intuitively. In particular, they achieved a fast collision detection which only performed on the 1-ring neighbours for each node of the model.

In contrast to the classic simulation method mentioned in Section 2.3.1, they calculate the positional difference between deformation configurations in every simulation step directly by solving a quasi-static problem. This then makes the position-based animation well-suited for satisfying interactive requirements. While it is generally not as accurate as the physically-based animation it provides visual plausibility. Therefore it provides an efficient and stable solution for visual-focused applications, i.e. special effects in movies.

2.4.2 3D Object Representation

It is common to employ deformable 3-dimensional geometric models to represent volumetric objects both in virtual surgery and visualization applications. The rigid volumetric models are widely used for fracture simulations in computer graphics applications while the soft volumetric models are heavily used in virtual surgery simulators. As the 3D mesh construction and cutting is not within the scope of our contribution we refer the readers to Delingette's review (Delingette & Ayache 2004) for the details of the 3D modelling and computational method used in virtual surgery and cutting technologies (Wu et al. 2015).

Fracture and cracking. Koschier et al. (2014) reproduce the brittle fracture of rigid bodies with an adaptive refinement scheme on the simulated tetrahedral model. The tetrahedral of the model are refined uniformly when the stress applied exceeds the material strength threshold. The further refinement is determined by the immediate stress change. Once the tensile stress decreases below the fracture threshold the reverse operation reduces the computational overhead caused by the refinement. Multiple cracks are allowed to generate under single stress applied without shattering artefacts.

Even though the neighbours of the refined tetrahedra also need to be refined the local refinement scheme reduces the computational cost without needing a high-resolution mesh. The uniform subdivision is straightforward to implement especially for 3D geometry. Indeed such subdivision based on tetrahedra generates the same type of geometry which avoids the multi-structure mesh. Therefore, it ensures an efficient re-meshing scheme and avoids the errors in the matrix assembly which is usually caused by non-uniform subdivisions. However, in trajectory-based rather than force-based cutting simulations, it may be necessary to subdivide the intersecting elements irregularly to represent the actual cutting edge.

Human organ surgery. Courtecuisse et al. (2014) reproduce an interactive cuttable heterogeneous soft-tissue constructed by tetrahedra. It supports more accurate collision response with coupling contact and friction in the dynamic interaction process between the simulated objects. It subdivides the interacted element, aligns with the real user-defined cut trajectory and mitigates the convergence issues associated with the generated ill-shaped elements resulting from an asynchronous pre-conditioner. They implement the method into three virtual surgery simulators including cataract surgery, laparoscopic hepatectomy and brain tumour surgery.

Ill-shaped mesh elements raise convergence issues in simulations. Indeed, they can increase the condition number of the Conjugate Gradient solver that will be introduced in Section 2.6.2. Courtecuisse et al. adopt the LDL^T factorisation of the global stiffness matrix for the simulated system to decrease the updating frequency of the pre-conditioner during the simulation. In this way the convergence problem is reduced but cannot be eliminated. Hence the quality of the generated elements explicitly influences the performance of the simulation and so maintaining a high mesh quality is required in any simulation which allows complex cutting.

2.4.3 Multi-layer Object Representation

In the real-world most thin plates are not made as a single layer. Compliant materials such as paper and composite cloth reflects different regional topological changes between layers. A straightforward way to simulate such materials is to employ multi-layer surface models.

Ageing skin simulation. The multi-layer surface model is mainly used in the simulation of ageing skin that changes in the mechanical properties caused by the changes between the various skin layers. Flynn & McCormack (2010) researched the simulated multi-layer finite element skin model to investigate the main factors that cause ageing. These factors including the moisture content of the stratum corneum and the variation in the dermal collagen fibre density. The generated in-plane compression and the wrinkled size are measured to evaluate the level of ageing. The research result showed that the reduction in the moisture content of the stratum corneum and the increase in the density of collagen fibres caused further ageing in the skin model.

Skin ageing is difficult to observe in the real world primarily due to the complex structure of the skin and its connection with various anatomical structures such as muscles and heart. A precise skin model that can simulate the natural ageing process makes it easy and controllable to investigate the reasons causing ageing. Flynn and McCormack's work is purely parameter-based on the static model, while a multi-layer skin model integrating physical-based dynamics is required for a more realistic skin simulation.

Facial animation. Warburton & Maddock (2015) present an approach to simulate convincing facial animations including expressing wrinkles efficiently. They represent the human forehead by a non-conforming voxel-based hexahedral deformable model. Throughout the simulation, a muscle contraction model generates the transversely isotropic active stress. Rigid nodes are then

represented as fixed nodes that cannot have any displacement while the sliding nodes can move over time. In this manner wrinkles can be generated due to the different movements when stress is applied to the simulated model. They utilise the GPU to store all variables precomputed (e.g., shape function derivatives, element volumes, and constants in the nodal displacement calculations) to increase the speed of CPU-GPU data transfer.

This work provides an efficient approach to simulate facial wrinkling. The non-conforming multi-layered finite element model has the flexibility to simulate a variety of soft-tissue by employing different material parameters. Since there is no adaptive refinement of the mesh the method is not sufficiently accurate. Compared with a volumetric mesh, 2D meshes is more suitable for the representation of thin shell materials such as skin which is only a few millimetres thick.

2.4.4 Applications of Cutting/Slicing/Tearing

Drilling holes. Seiler et al. (2011) propose an approach to efficiently and robustly simulate interactive cutting, i.e., drilling holes in deformable objects. They employ a regular hexahedral octree based simulation mesh while embedding a high-resolution triangular surface for rendering. In this manner, a cut is represented by locally refining the hexahedral simulation mesh and splitting the affected nodes when the simulated blade is interacting with the model. They introduce a back-transformation technique for the simulated blade to perform the model modification in the rest configuration of the model. Furthermore they present an efficient updating approach called a Z-curve for the sampled simulation nodes and the generated underlying geometry nodes to represent the cuts visually.

The work separates the mechanical representation from the visual representation by employing a hexahedral octree-based mesh. As this technique refines the impacted hexahedral into smaller tetrahedra it maintains the quality of the geometric mesh without generating ill-shaped elements. However, this limits the physically based behaviours of the simulated object as only the corner nodes of each hexahedral are chosen as the simulation nodes. Moreover, rather than the underlying geometrical mesh, the hexahedral octree based mesh is used for collision detection. Hence the collision with the boundary of the underlying geometrical mesh is not accurate. However, for visually focused applications, this can be an efficient and robust approach to simulate interactive cutting.

Tearing. Busaryev et al. (2013) present a study on how to simulate more realistic fracture animations resulting from the tearing of a thin-plate from three perspectives where thin-plate is a term used to describe cloth-like deformable bodies constituted from a surface only. They relax the stress in the elements that have been torn to avoid the model from fracturing into pieces. They employ a re-meshing algorithm based on a Delaunay triangulation. The use of a Delaunay mesh generation provides a smoother approach and lowers the possibility of irregular shapes. This approach is applied to a multi-layered model to illustrate complex behaviour.

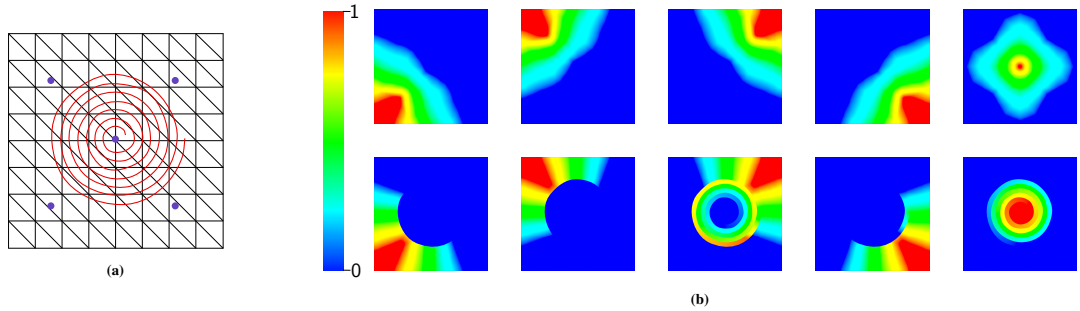


Fig. 2.11 The source from Manteaux et al.'s work (Manteaux et al. 2015). In the left figure (a), five control frames (whose centre is depicted by blue points) are embedded on the underlying geometric mesh which illustrate the topological structure of the model (black lines). In the right figure (b), each control frame is allowed to be non-manifold (bottom row) rather than manifold (top row) to represent accurate cut boundaries.

Unfortunately this work is not designed to provide real-time interaction. This is primarily due to the global re-meshing scheme which is employed where the whole surface model has to be evaluated and re-constructed irrelevant of the size of tearing. Considering the entire surface post-cut is an easier problem to solve. Moreover this work does achieve re-cuts and overlapping tearing by implementing the re-meshing scheme on a multi-layer model. It provides a convincing visual effect of composite materials in the real world. However the layer models are independently constructed, in a way such that the computational time can be doubled when determining the topological changes compared with a single-layer model.

Cutting. Manteaux et al. (2015) present a method for allowing cutting on deformable thin sheets at the interactive rate. The underlying geometric mesh which illustrates the topological structure of the model is embedded with low-resolution control frames that will induce deformation using the linear blend skinning method. The grid for each control frame can be non-manifold in order to represent detailed cut boundaries. Then each affected frame is duplicated times depending on the number of broken parts and each replica store different connectivities to illustrate the cut visually.

They replace the physical representation of the simulated model with a frame-based solution in which each frame is equipped with a single collision node. This decreases the computational cost of deformation as each frame can cover a vast region of the high-resolution underlying mesh. Their method determines the intersection between the simulated blade and the frames rather than the highly detailed geometry mesh which reduces the detection cost. The times of duplication for each frame depends on the number of broken parts. Therefore the detection time increases when the intersected frame is broken into more parts. It does achieve detailed cuts and multi-cuts inside a single frame.

2.5 Surface-based Model Cutting versus Volume-based Model Cutting

Both plane and volumetric models are employed in virtual cutting applications. However, because they differ in the way the mesh is constructed and the complexity of re-meshing they are used to model different types of material. The surface-based model is more suitable for modelling objects with a small thickness than the volumetric model. This is primarily due to the fact that the computation time rises as the thickness increases. However, surface-based models are not able to describe the characteristics of volumetric models, which are required in specific applications such as virtual heart surgery. This section discusses the fundamental difference between surface-based model and volume-based model cutting in relation to re-meshing and computational performance.

2.5.1 Difference Between Surface-based Model Cutting and Volume-based Model Cutting

For both surface-based models and volume-based models, a cut is detected and recognised as the intersection between the simulated blade and geometry mesh. However, the shape of cut trajectory determines the complexity of cutting in both surface-based models and volume-based models. For surface-based models, the cut trajectory can either be straight or curved whenever a cut is made through several elements or within a single element. In this manner, the final cut trajectory is approximated by connecting the path points. Usually, it is discretised into a piecewise linear path using intersection points between the simulated blade and affected edges. Consequently the smoothness of the cut incision relies on the number of sampled path points. In contrast, it must be a closed primitive whenever cutting a volumetric model progressively or non-progressively. Besides, the shape of the intersection surface varies from different volumetric geometry, e.g., the intersection surface is a triangle when cutting tetrahedra, while it can be triangle, quadrilateral, pentagon or hexagon when cutting a cube.

Furthermore, the re-meshing scheme also indicates the difference in cutting between the surface-based and volume-based models. Compared with the rectangle, the triangle is widely used in the cutting simulations of the surface-based model for several reasons. Firstly, when representing an arbitrarily-shaped cut edge, a higher resolution mesh needs to be generated by using rectangles rather than triangles. Moreover, the refinement of a single cut rectangle may result in a multi-dimensional mesh. In such meshes, as each element has different degrees of freedom they have to be analysed separately, which increases the mathematical calculation time. Similar to surface-based model cutting, the geometry structure and refinement scheme complexity influence the executability of cutting operation on volume-based models. Tetrahedra are mainly employed for the construction of the geometry mesh that allows cutting. As the subdivision and refinement of volumetric models are not included in our contributions, we would like the reader to refer to Bieler's work (Bielser et al. 2004) on the subdivision of tetrahedral during progressive cutting on volumetric model.

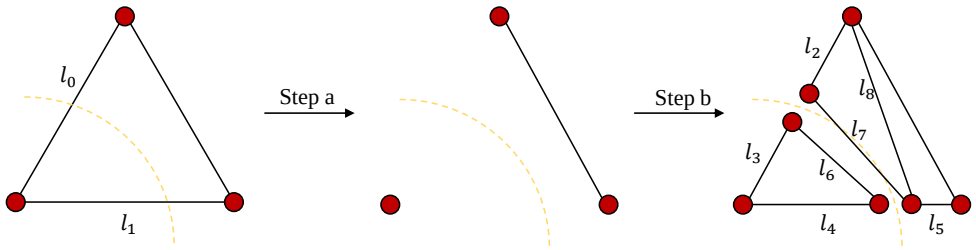


Fig. 2.12 The spring-based system cutting process illustrated by a triangular mesh. The intersected springs are removed first, and new springs ($l_2 - l_8$) with various rest lengths are created to represent the cut edge.

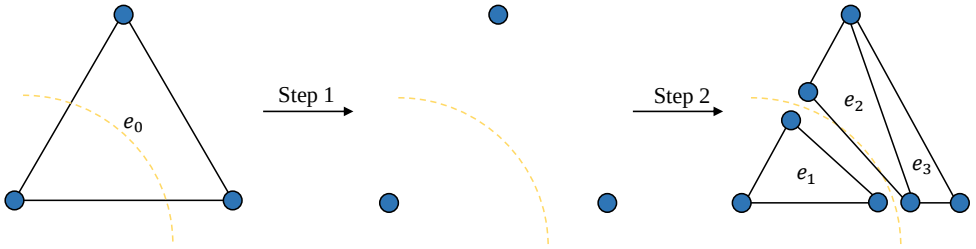


Fig. 2.13 The typical FEM-based mesh refinement allows cutting illustrated by a triangular element. The affected element (e_0) is removed in step 1, and new element (e_1, e_2, and e_3) create on both sides of the cut plane in step 2.

Figure 2.12 illustrates the cutting process when using a spring-based mesh. The process employs a discrete approach in which each spring is independent and the intersection detection operates between the cut plane and the individual springs in the mesh. The intersected springs (l_0, l_1) are removed first in step a. New springs (from l_2 to l_8) are generated with different rest lengths depending on the orientation and place of cut trajectory in step b. Conversely, in the FEM-based mesh the cut starts by detecting the intersection between the cut plane and each element within the geometry mesh. As shown in figure 2.13, the affected element e_0 that has edges intersecting with the cut plane are removed in step 1. On both sides of the cut plane, new elements are created to fulfil the topology.

The cutting method for volume-based models requires meticulous strategy due to the complexity of the intersection pattern. Figure 2.14 illustrates the all six possible intersection patterns (from A to F) during one continuous cutting process. Most researchers employ tetrahedra to carry out the cutting algorithm on volume meshes due to its relatively simple decomposition and reconstruction comparing with other volumetric primitives. As shown in figure 2.14, the different topological subdivision cases are distinguished by the number of edges intersected with the simulated cut blade. After an edge or face intersected with the cut plane, new surfaces (triangles here) are created to represent the outer shell of the volume mesh.

2.5.2 Performance Comparison

To compare the computational performance of cutting on surface-based models and volume-based models, we consider the mesh-based subdivision operation as an example. For

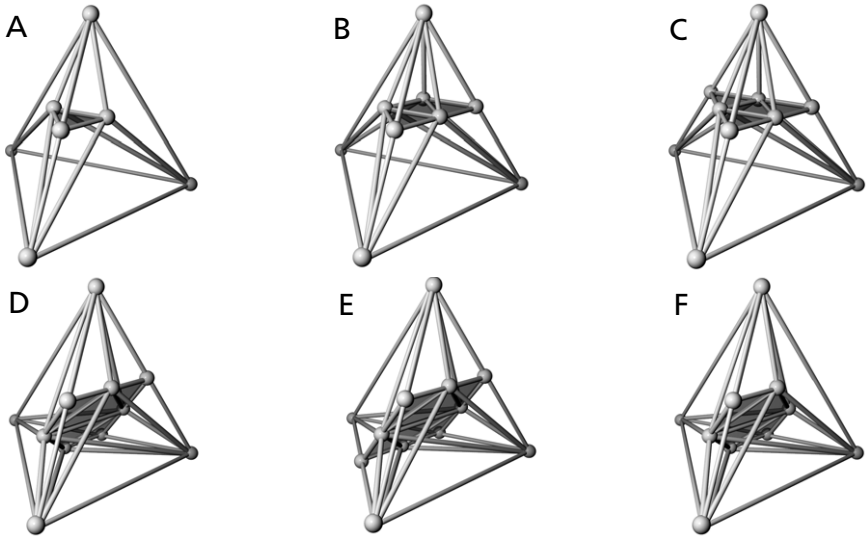


Fig. 2.14 Progressive construction of the tetrahedral subdivisions borrowed from the original paper (Bielser et al. 2004). In one continuous cutting process, different subdivision scheme applied depends on the number of intersected edges.

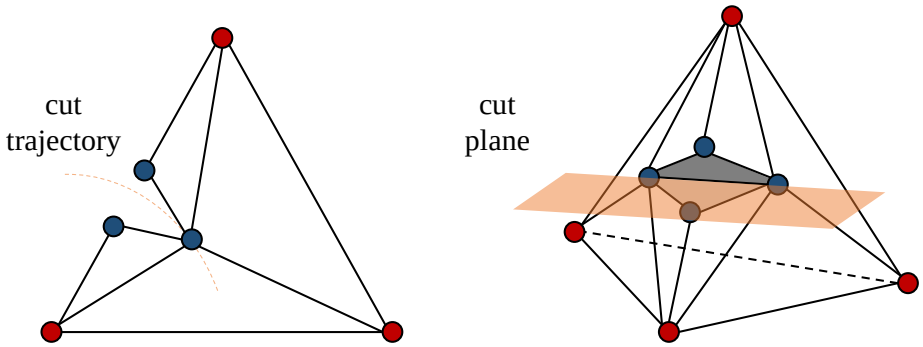


Fig. 2.15 Mesh subdivision performance comparison. The blue nodes denote the new mass-nodes. The tetrahedral volumetric mesh (right) requires more mass-nodes and faces when one edge is intersecting with the cut plane when compared with the triangular surface mesh (left).

both surface-based models and volume-based models, the mesh-based subdivision scheme can be summarised in three steps: insert new mass-nodes, reassign the node connectivity, and introduce new faces to represent the discontinuity. For simplifying the comparison we purely consider the case in which only one edge is intersected on both the surface-based model and the volume-based model.

As shown at the left of figure 2.15, the cut trajectory terminates inside the triangular mesh, when only one edge intersects with the simulated cut blade. Three mass-nodes are generated on with two on both sides of the cut trajectory and one at the termination position. Hence four new faces replace the original mesh face to represent the mesh discontinuity. The right side of figure 2.15 which depicts the subdivision case when one edge of the tetrahedral mesh is intersected is described in Bielser's original work (Bielser et al. 1999). Compared with the triangular mesh, an extra mass-node is created as there are two faces which are affected. Additionally in the

tetrahedral volumetric mesh, more triangles are required to represent the outer shell of the generated volumetric sub-elements.

Moreover, constructing a single volumetric typically requires more computational cost than the surface ones. We compare the computational complexity with the simplest linear plane element (triangle with three mass-nodes) and the simplest linear volumetric element (tetrahedron with four mass-nodes). Noting that we assume both elements employ linear elasticity which means that the deformation can only be generated along the edges connecting adjacent mass-nodes. The first-order Hooke's law (2.18) which describes the response of a linear spring to the applied forces that act in the direction of the spring is

$$F = kX \tag{2.18}$$

where F and X represent the applied forces and the free nodal displacement respectively. The constant k constant describes the stiffness of the spring. Both triangular and tetrahedral element extend the linear Hooke's law to represent the increased number of degrees of freedom. Figure 2.16 illustrates the nodal degrees of freedom for both a linear triangular element and a tetrahedral element. The element stiffness matrix for a triangular element is

$$\left[K_e\right] = \begin{bmatrix} \begin{bmatrix} K_{11} \\ K_{21} \\ K_{31} \end{bmatrix} & \begin{bmatrix} K_{12} \\ K_{22} \\ K_{32} \end{bmatrix} & \begin{bmatrix} K_{13} \\ K_{23} \\ K_{33} \end{bmatrix} \end{bmatrix} \tag{2.19}$$

where each sub-matrix is a 2×2 matrix representing the two degrees of freedom for each nodal displacement. For a tetrahedral volumetric element, the element stiffness matrix K_E is calculated as

$$\left[K_E\right] = \begin{bmatrix} \begin{bmatrix} K_{aa} \\ K_{ba} \\ K_{ca} \\ K_{da} \end{bmatrix} & \begin{bmatrix} K_{ab} \\ K_{bb} \\ K_{cb} \\ K_{db} \end{bmatrix} & \begin{bmatrix} K_{ac} \\ K_{bc} \\ K_{cc} \\ K_{dc} \end{bmatrix} & \begin{bmatrix} K_{ad} \\ K_{bd} \\ K_{cd} \\ K_{dd} \end{bmatrix} \end{bmatrix} \tag{2.20}$$

where each sub-matrix is a 3×3 matrix representing the three degrees of freedom for each nodal displacement (in the direction of u_n^E, v_n^E, and w_n^E, where $n = a,b,c,d$.).

2.6 Surface-based Models Cutting Problems

In this section, we will first describe the necessary steps for cutting surface-based models. These includes the definition of the cut path, the existing technologies for representing the mesh discontinuity, updating the numerical solution due to the cuts, and the simulation of the dynamic

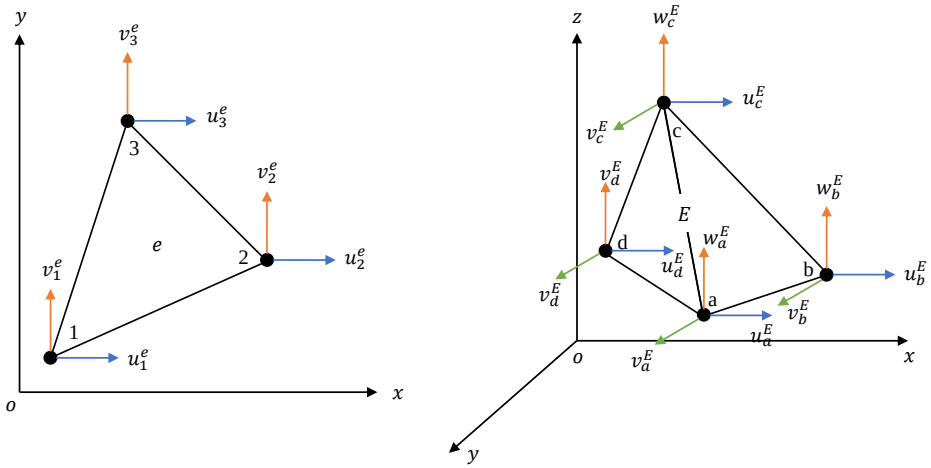


Fig. 2.16 Nodal degree of freedom comparison. The left figure illustrates the nodal degree of freedom under in-plane deformation for the triangular surface element e. The two displacement vectors u_i^e and v_i^e represent the node i ($i = 1,2,3$) displacement components. Similarly, the nodal displacements of the tetrahedral element E are determined by the three components u_j^E, v_j^E, and w_j^E, where $(j = a,b,c,d)$.

behaviour of the object. These steps are also common to the cutting of volumetric models, but we will focus on the difficulties and challenges for the surface-based models associating with its numerical performance and functional capability in practical applications.

As the intersection detection technologies used in all cutting simulations are fundamentally similar, we omit discussion of this aspect. However, a good review of the collision detection technologies used in deformable objects can be found in Teschner and his colleagues' publication (Teschner et al. 2005).

2.6.1 Basic Steps for Surface-based Models Cutting

The definition of cut path. Cutting techniques, when applied to a mesh-based model, start by defining the cut path before identifying which edges are to be considered as part of the cut and how such edges and associated vertices should be managed to produce the desired cut effect. Such an approach can be carried out progressively or non-progressively. In the progressive cutting process, the simulated blade is usually associated with an active point, and the cut incision is generated during the cutting process. The accuracy of the cut edge depends on the number of active sample points. Besides, the cut path can hardly be perfectly straight or curved line as it is manually drawn by users. In contrast, the non-progressive approaches deals with the entire cut path at a time. The cut algorithm can process after the intersection detection to the cut path and all the edges in the mesh. In this manner, the actual cut trajectory is approximated by the all intersection points. Furthermore, the smoothness of the cut path can be increased by adding more sampled nodes on the real cut path.

Mesh discontinuity representation. The following discussion considers how to visually represent the mesh discontinuity produced by the cut incision. Figure 2.17 summarises the cut algorithms

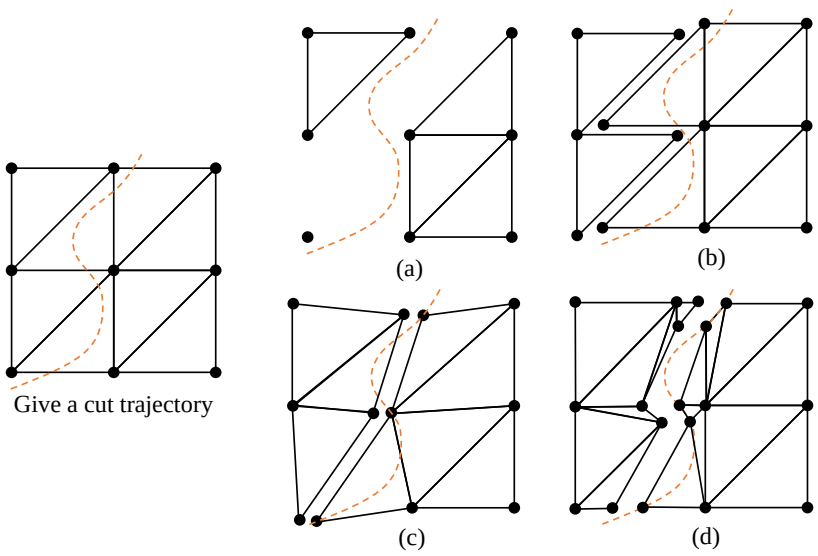


Fig. 2.17 Four types of cut algorithms that have been introduced in previous research. (a) Remove intersected elements (Cotin et al. 2000). (b) Separate along existing elements (Nienhuys & van der Stappen 2000). (c) Snap the nodes and then Separate along existing elements (Nienhuys & van der Stappen 2001). (d) Refine affected element and separate the edges connecting intersection points (Manteaux et al. 2017).

developed by past researchers. The most straightforward way as illustrated by (a) merely removes the affected element so that the generated gap between elements represent the cut edge. In this manner, no new mass nodes or elements need to be introduced and the accuracy of the cut edge depends on the resolution of the original mesh. Another approach without generating new elements is to separate the existing elements in the original mesh as shown in (b). Such an approach open the cut along the nearby existing nodes or edges. In such an approach, the elements sharing the sampled cut edge need a connection rearrangement with the generated nodes. An alternative way is to realign the nearby nodes to the cut path before separating the existing edges as illustrated in (c). Such a technique also known as *snapping* or *node snapping*. Unlike the approaches introduced above an alternative refinement cut algorithm works on the affected elements directly as shown in (d). In such an approach, the intersected element will be subdivided into two parts based on the cut plane. The topological mesh on both sides need to be reconstructed by introducing mass nodes. One essential step is to rearrange the node connections in the range of affected topology.

Numerical Update and Time Integration. The cutting algorithm usually removes or introduces elements and mass nodes in order to satisfy mesh discontinuities and results in a topological modification. Such changes invalidate the global equation system which assembles all elements equations in the previous simulation step. It is therefore crucial to update the mechanical properties and the global equation for the entire simulated object after any cut. For the cutting of FEM-based models the update primarily involves deleting the contributions of the elements that have been removed from the global stiffness matrix.

In addition to finishing the calculation due to the cuts, the continuous motion needs to be obtained to represent the dynamics of the simulated objects in the same time step. This is common in all physically-based animations. The time-dependent form of the governing motion equation (2.17) for a node which has unit mass can be expressed as

$$\ddot{\mathbf{x}} = F(\mathbf{x}, \mathbf{v}, t) \tag{2.21}$$

where $F(\cdot)$ is the global function given by the model. Utilizing the relationship

$$\dot{\mathbf{x}} = \mathbf{v} \tag{2.22}$$

the equation (2.21) can be written as

$$\dot{\mathbf{v}} = F(\mathbf{x}, \mathbf{v}, t) \tag{2.23}$$

Given a nodal position $\mathbf{x}(t)$ and velocity $\mathbf{v}(t)$ of a node at time t, our goal is to determine the new position $\mathbf{x}(t+\Delta t)$ and velocity $\mathbf{v}(t+\Delta t)$ at the next time $t+\Delta t$. Two numerical integration methods can be implemented to solve this problem: implicit integration method and explicit integration method (e.g. Euler's method). The straightforward approach is to rewrite the time derivatives $\dot{\mathbf{x}}, \dot{\mathbf{v}}$ using the finite differences calculated as

$$\dot{\mathbf{x}}(t) = \left[\mathbf{x}(t+\Delta t) - \mathbf{x}(t)\right] / \Delta t \tag{2.24}$$

$$\dot{\mathbf{v}}(t) = \left[\mathbf{v}(t+\Delta t) - \mathbf{v}(t)\right] / \Delta t \tag{2.25}$$

Substituting these into the equations (2.22) and (2.23), the quantities at the next time step $t+\Delta t$ can be obtained explicitly from

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t \mathbf{v}(t) \tag{2.26}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \Delta t F(\mathbf{v}(t), \mathbf{x}(t), t) \tag{2.27}$$

Conversely, an *implicit* integration method defined by equations 2.28 and 2.29 can be used to obtain the quantities at the next step

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t \mathbf{v}(t+\Delta t) \tag{2.28}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \Delta t F(\mathbf{v}(t+\Delta t), \mathbf{x}(t+\Delta t), t) \tag{2.29}$$

2.6.2 Challenges in Surface-based Cutting Models

The accuracy of the cut path. As mentioned in section 2.6.1, the cutting process can be classed as progressive or non-progressive. The progressive method allows a responsive effect to the on-going act of cutting which is desired in particular applications such as in a virtual surgery simulator. However, there is a computational overhead to realigning meshes dynamically in order to provide a realistic cut (with no delay) as the cutting tool moves through the mesh. Conversely, the non-progressive approaches can present a cut in its entirety at any desired point in a simulation.

In both a progressive or non-progressive cutting process, a set of active sample points comprise the cut trajectory. The precise visual representation of the cut incision usually requires that the generated incision closely coincides with the real cut path. One can achieve greater accuracy by duplicating each active sample points point to ensure the simulated incision matches the smoothness of the actual cut trajectory. However this may lead to an expensive computational cost especially when the incision is long. Conversely without introducing new mass-nodes, the cut edge can be poorly represented in meshes which have less elements. A simulated solution is required which can ensure the accuracy of the cut edge while maintaining a reasonable computational overhead.

Volume Mass Calculation. Figure 2.17 illustrates how a cut algorithm can solve for the intersected edges and elements, and this indicates the capability to accurately represents the cut. By simply removing individual edges and vertices (tensor-mass, mass-spring system) or whole elements of the material (FEM), non-uniform holes will present themselves in the mesh topology. Re-cutting in this manner would eventually result in there being no mesh left. Although edge splitting (which may include vertex snapping) does not reduce the number of elements, on a relatively coarse mesh, it can often lead to aesthetically jarring results. Comparing the first three methods illustrated in figure 2.17, the re-meshing technique will provide a highly-detailed cut edge even in a low-resolution mesh. Unfortunately, there is no general re-meshing solution for a mesh which can be discretised by any type of topology. Especially, in finite element method the order of the global stiffness matrix grows due to the increase of new elements. Therefore, a cut method which includes re-meshing techniques is desired to provide precise cut incisions. Such a method must ensure a controllable increase in the number of new elements so that the computational cost remains affordable.

Achieve a Cut in Real-Time. In most commercial cutting applications, there is only a limited time available for solving and updating the equations of the dynamic system. In particular, interactive cutting simulations always require the overall process to be operating in real-time. Figure 2.18 illustrates how the offline and real-time simulation works. When applying a cut to a material model in off-line simulations (cases (a) and (b)) any computational operation for the

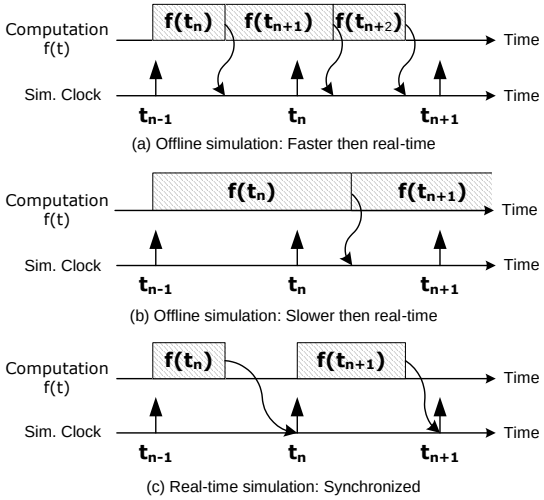


Fig. 2.18 The explanation of off-line and real-time in virtual simulations. The original source from Belanger et al. (2010).

next time step will start when the calculation and update due to the cut are finished. In this manner the time step can be either longer or shorter than the one in the real world. Such situations will make the simulation too slow or too fast compared with those in the real world.

Conversely, real-time simulations require that all the computational operations to be finished before the next step. Even if the computational time is faster than the given time-step any idle-time following the completion is lost and the system has to wait for the next time-step. Therefore, a challenge for such simulations is to determine and employ a suitable time-step which ensures an insignificant deviation between the simulated result and its corresponding physical counterpart's responses in the real world.

Mesh quality. In mesh-based cutting simulations, the mesh quality regarding the element shape influences the numerical stability of the whole simulated system. Recalling the generalised elastic models described in equation (2.18), the global stiffness matrix for the entire simulated system processes most accurately when the condition number is kept as low as possible. When the condition number is exceptionally high the matrix is ill-conditioned. Such ill-conditioned matrices can slow down the numerical solvers or introduce large computational errors into the results (Shewchuk 2002). Usually the elements in the model whose aspect ratio are extremely high can cause an ill-conditioned matrix. However, in mesh-based cutting simulations, corresponding to the actual cut trajectory, ill-shaped elements always need to be introduced with the chosen intersection nodes.

The challenge here for mesh-based refinement is to handle an ill-conditioned global stiffness matrix effectively. Busaryev et al. (2013) employs a global constrained Delaunay-based refinement to reduce the number of ill-shaped elements. This work does not provide real-time interaction. It is easier to implement when considering the whole surface post-cut. In order to reduce the condition number of the global stiffness matrix, the common technology is to introduce preconditioner P which is an approximation of the matrix A to the linear system in the

form of $Ax = b$, i.e., $P^{-1}Ax = P^{-1}b$, such that $P^{-1}A$ has a smaller condition number than A. Therefore, faster convergence can be obtained by solving the equivalent system $P^{-1}(Ax) = P^{-1}(b)$. This technology has been successfully implemented in real-time cutting simulation (Courtecuisse et al. 2014), assuming that the model is under small deformation. In this case, the built preconditioner is insensible to geometrical non-linearities. Koschier et al. (2017) further reduces the condition number by constructing a diagonal preconditioning matrix in which each magnitude is the scaled diagonal entries of the matrix. This method is adequate when the condition number is within a range. When the condition number is extremely small (e.g. between 0 and 1), this method can easily cause negative entries that crash the simulation.

The Allowed Cut Complexity. A high-performance cutting simulation should also carefully handle the complicated cut incision and the ability to recut (this may be in the same area. However, such a complicated cut incision requires that the cutting simulations allow a recut in the same area without losing) volume; otherwise there is no mesh left when applying cuts many times. Besides, as a cut may start or terminate inside an element, the re-meshing of the start or end elements needs to be adequately handled. For higher-level reproduction of realistic cuts, the simulations are required to cope with overlapping cuts (such as cutting a folded paper). This requires that the cutting algorithm should detect precise intersections in multilayer models. Moreover, integrating such technology into real-time cutting simulations is still a challenge.

2.7 Related Works

Various kinds of actions can cause the changes of the material objects. These actions include cutting, tearing and fracture, etc. Although this dissertation is focused on cutting simulations it is worth looking at the technologies behind other simulations resulting in the mesh discontinuities of simulated objects. In this section, we aim to provide a comprehensive overview of the state of the art for mesh discontinuity technologies. We pay particular attention to the applications employing surface-based models. However, we will also review and discuss the significant volumetric cutting and fracture technologies that reflect the analogous problems for surface-based models. This section is divided into parts based on the surface-model cutting problems described in Section 2.6.2, and we will review and discuss how these problems have been considered or resolved in past research. We will select the most significant works in surface-based model cutting, as these may concern and resolve more than one problem.

2.7.1 Cut Incision Reproduction Technologies

Skin cutting and slide. Lindblad & Turkiyyah (2007) propose a real-time progressive cutting method for skin cutting and wound closure simulations. They employ a surface model and an underlying tissue model to simulate skin layers and introduced constraints between layers. They also employ Extended Finite Element Method (XFEM) introduced by Belytschko & Black (1999) to achieve the cuts along arbitrary surfaces. In the surface model they duplicate the nodes

and each node and their duplicate are initially connected by constraints and have the same positions. Their method then removes such constraints (between the surface and the underlying tissue model, between each nodes of the surface model and their duplicate) when separating the surface model from the tissue to simulate skin sliding and cutting close to the nodes of the surface model. Conversely, when cutting through arbitrary element of the skin model, the original affected elements are locally re-meshed along the cut trajectory while introducing new degrees of freedom to represent the cut edge. They adopt an augmented global stiffness matrix using discontinuous basis functions (similar to XFEM) to efficiently handle the expensive updating due to the element deletion and addition.

This approach provides an efficient cutting strategy in terms of computational efficiency. It simplifies the problem of cutting through layered models with element deletion. However to avoid generating a jagged cut edge the resolution of the initial cutting mesh needs to be sufficiently high. Although the local re-meshing scheme mitigates the computational overhead since there is no need to reconstruct the whole mesh, it cannot guarantee the mesh quality as it moves along the actual cut trajectory without further mesh optimisation. It is simple and straightforward to implement when only need to simulate the cutting process; however it is not adequate when the high-detailed cutting is required.

Complex Cutting. Kaufmann et al. (2009) proposed a method that allows complex cuts , including multiple, crossing and partial cut edges in a single element. They implemented such a method in low-resolution meshes when considering an underlying thin simulation domain. When applying single or multiple cuts, their so-called enrichment texture stores different values(-1 or 1) for each texel (texture pixel) of affected elements. Afterwards, the basis functions for these elements are enriched by the extended finite element method (XFEM), which split the elements along the desired discontinuity edges (detected by the enrichment texture). Additionally when implementing the geometry shader, the method visually represents the cut edge by duplicating the affected elements and rendering once on each side of the cut.

This technique represents the cutting path by changing the texture value on both sides rather than modifying the underlying geometrical mesh. In this manner this technique largely reduces the computational cost caused by the re-meshing. However, the resolution of the enrichment texture sampled on the simulation mesh limits the accuracy of the cut edge. Also, when the basis functions for an element are split and enriched, the physically related material properties are not adequately projected onto the new basis. Consequently, it is still a challenge to integrate such technology into physically-based cutting simulations.

2.7.2 Re-meshing or Refinement Approaches in Cutting and Tearing

Node snapping. Serby et al. (2001) introduce a so-called node snapping technology in the cutting simulation for a surface-based model. The method first divides the cut line into segments by introducing cut points. The number of the cut points is incrementally adapted (decreased) ensuring each cut point has unit nearest node. Subsequently, the nearest nodes are displaced due

to the position of the corresponding cut points. New nodes are only generated at the positions of the filtered cut points to separate and represent the cut edge. This technique is also implemented in a volumetric model (Nienhuys & van der Stappen 2001, Lim et al. 2007).

This method successfully satisfies the real-time requirement primary because it avoids a substantial increase in the number of generated nodes and elements during the re-meshing operation. It can also reproduce a smooth cut edge without requiring a higher degree of freedom for the simulated mesh. However, it tends to result in mesh distortion since the adjacent elements are always influenced by the element whose nodes are dislocated. Although the method introduces a homogenization process in which the mass-spring strategy is combined into the distorted element (nodes are treated as mass points, and edges as springs) it is far from adequate since the inter-dependence of the parameters are not taken account (e.g. spring constants).

Refinement for Mass-Spring System. Bruyns et al. (2002) present an efficient cutting method based on the refinement technology for mass-spring system models. They implement the technique in both triangular surface and tetrahedral volumetric models whose deformation is governed by linear elasticity indicating that the deformation can only occur along the direction of each spring. They progressively detect the intersected elements and introduce middle-nodes when the simulated blade moves to the middle of the elements. After the blade cuts through an entire element, the neighbour will also be refined ensuring a conformal mesh. The affected springs are removed, and new springs and vertices are generated to model the holes on both sides of the cut trajectory.

The method successfully satisfies the real-time simulation requirement due to the fast calculation and updating of the mass-spring system. It improves the interactive realism by adopting the progressive cutting process while it also simplifies the problem by only performing the refinement operation when cutting through springs. However, the stability of the elastic deformation of the model heavily relies on the mesh structure. Moreover, the material properties (such as spring constant) of the new springs are difficult to control. The spring-based model is therefore limited in terms of its ability to reproduce realistic simulation when applying cuts.

Adaptive re-meshing. Pfaff et al. (2014) present an adaptive re-meshing approach for the force-based fracture propagation and customise-shaped tearing on variety of thin sheets. They implement the method using conformal triangular mesh, in which the visualisation, collision and simulation are represented by the same topological mesh. When the applied external forces exceeds a threshold the mesh will be broken and the elements around the broken edges will be refined (divide edges and introduce new triangles). The refinement will terminate when the stress distribution around the fracture tip is well resolved.

They provide an effective and flexible approach as it reduces the numbers of element when not necessary. It worth noticing that in their tearing simulation which allows arbitrarily-shaped tear edges, they first divide the elements at the corner of the target shape by duplicating the corner points, and refining the area involved in order to remove all the ill-shaped elements. The extra refinement here is necessary as the first time re-meshing does not consider the ill-shaped

elements; however it results in higher computational costs. Such problem can be properly resolved by implementing an adequate refinement scheme such as the Delaunay algorithm.

2.7.3 Real-Time Cutting Applications

Cutting on Free-Form Deformation models. Sela et al. (2007) produce a real-time interactive cutting simulation on surface model. They adopt a synthetic model, so-called FEM-based discontinuous free-form deformation (DFFD) to satisfy real-time performance and mesh modification due to the cuts. The method first pre-processes a cut using a FEM-based surface model and records the nodal displacement at every time step. The data is then encoded into a DFFD deformation surface model, which will be used for the real-time cutting simulation. When applying a cut on the DFFD surface model, the polygons are split along the cut trajectory to represent the incision. In the meantime, the encoded nodal deformation data become active around the incision to provide instant deformable behaviour.

This method achieved a real-time cutting simulation, utilising the accuracy of the FEM-based deformation and the speed of the DFFD strategy. The DFFD extends FFD (free-form deformation) that supports discontinuous generation in a continuous domain, and it drives the deformations of the geometry mesh by applying deformation functions. The proposed method encodes the accurate deformation data as the parameters into the deformation function of the DFFD model. However, it is far from adequate for cutting simulation, in which the model may be cut several times. This is primarily because the pre-processed data can only be encoded once. Consequently it is impossible to encode proper information for the geometrically modified model since the location and orientation of the mesh is unknown during the preprocessing stage.

Cutting on XFEM-based models. Turkiyyah et al. (2011) present an XFEM-based method to compute and represent the deformation of the simulated surface model at interactive rates when applying cuts. The method decouples the rendering mesh from the computational mesh which is used for the deformation calculation. After applying a cut, the rendering mesh linearly subdivides into each of the affected elements, while the connection in the corresponding computational mesh remains constant. Afterwards, the new nodes are introduced to open the incision. Discontinuous basis functions are introduced to the original basis functions to represent the deformations of the generated sub-domain.

The proposed approach mitigates the heavy computational burden during the re-meshing scheme as it keeps the computational mesh that is used for the deformation of the geometry. They authors progressively update the global stiffness matrix using an incremental approach to ensure a minimal time burden. However, this sacrifices the physical realism at cut edge whose deformation is not independent as it is restricted by the associated continuous computational mesh. In addition, partial cuts within a single element or multiple cuts are impossible with this method.

2.7.4 Mesh Quality Assured Cutting Algorithms

Edge flip. Nienhuys & van der Stappen (2004) proposed a method to produce cuts in the triangular surface model by introducing a minimal number of elements while maintaining the mesh quality. They implement a progressive cutting process, in which the movement of the so-called active node represents the cutting trajectory of the simulated scalpel. During a continuous cutting, the active node always attaches to the existing edges, and the affected element will be subdivided with the active node. In this step, any existing nodes close to the active node will be removed. The authors improve the quality of the generated sub-elements by detecting and flipping the edge that is associated with the ill-shaped elements and their neighbour. Afterwards, they split the larger-sized sub-elements in order to increase the smoothness of the incision edge. They also generalise the method to curved surfaces to handle bifurcations and annihilation incisions.

In the progressive cutting process, the standard Delaunay triangulation is not applicable as the incision boundary could be non-convex. To simplify the problem, the authors implement one essential step in the standard Delaunay triangulation which is the so-called edge flip to improve the shape of bad elements (in their experiment, it refer to the triangles with small angles and short edges). The method may change the shape of the triangular elements to increase the minimum angle but not the number of elements. Unfortunately, the method is only implemented on static models, which is not adequate for cutting realistic virtual deformable objects. In addition, the accuracy of the physically-based animation requires high-quality meshes. However, the motion of the model does not obey the law of physics after cuts.

Local adaptive re-meshing. Wicke et al. (2010) present an adaptive local re-meshing scheme addressing both elastic and plastic material behaviours. The scheme is implemented and illustrated on tetrahedral elastic and plastic solid models. It increases the number of elements in the mesh where there are large deformation with so-called topological transformations. Specifically, this involves removing the element that needs to be re-meshed, and introducing new sub-elements to fill the same space. Also, the number of elements is decreased when there is no need for extra details. A local optimisation operation ensures the mesh quality is not worse than the state of the mesh before any cut.

The local re-meshing and optimisation operation improve the mesh by introducing fewer degrees of freedom than the global approach that reshapes the whole mesh even though only a portion of the mesh is involved. This motivates us to evolve such local operations when considering discontinuities due to cutting. It also ensures the numerical stability of the simulated system by mesh optimisation, which operates edge flip after the *node smoothing* which moves a node to improve quality of the elements adjoining it (de L'isle & George 1995). However, the method is not designed for the precise cutting behaviour and so the generated cut incision may not be able to accurately reflect the actual unpredictable cut trajectory. Primarily due to that the elements that involved the edge-flipping operation may result in inaccurate cuts when the edge is a part of the cut boundary.

2.7.5 Cut Complexity

Cutting on the VNA-based model. Molino et al. (2004) introduce a virtual node algorithm (known as VNA) to reproduce an arbitrarily-shaped cut edge in the single element without generating ill-shaped elements that influence the numerical stability of the simulated cutting system. When applying a cut, each element in the material will be duplicated. The number of the replicas is determined by the number of broken parts within the element due to the cut. Afterwards, the cut edge is visually represented by passing a portion of the texture of the material model to each replica. Each replica consists of actual texture of the material model and empty regions. They implement the method in both thin-shell and tetrahedral models. They restrict linear cut edge within an element, and the position and velocity of the intersection points are calculated via linearly interpolating these values from the original element.

Wang et al. (2014) improve the original VNA technique introduced above in several aspects. They illustrate the method using a rigid tetrahedral model. The original VNA restricts every tetrahedral element can be cut into maximum of four subregions that must associate with one node of the original tetrahedral element. Their method allows one tetrahedral element to be split up into 24 sub-tetrahedra by introducing a centre node for each triangular face and edge. Each embedded copy is divided after being assigned a segment of the original element aligned with the cutting edge. In this manner, the portion with empty fragment is removed. Sequentially, the collision with external objects becomes more accurate and robust instead of using the duplicated well-shaped element (in which only partial materials holds).

The virtual node algorithm (VNA) is capable of representing arbitrarily-shaped cut incisions within the single element. However the maximum number of duplicates for a single tetrahedral element still limits the number of cuts in each element. Such available split of tetrahedra is increased from four (in Molino's work) to 24 (in Wang's work). It is also still a challenge for the VNA-based algorithm to satisfy the requirement for a high-quality mesh and accurate collision detection simultaneously. Specifically, Molino prevents the ill-shaped elements by embedding them into well-shaped duplicated elements, while the collision is not accurate as only the texture information is assigned into the boundary of each portion but not physically attributes. Wang's work resolved this problem with adaptive subdivision to ensure precise collision detection. However, the subdivision may result in ill-shaped elements which is not acceptable in physically-based deformable models.

2.8 Conclusion

In computer graphics applications, most approaches use a unit mesh for both deformation and visualisation representation. When adopting such a method the mesh modification due to the cuts results in the reconstruction for both deformation and visualisation. However, the physically related properties of each portion of the model are accurate even under modification. The cutting technologies in this field can be generally classified as element removal and change the element

which includes snapping or refinement. Simply removing the affected elements efficiently represent the discontinuity, satisfying the need for time efficiency in real-time or interactive simulations. However, the initial resolution of the mesh restricts the accuracy and details of the incision, which has always been handled in re-meshing or refinement schemes. In contrast, the increase in the degrees of freedom and the mesh quality usually pose a challenge for re-meshing approaches. This usually requires the introduction of an optimisation operation, which can either process locally (constraint the optimised region) or globally (for the whole mesh).

The alternative approach is to decouple the deformation and visualisation representation with different meshes, which will then be embedded to constitute the unit simulation system. It allows different resolutions to discretise the geometrical behaviours and the rendering. The XFEM and virtual node algorithm (VNA) are mostly used when considering cutting operations. In FEM-based models the equations constructed for each element is continuous, XFEM enriches such continuous equation with equations to represent the discontinuities within elements. In contrast, the virtual node algorithm usually employs a lower resolution mesh for the collision and deformation, with a higher resolution for the geometrical construction. Such approaches allow an arbitrarily-shaped cut edge by employing supporting rendering technologies such as texture-based enrichment. However, the challenge is that such techniques are only pursuing visual plausibility, but not the complexity of physical behaviours. This motivated us to invent an optimal framework which allows detailed cuts pre-defined by users in complex cutting scenarios, meanwhile eliminating ill-shaped elements in order to achieve the cuts in real-time robustly. In the next chapter, we will introduce the proposed framework in details.

Chapter 3. Realistic Cuts of 2D Physically-based FEM Modeling

In this chapter, we provide an interactive framework that allows precise cutting on the surface of a deformable model while retaining the correct physical behavior of the underlying geometric model. Considering that non-uniform cutting will inevitably produce ill-shaped elements, these elements will lead to incorrect physical behaviour and even collapse the system. To this end, we have attempted to address this problem from *geometrical* and numerical perspectives. From a geometric perspective, we propose a new refinement scheme, which combines the Delaunay mesh generation method and an efficient local optimisation operation. Furthermore, we observe that ill-shaped elements are caused by the ill-conditioned global stiffness matrix. This motivates us to deal with this problem by providing *numerically* stable solutions to the finite element equations.

The proposed framework brings two main benefits:

- Accurately representing the shape of a cut given by users by refining the elements that been cut through and their neighbouring elements. This restricts the computational cost due to the refinement focusing on the areas of material model closest to the cut in order to achieve real-time cutting. Non-linear cuts within a single element are allowed to provide smoother cuts.

- Maintaining the quality of the modified mesh without causing substantial computational cost by introducing an efficient local optimisation method. This enables the physical accuracy of the simulated material after the cuts.

The refinement scheme is implemented and illustrated in an elastic FEM 2D model which is discretised by triangular elements. This chapter outlines and discusses the main contributions of this dissertation. In such simulations, the tasks that have to be achieved in order to reproduce a physically accurate solution in real-time are:

- **Accurate Cut Reproduction.** An interactive cutting simulation always allows users to cut the model. Such a simulation needs to cut the model where the user defines the cut path.

- **High Mesh Quality.** Cutting on FEM-based models always requires to alter elements or introduces new elements. Elements with different sizes and shapes can be generated in order to represent an arbitrarily-shaped cut path. For a triangular mesh, either thin (i.e. needle-like) or flat triangles are treated as ill-shaped. Such triangles slow the convergence

speed of iterative methods which are used to solve the system of finite element equations (Dey et al. 1992, Shimada & Gossard 1995). For physically-based FEM models, such triangular elements can reduce the realism of the physical behaviours.

- **Achieving a Cut in Real-Time.** A real-time cutting simulation is in great demands in both research and industries. It allows the model acts and reacts without noticeable delay. The *Frame Per Second* (FPS) is typically used in games and films to describe how frequently the sequence of a frame is presented to the viewers. This is the same measurement as *Hertz* (Hz) which is a unit describing the number of cycles per second, i.e. 1 FPS = 1 Hz. In virtual surgeries, at least 25 Hz is needed to provide a realistic forecast of the results of a given action to the model (Cueto & Chinesta 2014). Films typically run at 24 FPS. Video games run at 30 FPS or 60 FPS in North America and Japan, respectively while in Europe and most of the rest of the world games run at 50 FPS (Gregory 2017). Therefore, a practical framework which aims to reproduce cuts in real-time should run at least 25 PFS.

This chapter will discuss and resolve these tasks in turn. In each task we analysis the challenges and provide a solution with a discussion.

3.1 Accurate Cut Reproduction

An accurate cut path representation is crucial to simulate a realistic cutting process. The simulated cut path is required to closely estimate the actual cut trajectory and also allows an arbitrarily-shaped incision during a cut. As discussed in section 2.6.1, the progressive cutting process detects the instant position provided by users. In such a situation ensuring a smooth incision is difficult to control. Therefore we employ a non-progressive process in which the cut occur after the whole cutting is given. This allows smooth and arbitrarily-shaped incisions (e.g. a straight line or a curve) that can be hardly handled in progressive process. The actual cut trajectory can be accurately reproduced if there is an adequate number of sampling points on the path. However it can be time-consuming when the number is large. Conversely, we detect the intersections between elements and the cut path at edges and only add nodes at the intersection points to keep the number of sampling points low. For each intersected element, if the line connected by the intersected points is longer than the line connected by the middle points of two intersected edges, a point will be added at the place that holds the maximum curvature on the curve segment within the element to smooth the cut.

3.1.1 Locality of the Cut

Non-progressive but arbitrarily-shaped cutting. A virtual cutting simulation starts with applying a cut to the material model. We require an algorithm which can reproduce a smooth arbitrarily-shaped cut at exactly where is desired. This motivates us to employ a non-progressive

cutting process. Recall the discussion in section 2.6.1, the non-progressive cutting process enables extremely clean and clear cuts to be produced. In contrast to the progressive process, the non-progressive cutting method operates the cut algorithm after introducing the entire cut.

Unlike simulating a progressive cutting process, our goal is to provide the users with a pilot study of the cuts which demonstrate the complex cut path pre-defined by users. Moreover, the cut should be smooth and can be infinitely redefined. The typical strategy used in progressive cuts is using polylines which are obtained by connecting all sequenced vertices during the cutting. This strategy is straightforward to implement; however, the number of vertices is hard to control. The alternative approach is to define the cuts utilising smooth curves the shape of which is determined by the control points. Hence, the users benefit from defining and controlling a complex cut using few control points; moreover, the number of control points are easier to control than the vertices in polylines, so that is good for storage.

The path of the cut is generated from a cubic Bézier curve , which is widely used in computer graphics applications to model smooth curves. It can be customised by adjusting the four control points. A clear and detailed generation algorithm of the cubic Bézier curve is given by Joy (2000). For the purposes of our simulation a set of four mouse clicks P_0, P_1, P_2, and P_3 are chosen as the control points to identify the location of an incision. Therefore, the position vector B of any point on the curve can be calculated as:

$$B = P_0(1-t)^3 + 3P_1t(1-t)^2 + 3P_2(1-t)t^2 + P_3t^3, \quad t \in [0,1], \tag{3.1}$$

then, the x and y coordinates of the curve point are:

$$\begin{aligned} x(t) &= a_0t^3 + a_1t^2 + a_2t + a_3 \\ y(t) &= b_0t^3 + b_1t^2 + b_2t + b_3, \end{aligned} \tag{3.2}$$

where a_i and b_j are the coefficients of t^3, t^2 etc. calculated from Equation 3.1.

We identify the intersection between each edge of a triangle in the mesh and the cut path using

$$Ax(t) + By(t) + C = 0, \tag{3.3}$$

where A, B, and C are the coefficients of the line along the edge of the triangle. The roots of this cubic equation 3.3 indicate the intersection points if $0 < t < 1$ (i.e. the intersection point is within the boundary of the triangle's edge). Triangles that have only one intersected edge indicate that the cut is terminated inside, while those which have two intersected edges are been cut entirely. Both situations will be described and handled more fully later. The cut may intersect with all the three edges and even more than once with each edge. This is one of the directions for expanding our algorithm in the future.

In order to represent the cuts in which the edge is only inside the simulated material, such as circular or elliptical cuts, we explore the *Rational Bézier curve* which is an extension of the

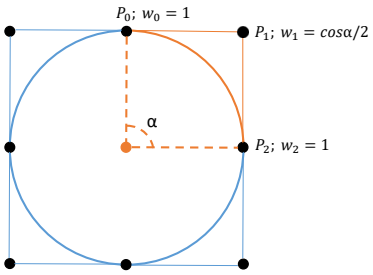


Fig. 3.1 A circular cut can be generated by integrating four quadratic *rational Bézier curves*. The shape of each curve is controlled by three control points (black dots). A quarter-circle can be obtained when $w_0 = w_2 = 1$ and $w_1 = cos(\alpha/2)$ in Equation 3.7.

Bézier curve. The mathematical basis for a *Bézier curve* is the *Bernstein polynomial* initially developed by *Paul de Casteljau (1959)*. With n control points a *Bézier curve* can be denoted as

$$B(t) \quad = \quad \sum_{i=0}^{n} B_i \beta_{n,i}(t), \tag{3.4}$$

where

$$\beta_{n,i}(t) \quad = \quad \binom{n}{i} t^i (1-t)^{n-i}, \quad t \in [0,1]. \tag{3.5}$$

In Equation (3.4), each position vector B_i is multiplied by its weight function $\beta_{n,i}(t)$ which is restricted in a range of $[0,1]$.

Based on the *Bézier curve*, the *Rational Bézier curve* introduces a new weight w_i which acts as an additional parameter to control the shape of the curve more precisely. To represent a *rational Bézier curve*, Equation (3.4) can be rewritten as

$$B(t) = \frac{\sum_{i=0}^{n} w_i B_i \beta_{n,i}(t)}{\sum_{j=0}^{n} w_j \beta_{n,i}(t)} = \sum_{i=0}^{n} B_i \left[\frac{w_i \beta_{n,i}(t)}{\sum_{i=0}^{n} w_j \beta_{n,i}(t)} \right] = \sum_{i=0}^{n} B_i R_{n,i}(t), \quad t \in [0,1]. \tag{3.6}$$

A quarter-circle which is also known as a quadratic *rational Bézier curve* can be obtained by setting $w_0 = w_2 = 1$ and $w_1 = cos(\alpha/2)$, where α is the angle formed by the three control points placed at the three corners of an isosceles triangle (Beach 1991). Therefore, a circle can be obtained by integrating four quadratic *rational Bézier curves* as shown in figure 3.1. Each curve is governed by Equation (3.6) with $n = 3$, i.e.,

$$B_{quadratic}(t) = \frac{P_0 w_1 (1-t)^2 + P_1 w_2 2t(1-t) + P_2 w_3 t^2}{w_1 (1-t)^2 + w_2 2t(1-t) + w_3 t^2}. \tag{3.7}$$

Intersection detection in the 3D environment. To handle complex intersection between the simulated cut path and the dynamic surface model, we introduce an efficient and accurate detection scheme. The process of Delaunay triangulation and incision generation are only valid

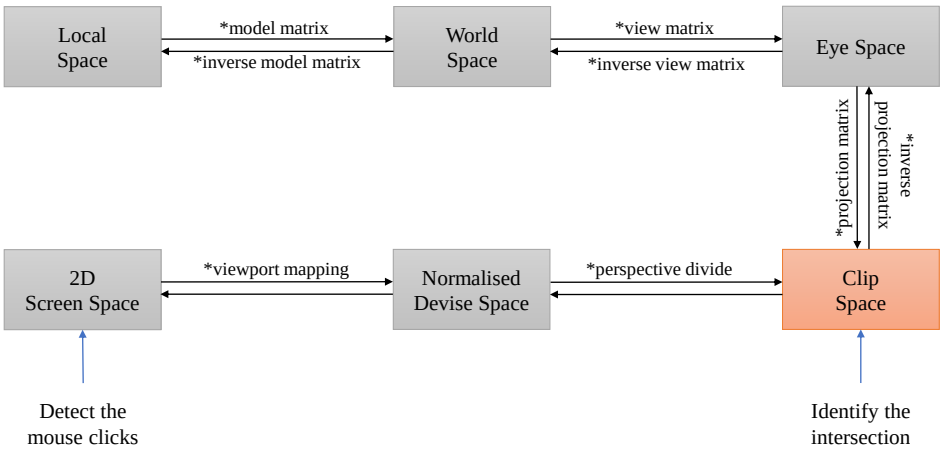


Fig. 3.2 OpenGL transformation pipeline. For our simplified intersection detection in the clip space, we build a 2D version of the deformed model by transferring its local coordinates into the clip space by multiplying the view and projection matrix. Similarly, the detected screen coordinates of the mouse clicks are transferred back to the clip space by multiplying the inverse projection matrix.

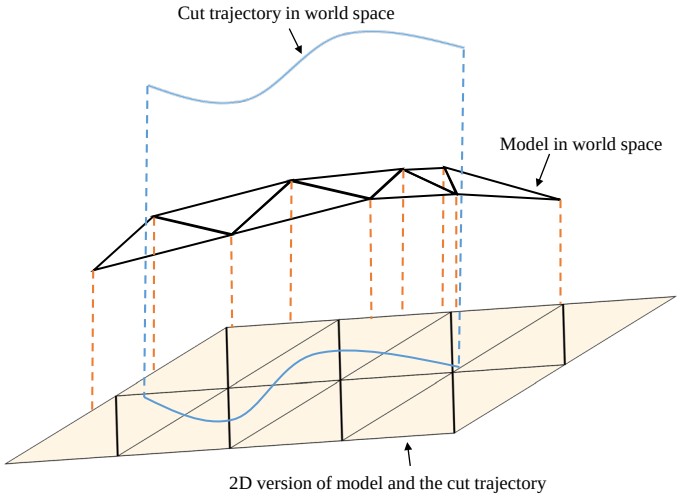


Fig. 3.3 Project the cut trajectory and the model from 3D into a 2D space followed by the intersection detection which operates in the 2D space.

in the local space of the material mesh with no deformation due to external forces. However, the cause of the cut takes place in world space, where the model is likely to be deformed (probable causes are the collision of cloth with a sharp object in the environment or penetration of the cloth by a cutting implement).

To simplify the problem, we project the intersection into the 2D clip space. Figure 3.2 illustrates how we utilise the OpenGL transformation pipeline to simplify the complex intersection in 3D environment. The model coordinates are defined in the local space. Afterwards, the transformation matrices, including model, view and projection matrix transfer it into the final screen space. Utilising the customisable transformation matrices, we build a 2D version of the model by transferring it from the world space to the clip space (the local mesh coordinates multiply the view and projection matrix) as shown in figure 3.3. Analogously, after detecting the

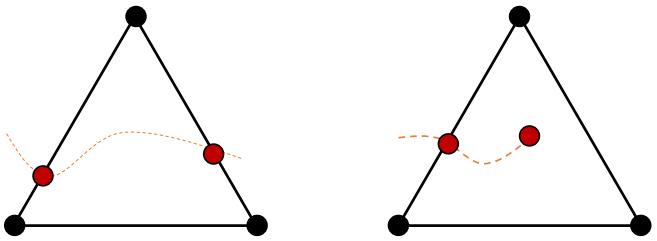


Fig. 3.4 Two cases in surface subdivision: triangle intersect with the cut path at two edges (left), and triangle only has single intersected edge (right). The later case indicate the cut is terminate inside the triangle.

set of four mouse clicks in the screen space, the positions are transferred back to the eye space by multiplying with the inverse projection matrix. In this manner, it simplifies the intersection detection in the 3D environment. Also, the complex intersections, in which the cut path may intersect with multi-layered folded meshes can be easily handled without extra consideration.

3.1.2 Surface Refinement

Validity of the Delaunay triangulation generation. The methods for generating a Delaunay triangulation that identify a group of points as input, always generate a convex triangulation. Such methods are difficult to be implemented directly in cutting simulations primarily due to that a cut may cause a non-convex boundary. One can globally refine the whole mesh using a Delaunay generation algorithm with the added mass nodes at all intersection points, while it may results in a substantial computation cost. To solve this problem, we restrict the Delaunay generation algorithm to be implemented within each element that been cutting through. However, the edges connecting the intersection points are fixed during the generation to respect the actual cut path. We call it conditional Delaunay generation algorithm which will be explained later.

Surface subdivision. The method whereby we subdivide the affected triangle depends on the number of edge intersections. Figure 3.4 shows the two cases considered in the following refinement. As shown in the left half of the figure, the most common case is when two lines of the triangle are intersected by the cut (i.e. the cut originates and terminates outside of the triangle so it cuts completely through the triangle). In this case, we provide two subdivision schemes. For each intersected element, we determine whether the line segment formed by the two intersected points is longer than the one formed by the middle points of the two intersected edges. As shown in figure 3.5, if the cut segment $a_{cut}b_{cut}$ is shorter than a_hb_h the element will be subdivided with the two intersected points only. Conversely, if $a_{cut}b_{cut}$ is longer than a_hb_h as shown in figure 3.6, we add a point at which the curvature of the curve segment (yellow dashed) is maximum. For a parameter curve, the curvature is given by

$$k(t) \quad = \quad \frac{|B'(t), B''(t)|}{\|B'(t)\|^2}, \tag{3.8}$$

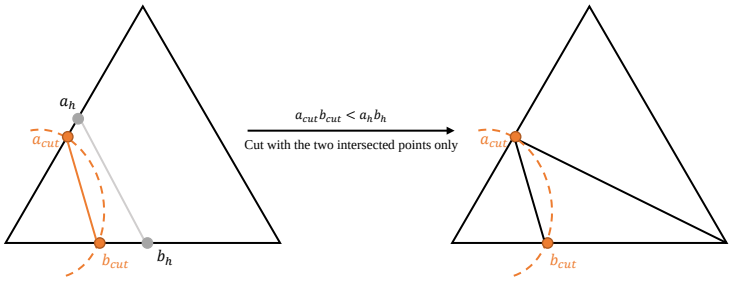


Fig. 3.5 The linear cut in an element when the cut line $a_{cut}b_{cut}$ is shorter than a_hb_h.

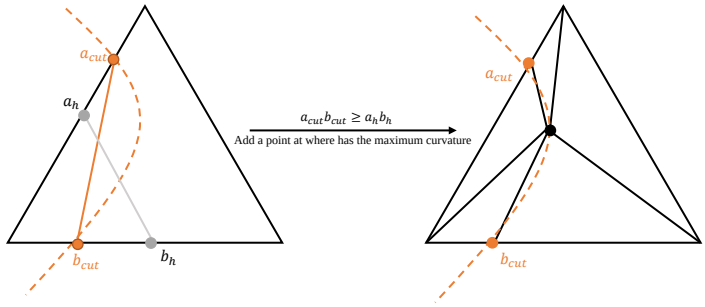


Fig. 3.6 The non-linear cut in an element when the cut line $a_{cut}b_{cut}$ is longer than a_hb_h. A vertex is added at the place where the curvature on the curve segment within this element is max.

in which the maximum curvature of the curve segment can be obtained by finding the maximum value of $k(t)$, where $B(t)$ is the function of the curve and $t \in [0,1]$.

As we have identified the edges of each triangle that are in the path of the cut we can insert an node at the exact point where the cut crosses the triangle edge to ensure the accuracy of the cut incisions. Special care is needed when a triangle only has one intersected edge as shown in the right half of figure 3.4. In this case the cut terminates inside a single element. To accurately reproduce the cut we identify the end position of the cut path in the element and add it to the point group prepared for the Delaunay triangulation generation.

3.2 High Mesh Quality

Cutting a material model makes it discontinuous along the cut. In FEM-based models the mesh always needs to be modified or refined to represent the cut. Such methods are also known as mesh-based cutting methods in the literature. The FEM obtains the behaviour of the whole problem domain by solving and assembling each governing equation that is constructed on a element with numerical methods. However the generated non-uniform element can influence the accuracy and efficiency of these numerical methods. Such elements are referred as ill-shaped elements. Therefore it is crucial for a mesh-based method to avoid ill-shaped elements.

In this section we will explain how an element can be ill-shaped in terms of it's size and shape. We then evaluate how ill-shaped element impact the FEM mesh-based cutting method in relation to the interpolation errors, discretisation errors, and conditioning of the stiffness matrix. As the

measurement criteria for satisfying all of these aspects may be contradictory to each other we finally evaluate and choose the optimal measurement criterion for our experiment. We will explain how our solution addresses these problems based on the chosen criterion. In particular, as triangle is commonly used in modelling surface materials all analysis and examples are based on the triangular elements.

3.2.1 Ill-shaped Element

Interpolation errors. In most computer graphics modelling tasks, the interpolation errors are significant as it compromises the accuracy of the simulated system. Finite Element Method constructs the interpolation function for each element. In this manner the animation at any place within elements can be obtained by the interpolated value. Therefore the accuracy is determined by how the interpolation function g approximates the true function f. The interpolation errors can be classified into two types: the difference between the interpolation function and the true function that expresses the analytical solution of the problem ($f-g$), and the gradient difference between them ($\nabla f - \nabla g$). In finite element method, f describes the displacement and ∇f indicates the strains.

To reduce the difference between the interpolation function and the true function, one can constructs smaller elements to increase the smoothness of the estimation. In cutting simulations, it is reflected in the level of details around the cut edge. Compared with approaches that remove the intersected elements or separate the mesh along the existing edges, a refinement strategy which increases the number of elements while reducing their size can provide more realistic physically-based behaviour around the cut path. However, the computation time is approximately proportional to the number of the triangles in the mesh. So controlling the number of new elements to ensure interactive or real-time satisfaction will be addressed in our method. Conversely the error in the gradient that primarily influences the mesh quality heavily depends on the shape and size of elements in the mesh of a material model. In particular, in finite element method it causes the discretisation errors, which will be described in following.

Discretisation errors. For physically-based FEM modelling, large angles approaching 180° in triangular elements result in discretisation errors that can cause unrealistic physical behaviours. The FEM discretises a continuous problem domain into a finite number of discrete elements, aiming to estimate the unknown solution $f(x)$ of a partial differential equation by a piecewise approximation $h(x)$. The discretisation errors occur when the estimated solution computed by the finite element method is different from the true solution of the problem, i.e., $f-h$. As mentioned previously, the interpolation errors are closely related to the discretisation errors primarily because the discretisation error can converge to zero only when both the interpolation errors $f-g$ and $\nabla f - \nabla g$ approach zero (Shewchuk 2002). Babuška & Aziz (1976) have proved, it is large angles that can make the error in the gradient ($\nabla f - \nabla g$) extremely large when it approaches 180°.

One can avoid this problem by offering uniform subdivision during a cut resulting in sub-triangles with the same shape and size. Ideally, if the intersected elements are sufficiently subdivided by the uniform subdivision method, cutting along the edges of existing elements can highly approximate the actual cutting trajectory. However, such method is far from adequate when considering the operational efficiency as it is difficult to control the levels of subdivision in applications requiring precise incisions. Whereas it inspired us to compromise the speed and high mesh quality by exploring a non-uniform subdivision that can accurately estimate actual cut path but mitigate the discretisation errors due to the harmful triangles with large angles.

Stiffness matrix conditioning. Unlike in the discretisation errors, only small angles (but not large angles in elements in the absence of small ones) ruin the condition number of the stiffness matrix. The finite element equation for the simulated system is given by:

$$[K]\{u\} = \{F\} \tag{3.9}$$

where $\{u\}$ and $\{F\}$ represent the unknown nodal displacement vector and the known loaded forces for the system. As we discretise the problem domain with linear triangles, the global stiffness matrix F is a symmetric and square $n \times n$ matrix where n denotes the number of vertices of the model. The condition number of $k = \lambda^K_{max}/\lambda^K_{min}$, where λ^K_{max} and λ^K_{min} represent the largest and smallest eigenvalues of K respectively. As the global stiffness assembles each element stiffness matrix, k is roughly proportional to the largest eigenvalue in the element stiffness matrices. Freitag & Ollivier-Gooch (2000) have observed that for an element e in the system, if there are extremely small angles (approaching 0°), its largest eigenvalue λ^e_{max} can be extremely high, so that make the condition number of the global stiffness matrix high.

We employ the Conjugate Gradient Method (henceforth, CG), the most popular iterative solver to solve the system of simultaneous linear equations given by 3.9. This method converts the system into having to determine the minimisation of the associated quadratic form (Golub & Van Loan 2012). However, the ill-conditioned global stiffness matrix requires more steps to guarantee convergence which slows down the computation speed of the CG solver. In practice, a cut algorithm that refines the initial discrete elements should keep the condition number of the global stiffness matrix low by generating sub-elements without large angles.

3.2.2 Mesh Quality Insurance

Mesh measurement. According to the discussion in section 3.2.1, the mesh quality in FEM-based modelling is influenced by the element size and shape. However, it is not difficult to notice that a mesh measurement criterion may cause contradictory tensions between the interpolation errors, discretisation errors and stiffness matrix conditioning. Larger elements can cause more substantial interpolation errors. Both small angles approaching 0° and large angles approaching 180° slow down the process of solving the numerical solutions of the system when using iterative methods. As in Section 3.2.1, the uniform subdivision is time consuming in the

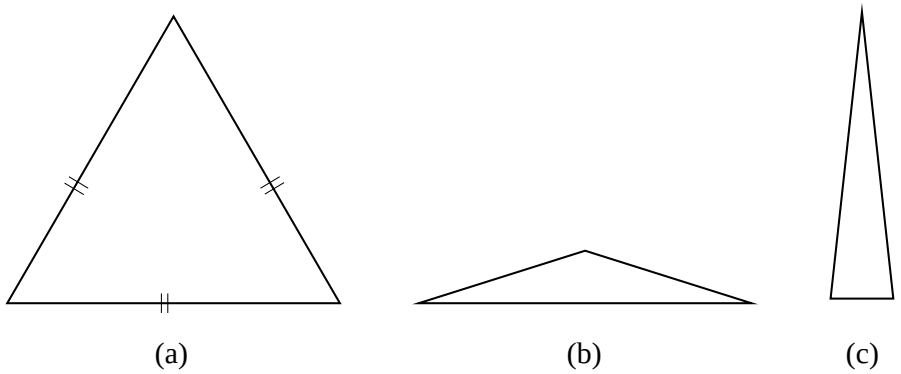


Fig. 3.7 The examples of the aspect ratio of triangles with different shapes. The aspect ratio for an equilateral triangle (a) is 1. The triangles in (b) and (c) that are identified in our experiment have higher aspect ratio.

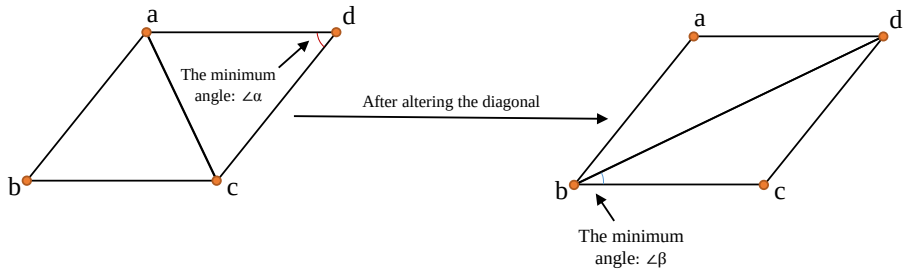


Fig. 3.8 Lawson's criterion. The nodes a, b, c, and d can form two pairs of adjacent triangles. If the minimum angle $\angle\alpha$ is larger than $\angle\beta$ the topology on the left hand side is more suitable for FEM-based modelling, otherwise is the topology on the right hand side that should be adopted.

cutting simulations that require an accurately representation of a cut; therefore this section focuses on the evaluation of the mesh quality regarding the shape of the elements in the modified mesh after applying cuts.

It can be proven that in a triangle (with three control points), the longest edge is always opposite the largest interior angle while the shortest edge is always opposite the smallest interior angle. Our goal is to avoid both large angles (approaching 180°) and small angles (approaching 0°). The *Aspect Ratio* of a triangle describes the ratio of the longest edge to the height from it. Figure 3.7 shows the impact of *Aspect Ratio* on the shape of triangles. The aspect ratio of a triangle is defined as the ratio of the length of the longest edge to the smallest edge. As shown in (b) and (c) a triangle which has an extremely large angle or small angles between the shortest side and the other two sides results in a higher aspect ratio that the ideal triangle (a). For our method it is both suitable and adequate to choose the aspect ratio as the determining criterion during our refinement scheme.

Conditional Delaunay Triangulation. Lawson (Lawson 1972) claimed a useful criterion for evaluating the FEM-based modelling tasks, saying that the minimum angle in any two adjacent triangles, which form a convex quadrilateral, should be larger than it would be if the diagonal altered and the other pair of triangles constructed. The Delaunay triangulation (Delaunay 1934) has been proven as the unique topology satisfying the criterion (Lawson 1977, Sibson 1978). It

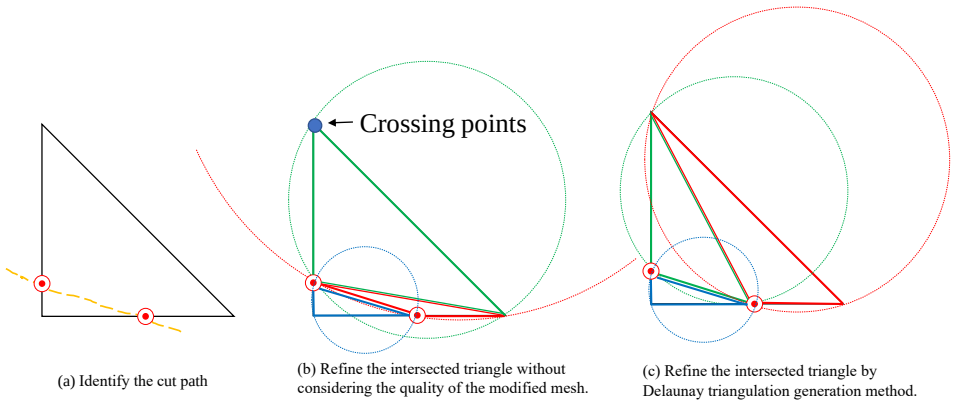


Fig. 3.9 Delaunay generation principle. The typical way is to refine each intersected triangle into three sub-triangles in a random connection (b) which can cause crossing points (the blue solid point) to occur in the circumcircle of the red triangle. A Delaunay triangulation satisfies the empty-circumcircle property which expresses that the circle formed by the vertices of each triangle does not enclose any other vertices in the triangulation (c).

is achieved by ensuring that no nodes are created within the circumcircle of existing triangles, also known as the empty-circumcircle property. The effect of this is to ensure that the minimum angle of any created triangles is maximised thereby improving the mesh quality.

Reproducing accurate virtual cuts always needs a non-uniform subdivision (Wu et al. 2015). However when using such a subdivision approach it is easy to generate ill-shaped triangles as those described in the two previous paragraphs. To maintain the quality of the modified mesh while making a cut we implement the Delaunay triangulation generation algorithm before separating the cut edge. Figure 3.9 illustrates how the Delaunay triangulation improves the mesh quality locally without introducing an extra level of detail. The traditional approach (b) randomly rearranges the node connectivity when introducing new nodes into the group (Christian et al. 2004). In this case, one triangle in the triangulation may contain nodes of other triangles (so-called cross point) within its circumcircle. In contrast, the Delaunay triangulation (c) adjacent the nodal connectivity ensuring no node is in the circumcircle of any triangle in the triangulation.

Adopting the Delaunay mesh generation method in mesh-based cutting simulations is challenging. After applying a cut the mesh is required to separate along the cut path which should not alter. However, in the Delaunay mesh generation method, it is assumed that each edge of the element can be changed. In this manner, if employing the Delaunay mesh generation method directly to refine the elements around the cut path, the edge which represents a part of the cut path may be changed. Therefore before implementing the Delaunay mesh generation method we give a condition which declares that the edge that is a part of the cut path cannot be altered during the refinement. We define the triangular mesh that is generated during such refinement as the *Conditional Delaunay Triangulation*. This condition will be further explained in section 3.3.1.

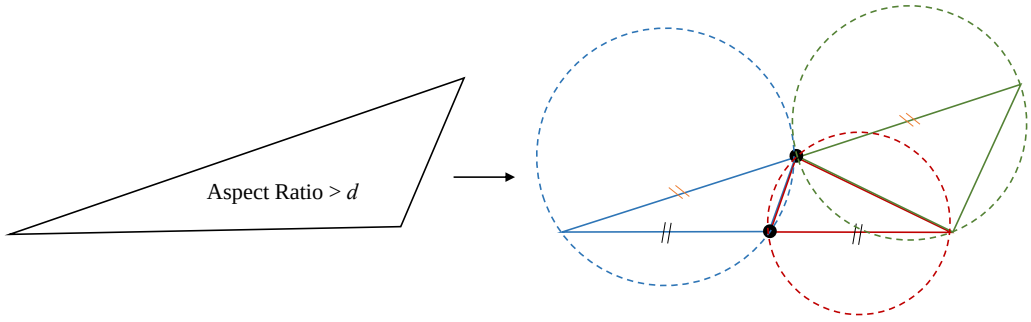


Fig. 3.10 One step in localised optimisation. For any triangle whose aspect ratio is higher than a tolerance value d a new vertex will be introduced at the middle of its two long edges respectively.

Bisection-based localized optimisation. Although we implement the Delaunay generation after the intersection detection stage, we fix the cut edge to respect the actual cut path so that ill-shaped triangular elements can generate. As discussed in section 3.2 the triangle aspect ratio is suitable to determine the mesh quality in our experiment. We choose this to evaluate the quality of individual triangles created during the optimisation operation. For a linear triangular (with three endpoints), the aspect ratio AR can be calculated as:

$$AR = l_{max}/l_{min}, \tag{3.10}$$

where l_{max} is the longest edge length, and l_{min} refers to the shortest edge length in the triangle.

Figure 3.10 shows an example of our localised optimisation operation. It takes place within each intersected triangular element which has been subdivided into a *Conditional Delaunay Triangulation*. In our experiment we constructed a regular-sized triangular mesh with an approximate aspect ratio of 2. According to our experimental pick the tolerant value $d <= 3$.

Algorithm description. Algorithm 1 illustrates the iterative optimisation process after each intersected triangular element is subdivided into a *Conditional Delaunay Triangulation* which may result in some ill-shaped triangular elements. The optimisation starts with cleaning the container v, which then is initialised with all bad triangles generated in the conditional Delaunay triangulation. We operate the bisection-based subdivision on each triangle tri in the current container. It is worth noting that a cutting algorithm should avoid unexpected cracks in the mesh after the cut. Such a mesh is also known as *conformal mesh*. Let τ_c be a triangular mesh of a problem domain Ω. It can be justified as a conformal mesh if the intersection of the interior of any two elements in τ_c is none of the empty set, a vertex, an edge or a face (Ho-Le 1988). See figure 3.11 for examples of an *non-conformal mesh* (left) and *conformal mesh* (right). Afterwards, each bad triangle and identified neighbours are reconstructed under the Delaunay triangulation criterion. Any new bad triangle generated in each optimisation time will be tracked and added to the container. The optimisation function will iterate until the container is empty, indicating the aspect ratio of any triangle in the mesh is less than 3.

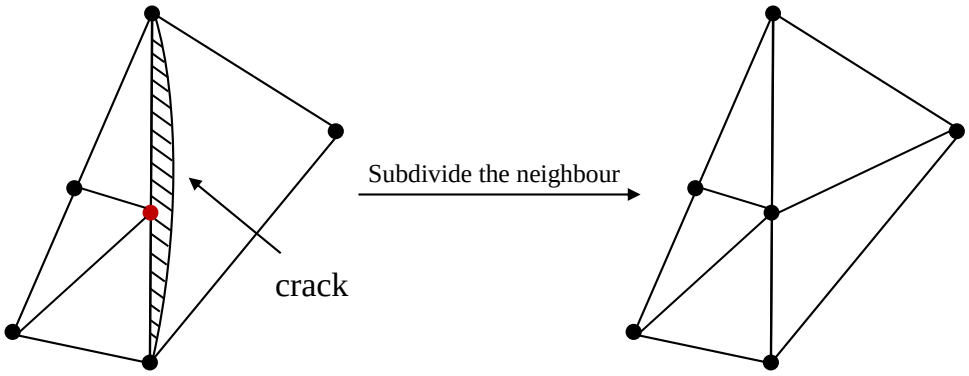


Fig. 3.11 Examples of a non-conformal and conformal triangular mesh. As shown in the left of the figure, only one of the triangles that share an edge is subdivided so that causes a hanging node (the red node). Such a mesh is a non-conformal mesh (left) which can cause a crack as shown. A conformal mesh (right) can be achieved by subdividing the both triangular elements that share an edge to avoid a crack.

3.3 Realistic Cuts with Affordable Computation Cost

To achieve a cut that is aesthetically pleasing while avoiding the volume reduction of the material model provides an opportunity for re-cutting in the same area. In such situation, a subdivision algorithm is required to be applied to the material model. Simply removing individual edges and vertices (spring-based models) or whole elements of the material (FEM) means that non-uniform holes will present themselves in the mesh topology. Re-cutting in this manner would eventually result in there being no mesh. On the other hand, a capable subdivision scheme in interactive or real-time simulations should be finished in a limited computational time. Therefore, a subdivision algorithm that can provide highly-detailed incisions but requires a reasonable level of detail increase is desired in real-time high-performance cutting simulations.

In this section, we address the proposed challenge above by presenting an approach that combines a Delaunay-based refinement scheme implemented by the Bowyer-Watson algorithm with an effective localised optimisation operation after opening the cut edge thus avoiding an unnecessary increase in the number of sub-triangles.

3.3.1 Conditional Non-Uniform Optimal Triangulation

Principles. A realistic incision simulation in interactive or real-time applications should meet the essential criteria. This means that it is capable of reproducing the arbitrary-shaped cut path while also providing detailed physical behaviours around the cut edge without any substantial increase in the computational cost. As discussed in section 3.2 the construction of larger triangular elements creates more substantial interpolation errors and discretisation errors. Therefore, we refine each intersected triangular element in turn into a non-uniform Delaunay triangulation while respecting the intersection point, preparing for separating the mesh along the

cut path. As described in section 3.1.2, each cut edge connected with introduced intersection nodes in intersected triangular elements estimate the actual cut path that is defined by users.

When considering the operational efficiency, we implement a Delaunay triangulation generation algorithm to reconstruct an optimal mesh without increasing a substantial number of elements. The standard Delaunay generation assumes that each edge in the triangulation is adjustable and can be changed. However if an edge which represent a part of the cut path is changed during this generation the shape of the cut path will change as well. To solve this problem, we fix all edges that are connected by the intersection nodes before refining the triangle. As a consequence, the initial refinement stage is capable of accurately reproducing arbitrary-shaped incisions and also increasing the physical accuracy of the cut edge by introducing the number of triangular elements around. Nevertheless, it may not avoid bad triangles that contain the fixed edges. To efficiently resolve this problem, we introduce a localised optimisation that will be described in detail in the next section.

Algorithm description. We employ an efficient implementation of the Bowyer-Watson algorithm (Bowyer 1981, Watson 1981) to generate the planar Delaunay triangulation within each intersected triangle. Many implementations aim to satisfy the empty-circumcircle property in a triangulation. In existing research, such as Nienhuy and his colleagues' work (Nienhuys & van der Stappen 2004), the conventional way is to optimise the refined mesh after a cut with the Flip algorithm. It achieves this by identifying any non-Delaunay triangle, combining it with a neighbouring triangle and then "flipping" the edge that divides the resulting quadrilateral into two triangles. The technique continues to iterate through the triangle list until no non-Delaunay triangles are found. Consequently, the computational demand can be high and an endpoint to the process is not guaranteed.

Conversely, the Bowyer-Watson algorithm introduces an incremental approach to generate Delaunay triangulation for discrete points in any dimension. It inserts one node at a time, and then locally validates the triangulation of the subset of the desired points. For a set of points N, the time T to generate a Delaunay triangulation is:

$$T = \sum_{k-1}^{N} T_k + S_k \tag{3.11}$$

where T_k is the time to find the triangle within which the new point lies, and S_k the time to determine all boundary points in the cavity. S_k has a complexity of $O(l)$ that is proportional to the number of elements in the cavity and independent of the number of points. Therefore, T_k is the dominant factor in the computational cost of this algorithm, which can take $O(N \times \log N)$ to triangulate N points. To avoid repeated searching all existing triangle for intersection when inserting each point, Sloan & Houlsby (1984) proposed to sort the discrete points in ascending sequence according to one coordinate, say the x-coordinate before the operation. Therefore if the distance from the x-coordinate of the first point in the list to the testing triangle circumcentre is no less than to the radius of the triangle circumcircle, then this search stops here. Also, such

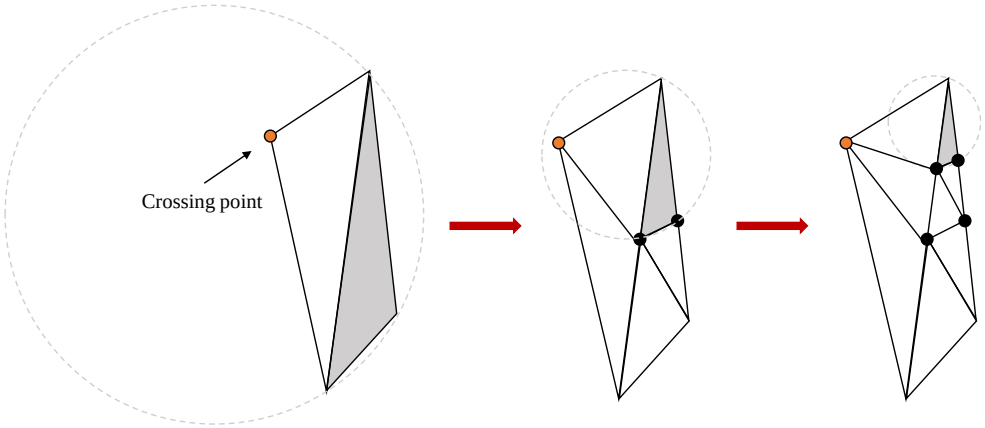


Fig. 3.12 An example of the recursive local bisection-based optimisation. The impact of the bad triangle (depicted in grey) is mitigated by the recursive process (two times here) to satisfy the empty-circle principle.

triangle is then a Delaunay triangle which can skip the following detections and modifications thereby reducing the computation time T_k.

3.3.2 Efficient but Adequate Localised Optimisation

Principles. The job of our further localised optimisation is to handle the ill-shaped triangular elements (also referred to as non-Delaunay triangles) generated in the previous conditional refinement due to the fixed edges which represent the cut path. It is theoretically capable of refining the non-optimal mesh further using the Delaunay algorithm. As mentioned above, the computation time can be up to $O(N \times \log N)$ to re-triangulate a mesh containing N points. However, compared with the Delaunay refinement algorithm, the bisection refinement algorithm has only a linear cost time complexity which is more suitable for practical real-time applications. Tanaka et al. (2006) has shown the computational efficiency of bisection refinement algorithm using experimental results.

We present a recursive local bisection refinement algorithm combined with the empty-circle principle to efficiently optimise the bad triangles whose aspect ratio is higher than 3. Figure 3.12 illustrates the recursive process of our local bisection-based optimisation. A bad triangle whose aspect ratio is higher than three is identified before the optimisation algorithm (left). The two longest edges of the bad triangles are then divided into two parts, and new nodes are introduced, preparing for the following refinement. If there are still triangles that violate the empty-circle principle as shown in the middle of figure 3.12, the optimisation operation continues until all generated triangular elements meet the measurement criterion (right).

Multi-resolution but conformal mesh. Most existing real-time applications construct the initial mesh in a relatively coarse level with uniform resolution. However in cutting simulations the region near to the cut incision has to be in a much higher resolution. Conversely other regions should avoid this increase in resolution to ensure an affordable computational overhead. This prompts us to employ a multi-resolution mesh, which allows a higher computational cost due to

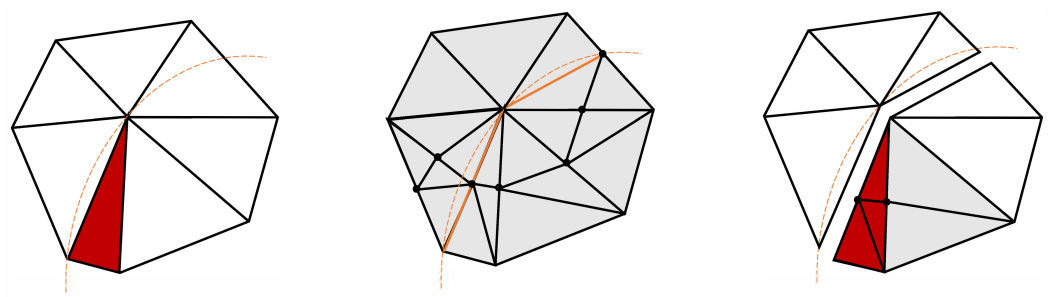


Fig. 3.13 The *Adapted Constrained Delaunay Method (ACDM)*. The red triangle is firstly identified to be subdivided. The global *Constrained Delaunay Method* (middle) remains the constraint edges (the yellow lines) and refines the whole mesh. The proposed *ACDM* approach refines the red triangle after the edges along the cut are separated.

the subdivision focus in the cut region which is known as an adaptive subdivision (Manteaux et al. 2017). Recall the discussion in section 3.2.2, in FEM-based models any topological modification should ensure the conformal property of the mesh otherwise, it can cause undesired holes or cracks.

Different level of subdivision between adjacent triangles results in a non-conformal mesh. To solve this problem the neighbours of each bad triangle also need to subdivide as described in Algorithm 1. These neighbouring triangular elements increase the computational cost in subdivision operation. Utilising the feature of the introduced localised optimisation process, which is valid in both convex and non-convex topologies, we operate it after generating the conditional Delaunay triangulation and separating the cut edges.

As shown to the left of figure 3.13, the red triangle has been identified as an ill-shaped triangle in the conditional Delaunay triangulation. In the middle of the figure, the global *Constrained Delaunay Method* guarantees certain edges which are referred to as *constraints* (Sloan 1993). Such a method typically refines the whole mesh. (Busaryev et al. 2013) employs a global *Constrained Delaunay Triangulation* method in the simulation of fractures on thin shells. The computational complexity of this method is usually higher than $O(N)$. In the worst case, it can reach $O(N^2)$. N indicates the number of points in the triangulation (Lee & Lin 1986, Sloan 1987, Nam et al. 2009).

Adapted Constrained Delaunay Method (ACDM). We propose a new subdivision scheme which only refines the elements around the cut and avoids the ill-shaped elements. The working process of ACDM is provided in Algorithm 1. In this section, we will discuss the computational complexity of ACDM and introduce the data structure used for supporting fast searching and updating phases.

A.Computational Complexity

$Optimisation(t)$ method starts by collecting all the indices of ill-shaped elements in container v. ACDM is implemented in both $BisectionRefinement(\text{tri,neighbours})$ and $BisectionRefinement(\text{tri})$ functions. $BisectionRefinement(\text{tri})$ refines an ill-shaped element and $BisectionRefinement(\text{tri,neighbours})$ also refines all its neighbouring element(s). ACDM

adopts the *Bisection-Based* approach which holds a linear computation complexity (Tanaka et al. 2006), i.e. $O(N)$, where N is the number of points involved in the refinement. In our algorithm, $N = 4$ or $N = 5$ when refining a single ill-shaped element. Lines 13 to 15 detect whether there is any new ill-shape element generated and store the indices into container v.

The $Optimisation(t)$ function runs until there are no ill-shaped elements left in the mesh. Assuming there are M ill-shaped elements initially in the mesh, the computation complexity of the $Optimisation(t)$ function is $O(K(NlogM))$ on average, where K is the repeating time of the *while* loop from lines 3 to 18 in Algorithm 1.

Algorithm 1: Optimasation(t)

```
   Input: container v
 1 clean the container v;
 2 initialise v with t;
 3 while v is not empty do
 4     foreach triangle tri in v do
 5         find the longest edge;
 6         insert a new vertex at the middle of the edge;
 7         searchNeighbors(tri);
 8         if find any neighbor then
 9             BisectionRefinement(tri,neighbours) //O(N) with N vertices
10         else
11             BisectionRefinement(tri) //O(N) with N vertices
12         end
13         if have ill-shaped triangles created then
14             put them into v;
15         end
16     end
17     erase tri from v;
18 end
```

B. Data Structure

Four structures and a class are initialised when constructing each vertex and triangular element. The **Vertex** list stores the index and position of a vertex in the mesh. The **Edge** list collects the indices of the vertices for each edge as well as the elements which share this edge. In the **Triangle** class, *tri_idx* represents the index of an triangular element. The **Vertices** and **Edges** collects the indices of vertices and edges for the triangular element, respectively. Figure 3.14 illustrates the associations between the structures and the class. The position of each vertex in the mesh can be obtained by matching the index collected in the **Vertices** structure. All structures and the class are linked by references to save the storage and allow fast searching and updating phases.

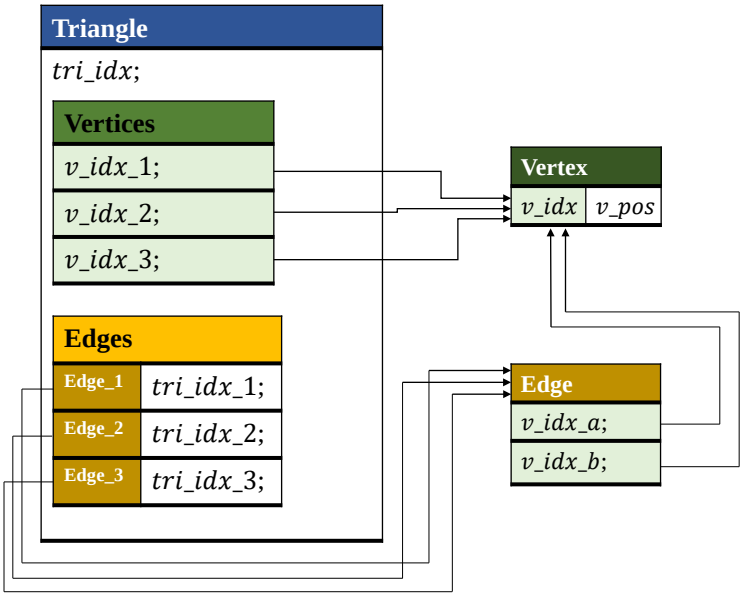


Fig. 3.14 The **Triangle** class and its association with other structures.

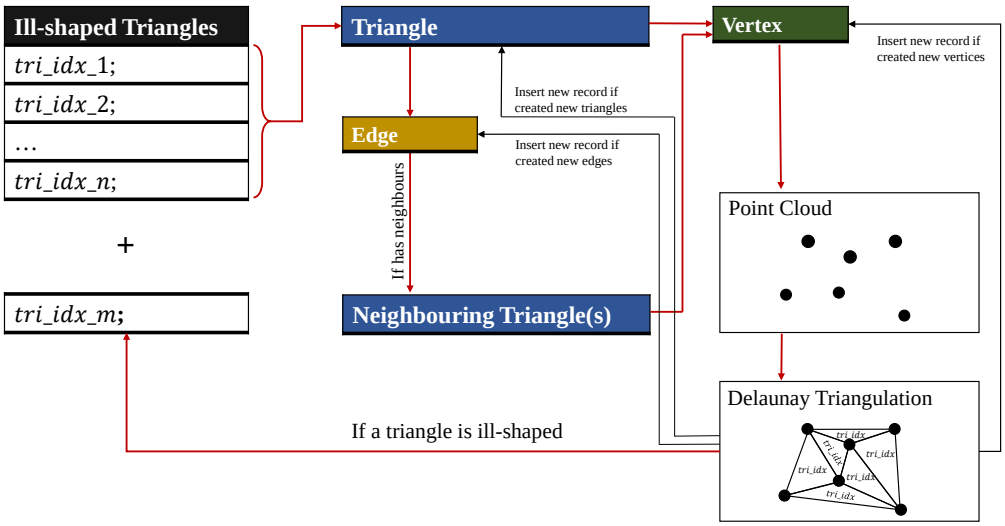


Fig. 3.15 The data searching and updating phases in the *BisectionRefinement*(tri) and *BisectionRefinement*(tri, neighbours) functions.

Figure 3.15 shows the data searching and updating phases after applying a cut to the simulated material. First, the **Ill-shaped Triangles** list is initialised with the indices of all ill-shaped elements detected. Each index is used to find the record in the **Triangle** class by matching the index as a key. Afterwards, the information (i.e., indices and positions) of all vertices involved in the following refinement will be put in the **Point Cloud** list. The information of any new vertices created will also be added in both **Vertex** and **Point Cloud**. Finally, an ill-shaped triangular element will be refined by using the vertices stored in the **Point Cloud** list. After the refinement, all structures, lists, and classes will update accordingly. Moreover, if any of the new element is recognised as ill-shaped, its element index will be added to the **Ill-shaped Triangles** list.

3.4 Fast and Robust Matrix Updating

The time efficiency is of crucial importance when evaluating the applicability of a cutting method. Both model initialisation and the modification due to the cuts influence the performance resulting from the computational complexity. In this section, we will provide effective solutions for both the model initialisation (Section 3.4.1) and the topological modification (Section 3.4.2) after cuts are applied.

3.4.1 Model Initialisation.

The principle. In FEM-based simulations, the simplest and fastest way is to employ linear elasticity as the governing rule. However this compromises the deformation accuracy by ignoring the effect of rotations. In the real world pure rotations on an object should not change its shape. Modelling using linear deformations can cause unrealistic or unstable effects when non-linear deformations are present. To provide a physically accurate model in real-time applications we employ the co-rotational FEM proposed by (Felippa 2000). This method translates the material models in the deformed configuration back to a rotational un-deformed configuration, and then computes the linear deformations without the rotation part to efficiently handle large deformations. Such an approach allows us to model realistic physical behaviours while keeping the computational time low.

We aim to represent the dynamic deformation of materials. In FEM-based modelling, the motion of a mass-node in the material model and its dynamics behaviour at time t can be described as:

$$\mathbf{M}\ddot{\boldsymbol{x}} + \mathbf{D}\dot{\boldsymbol{x}} + \mathbf{K}(\boldsymbol{x} - \boldsymbol{X}) = F, \tag{3.12}$$

where F is the applied external forces to the material model and it is represented in a matrix form here. $\mathbf{M}$ is the diagonal mass matrix in which the mass of each node in the material model is placed at the diagonal of the matrix while the rest of the places are zero. $\mathbf{D}$ is the global damping matrix and $\mathbf{K}$ is the global stiffness matrix of the material model. $\ddot{\boldsymbol{x}}$, $\dot{\boldsymbol{x}}$ represent the second (acceleration) and first (velocity) derivatives of $\boldsymbol{x}$ (the node position in the current

configuration) with respect to t while $\boldsymbol{X}$ is the node position in a reference configuration. In order to simulate a model's dynamic behaviours, a time integration method is required to calculate and update each vertex's position for the next time step. In this case, both $\boldsymbol{x}$ and $\boldsymbol{X}$ are time-dependent variables. $\boldsymbol{x}$ represents the position in the current time step, while $\boldsymbol{X}$ holds the position in the previous time step. Recall the introduction in section 2.6.1, both explicit and implicit time integration method can be employed. As discussed in this section, the explicit time integration method is only stable when the time step is sufficiently small so large time steps are not possible. Therefore we employ an implicit time integration method (Baraff & Witkin 1998) which allows a larger time step so that achieve a real-time simulation.

Conjugate Gradient. Obtaining $\boldsymbol{x}$ in Equation 3.12 requires the solution of a system of equations in the form of

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.13}$$

where $\mathbf{A}$ is a symmetric square matrix that is positive definite. Such a system can be solved by direct methods which calculate the inverse of the matrix $\mathbf{A}$, i.e. $\mathbf{A}^{-1}$. Multiplying both sides of Equation 3.13 by $\mathbf{A}^{-1}$ obtains

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}. \tag{3.14}$$

However, the inversion of a matrix can be expensive when the size of the matrix is large (Wu et al. 2015). Alternatively, we employ the Conjugate Gradient method (CG) which obtains a good approximation to the solution for $\mathbf{x}$ in an iterative way. In each iteration, a guess for $\mathbf{x}$ is determined if it is closer to the real solution. In practice, only a few iterations are required to obtain a good approximation for the solution.

In practice. We discretise the problem domain with n vertices represented in the matrix form $\mathbf{M}$, which is a $n \times n$ diagonal matrix. Therefore, the time-dependent dynamic motion equation 2.21 can be rewritten as

$$\mathbf{M}\ddot{\mathbf{x}}^{t+\Delta t} = \mathbf{f}(\mathbf{x}^{t+\Delta t}, \dot{\mathbf{x}}^{t+\Delta t}). \tag{3.15}$$

According to the first-order Taylor approximation of f,

$$\mathbf{f}(\mathbf{x}^{t+\Delta t}, \dot{\mathbf{x}}^{t+\Delta t}) \approx \mathbf{f}(\mathbf{x}^{t}, \dot{\mathbf{x}}^{t}) + \frac{\partial \mathbf{f}(\mathbf{x}^{t}, \dot{\mathbf{x}}^{t})}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial \mathbf{f}(\mathbf{x}^{t}, \dot{\mathbf{x}}^{t})}{\partial \dot{\mathbf{x}}} d\dot{\mathbf{x}}, \tag{3.16}$$

in which

$$\begin{cases} d\mathbf{x} = \mathbf{x}^{t+\Delta t} - \mathbf{x}^{t} = \Delta t \dot{\mathbf{x}}^{t+\Delta t} \\ d\dot{\mathbf{x}} = \dot{\mathbf{x}}^{t+\Delta t} - \dot{\mathbf{x}}^{t} = \Delta t \ddot{\mathbf{x}}^{t+\Delta t}, \end{cases} \tag{3.17}$$

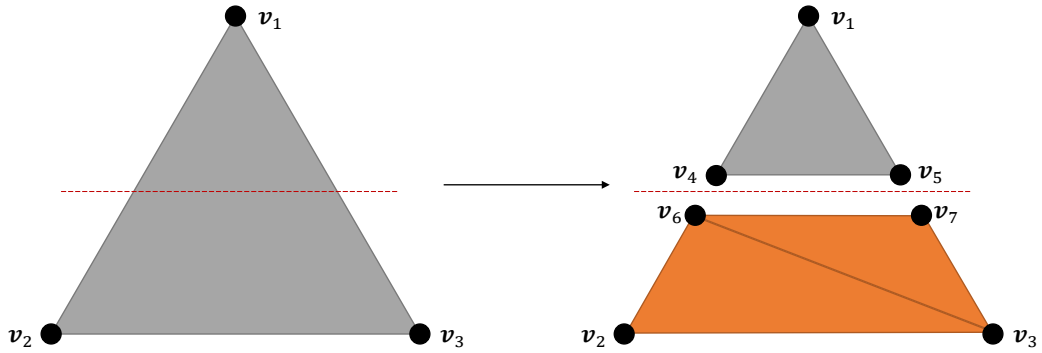


Fig. 3.16 A example of the index modification for an original intersected triangle.

and

$$\mathbf{f}(\mathbf{x}^{t+\Delta t}, \dot{\mathbf{x}}^{t+\Delta t}) = \mathbf{f}^{int}(\mathbf{x}^{t+\Delta t}, \dot{\mathbf{x}}^{t+\Delta t}) + \mathbf{f}^{ext}, \tag{3.18}$$

where $\frac{\partial \mathbf{f}(\mathbf{x}^t, \dot{\mathbf{x}}^t)}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}(\mathbf{x}^t, \dot{\mathbf{x}}^t)}{\partial \dot{\mathbf{x}}}$ refer to the so-called the stiffness matrix $\mathbf{K}$ and damping matrix $\mathbf{D}$, describing the changes of nodal forces with respect to the position and velocity variables respectively. $\mathbf{f}^{ext}$ is the external force caused by the self-collision of the simulated material.

Afterwards, the dynamic motion equation 3.16 is transformed into a linear system that will be solved by the Conjugate Gradient method at each time step during the simulation

$$\mathbf{A}\ddot{\mathbf{x}}^{t+\Delta t} = \mathbf{b}, \tag{3.19}$$

in which $\mathbf{A} \equiv (\mathbf{M} - \Delta t\mathbf{D} - \Delta t^2\mathbf{K})$, and $b \equiv \mathbf{f}^{int}(\mathbf{x}^{t+\Delta t}, \dot{\mathbf{x}}^{t+\Delta t}) + \mathbf{f}^{ext} + \mathbf{K}\Delta t\dot{\mathbf{x}}^t$. As mentioned in section 3.2, the mesh quality influences the time of the CG iterative solver. Our mesh quality insurance avoid ill-conditioned matrices so this makes the employment of the CG solver adequate for satisfying the real-time requirement.

3.4.2 In Model Modification

Index replacement. In the proposed cutting method, each intersected triangle will come under the refinement into a local triangulation. It disconnects the relationship between the intersected triangle and those triangles that share the nodes or edges. In the global mass, damping and stiffness matrix of the system the contribution of the original intersected triangles needs to be removed before starting the next time step. Such changes invalidate the previous construction of these global matrices which then need to be re-constructed in preparation for the next cut operation. However it can be time-consuming to construct the whole matrices when the size is large. Contrary to most methods we replace the contribution of the original intersected element by modifying the nodal indices of all intersecting faces in the rendering buffers where the information for searching and indexing is stored.

For instance, figure 3.16 gives an example of the index modification process for an original intersected triangle whose element stiffness matrix is $\mathbf{S}$ constructed with its nodes v_1, v_2, and v_3. When the triangular element is intersected with the cut path two of its nodal indices are replaced

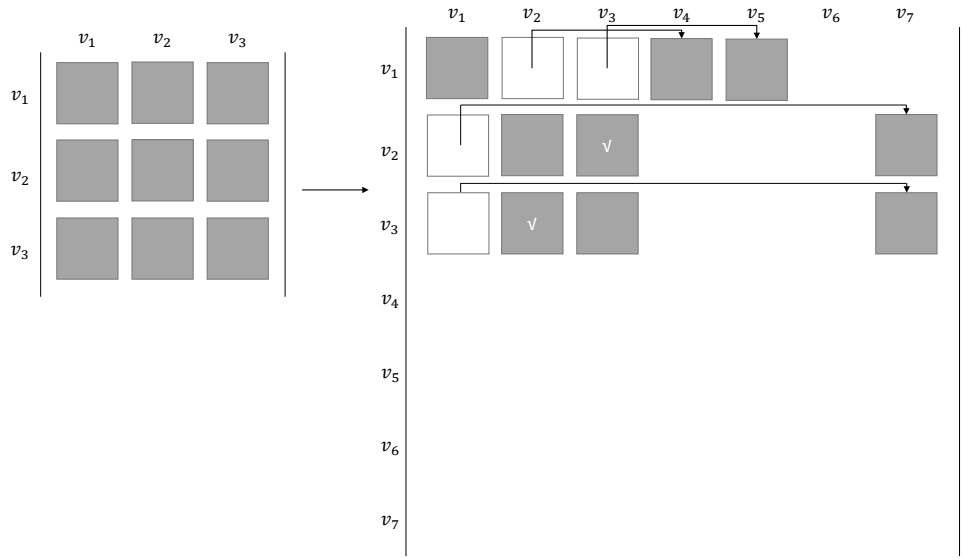


Fig. 3.17 Index modification of the global stiffness matrix after the cuts. After the triangular element $v_1 - v_2 - v_3$ has been cut, we transport them to the place where they can be reused.

rather than totally removing the three nodal indicates from the rendering buffer. Afterwards, the nodal indices of new triangular elements due to the following refinement are added at the end of the rendering buffer. This method can partially simplify the process of updating due to the cut. As shown in the left of the figure, the triangle has two intersection edges (v_1v_2, v_1v_3). Furthermore we replace the affected indices in the rendering buffer by the new nodal indices on the corresponding intersected edge (from v_2 to v_4, from v_3 to v_5) but keep the nodal index v_1, to fill in the topology on one side of the cut. This process can also be deposited in a matrix form (see figure 3.17). In the next step the element stiffness matrix needs to update from **S** into the form as shown on the right. Figure 3.18 shows the updated matrix after integrating new contributions of generated sub-elements. We mitigate the computational inefficiency due to the discrepancies in the nodal indices in this index modification step.

Fast global matrices updates. The refinement changes the initial nodal connections and introduces new nodes to represent the mesh discontinuity. Both modifications require the reconstruction of the global matrices. When considering the computational efficiency, we resolve the modifications on the original intersected element in the index replacement step. Therefore, this stage purely handles the addition of the global matrices, and therefore accelerates the process of updating. Instead of reconstructing the whole global matrices we incrementally update the global matrix

$$\mathbf{K}^{'} = \mathbf{K}_0 + \sum_{i=1}^{m} \mathbf{G}_i \mathbf{K}_i \mathbf{G}_i^T \tag{3.20}$$

where the current state of the global stiffness matrix $\mathbf{K}_0$ is added with the global matrix $\mathbf{K}'$ before the next time cutting is updated by adding the contributions of new generated elements with new m nodes. The spares matrices $\mathbf{K}_i$ and $\mathbf{G}_i$ map the rows and the columns of the new discrete element matrices to the global matrix. In our experiment the mass and damping

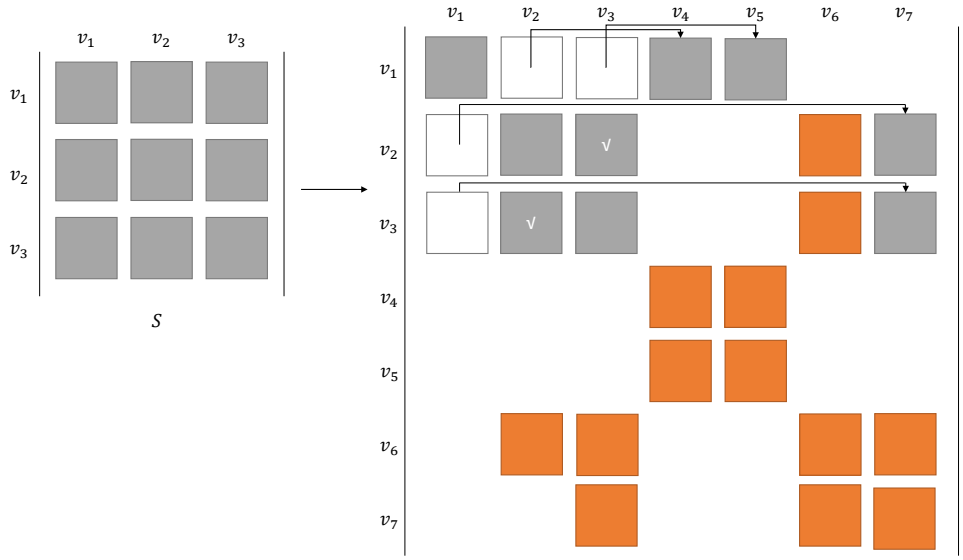


Fig. 3.18 The new global stiffness matrix for the model mesh shown in figure 3.16. New entities (shown in orange) are generated to integrate new triangular elements into the mesh.

matrices are updated in a similar way. In this manner the contributions of new elements are incrementally added into the global matrix providing a dynamic incorporation process between each new node and the index list. This then avoids the substantial computational cost due to recalculating the whole global matrices for the entire simulated model when the material model meets the topological modifications due to cuts.

3.5 Matrix Optimisation

Finite Element Method (FEM) aims to solve a set of partial differential equations (PDEs) by finding the corresponding numerical solutions. We thereby introduce the *numerical* solutions for eliminating the impacts of ill-shaped elements.To be more concrete, adopting FEM implies solving a certain number of matrix equations in the form of $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is the global stiffness matrix, $\mathbf{b}$ is a column vector. Given $\mathbf{A}$ and $\mathbf{b}$, our target is to obtain the value of $\mathbf{x}$. Specifically, in the simulation of physically-based FEM models, $\mathbf{A}$ stores the global coefficients, such as stiffness, mass, and damping; $\mathbf{b}$ assembles all external forces applied to the model. $\mathbf{x}$ expresses the displacements of each vertex in the model. Therefore, the efficiency of simulating a physically-based FEM model depends on the computational time spent on solving the equations system. For mesh-based methods, achieving a complex pre-defined cut remains challenging due to the inflexibility of the matrix inversity computation. Specifically:

- directly solving the equations system offers an accurate solution for physics, while it neglects the fact that matrix inversity computation is expensive when the dimension of the matrix is large. Namely, it suffers from the problem of "curse of dimensionality" since the cuts cause the generation of new vertices and new elements. The computation is significantly slowed down when the magnitude of matrix $\mathbf{A}$ increases in an exponential

way. Therefore, a direct method for solving the equations system is not capable of maintaining the efficiency of the cutting simulation.

- instead of computing the matrix inverse, iterative methods, such as Conjugate Gradient (CG), obtain an approximate solution of the equation system in an iterative way. However, the convergence speed of such methods is decided by the condition number $k(\mathbf{A})$ of the matrix $\mathbf{A}$: a large $k(\mathbf{A})$ slows down the speed.

3.5.1 ES-CGM

Recalling the discussion in Section 3.2.1, the ill-shaped elements will substantially stretch the condition number of the global stiffness matrix, i.e., $\lambda_{max}/\lambda_{min}$. Such a matrix is also known as an ill-conditioned matrix. When employing the Conjugate Gradient iterative solver to approximate the solution of an equation system such as $Ax = b$, the convergence speed will slow down if the matrix A is ill-conditioned or even worse, influence the numerical stability of the solver. In the mesh-based cutting method, the new triangular elements are arbitrarily-shaped which can easily cause an ill-conditioned global stiffness matrix. In this section, we will continuously investigate how to eliminate the harms due to the ill-conditioned global stiffness matrix.

Our strategy is to prevent the global stiffness matrix from being ill-conditioned by avoiding the condition number being over-stretched. To accomplish this target, we propose the Exact Square Root of Covariance Global Stiffness Matrix (shortened as ES-CGM), which shrinks all eigenvalues equivalently by the operation of the square root, thereby reducing the ratio of the largest eigenvalue to the smallest eigenvalue. After applying the cuts to the simulated surface model, the global stiffness matrix constructed previously is invalidated due to the changes of node connection and its dimension. Moreover, those changes can make the global stiffness matrix ill-conditioned. In order to apply Cholesky factorisation, we transform the global stiffness matrix after the cuts into a covariance matrix which can be obtained by:

$$COV = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \overline{\mathbf{x}})^T(\mathbf{x}_i - \overline{\mathbf{x}}), \tag{3.21}$$

where $\mathbf{x}_i$ is the n^{th} row vector of the matrix $\mathbf{A}$ and $\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ is the mean of $\mathbf{x}_i$.

Following Equation (3.21) the global stiffness matrix $\mathbf{A}$ can be transformed into a Symmetric Positive Definite (SPD) matrix $\mathbf{M}$ which can be used for the Choleskey factorisation.

Recalling the Cholesky factorisation, $\mathbf{M}$ can be decomposed as:

$$\mathbf{M} \mapsto (\mathbf{L}, \mathbf{D}), \mathbf{M} = \mathbf{L}\mathbf{D}\mathbf{L}^T, \tag{3.22}$$

where $\mathbf{L}$ is a lower unit triangular matrix and $\mathbf{D} = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal of eigenvalues λ_i, where $i = 1, \ldots, n$. However, in practice, matrix $\mathbf{D}$ often suffers from numerical instability. Therefore, $\mathbf{D}$ is regularised by a regularisation term $1e^{-6}I$ so that it can be replaced by $\mathbf{D}^{'} = \mathbf{D} + \varepsilon\mathbf{I}$, where $\varepsilon = 1^{-6}$ and $\mathbf{I}$ is the identity matrix with the same dimension as $\mathbf{D}$. After this, we adopt p-norm which allows us to transform the power of the matrix to the eigenvalues by $\mathbf{L}\mathbf{D}^{'}\mathbf{L}^T$ decomposition. This mapping can be written as:

$$(\mathbf{L}, \mathbf{D}) \mapsto \mathbf{M}', \mathbf{M}' \triangleq P^p = \mathbf{L}F(\mathbf{D}')\mathbf{L}^T, \tag{3.23}$$

where $F(\mathbf{D}')$ is a power function of $\mathbf{D}$. Considering the truth of that the elements of a SPD matrix naturally endow on the Riemannian manifold, it is appropriate to perform logarithm to the eigenvalue matrix which can be denoted as:

$$\begin{aligned} F(\mathbf{D}') &= \text{diag}(f(\lambda_1), f(\lambda_2), \ldots, f(\lambda_n)) \\ &= \text{diag}(log(\lambda_1), log(\lambda_2), \ldots, log(\lambda_n)). \end{aligned} \tag{3.24}$$

where $f(\lambda_n)$ is the power of eigenvalues. The logarithm of eigenvalues allows us to measure the tangent distance of Reimanian manifold in Euclidean space. However, the logarithm of eigenvalues can significantly change the magnitudes of eigenvalues, especially, overstretch the small eigenvalues and reverse its order. For example, for a large eigenvalue $\lambda = 30 \mapsto log(\lambda) \approx 1.5$ but for a small eigenvalue $\lambda = 10^{-3} \mapsto log(\lambda) = -3$. As an alternative, we explore an efficient way to normalise eigenvalues while avoiding the negative ones by defining $f(\lambda_n) = \lambda_n^p$, where p is a positive real number ($p - norm$, henceforth). Similar to Equation (3.24), it can be written as:

$$F_p(\mathbf{D}') = \text{diag}(\lambda_1^p, \lambda_2^p \ldots, \lambda_n^p). \tag{3.25}$$

The p and the logarithm normalisation are closely connected. The relationship can be expressed in the following proposition:

Proposition 3.5.1. *Let* $\mathbf{H}$ *and* $\mathbf{H}'$ *be two matrices, then the limit of the power of eigenvalue metric* $d_p(\mathbf{H}, \mathbf{H}') = \frac{1}{p}||\mathbf{H}^p - (\mathbf{H}')^p||_F$ *as* $p > 0$ *equals the Logarithm of eigenvalue metric, implies that* $\lim_{p\to 0} d_p(\mathbf{H}, \mathbf{H}') = ||log(\mathbf{H}) - log(\mathbf{H}')||_F$*, where* $||\cdot||_F$ *denotes the Frobenius norm of matrix.*

Proof. Recalling that $d_p(\mathbf{H}, \mathbf{H}') = \frac{1}{p}||\mathbf{H}^p - (\mathbf{H}')^p||_F$. Based on the Choleskey decomposition of $\mathbf{A}$ we have $\frac{1}{p}(\mathbf{A}^p - I) = L\text{diag}(\frac{\lambda_1^p - 1}{p}, ..., \frac{\lambda_n^p - 1}{p})\mathbf{L}^T$. The identity about the limit in **Proposition** 3.5.1 follows immediately by recalling $lim_{p\mapsto 0}\frac{\lambda^p - 1}{p} = log(\lambda)$. □

It is not hard to find that the logarithm normalisation measures the true geodesic distance while the p-norm is the approximation of the measure. However, we argue that the p-norm is a more stable option in our experimental scenario. Specifically, after applying cuts to the simulated surface model, the generated triangular elements with small areas mainly contribute the small eigenvalues to the global stiffness matrix. As the sequence, the negative normalised eigenvalues can easily be generated. According to our experimental pick, $p = 1/2$ is employed in our algorithm.

Once the p-norm, $p = 1/2$ has been finished on $\mathbf{D}'$, the resulting $\mathbf{D}''$ can be directly used for the ongoing reconstruction $(\mathbf{L}, \mathbf{D}'', \mathbf{L}^T) \mapsto \mathbf{A}'$ and the optimisation process.

3.5.2 AS-CGM

The ES-CGM method proposed in Section 3.5.1 obtains the exact square root of a SPD matrix by utilising the Cholesky decomposition. Although such an approach is able to gain the accurate solution of matrix square root, the matrix decomposition can be time-consuming when the dimension of the matrix gets large. Alternatively, we adopt the Newton-Schulz iteration method (Higham 2008) to effectively compute an approximate square root for the covariance global stiffness matrix (shortened as AS-CGM) without needing matrix decomposition.

Let $\mathbf{P}$ be the square root of matrix $\mathbf{A}$ with $\mathbf{P}_0 = \mathbf{A}\ \mathbf{Q} = \mathbf{I}$. For $k = 1, ..., n$, each iteration step computes in the form:

$$\begin{aligned} \mathbf{P}_k =& \mathbf{P}_{k-1} p_{lm}(\mathbf{Q}_{k-1}\mathbf{P}_{k-1}) q_{lm}(\mathbf{Q}_{k-1}\mathbf{P}_{k-1})^{-1} \\ \mathbf{Q}_k =& p_{lm}(\mathbf{Q}_{k-1}\mathbf{P}_{k-1}) q_{lm}(\mathbf{Q}_{k-1}\mathbf{P}_{k-1})^{-1} \mathbf{Q}_{k-1}, \end{aligned} \tag{3.26}$$

where p_{lm} and q_{lm} are polynomials with l, m representing non-negative integers. Such an iteration converges locally only if $\|\mathbf{A} - \mathbf{I}\| < 1$, $\|\cdot\|$ which represents any induced (or consistent) matrix norm. By doing so, $\mathbf{P}_k$ and $\mathbf{Q}_k$ converge to $\mathbf{Y}^{\frac{1}{2}}$ and $\mathbf{Y}^{-\frac{1}{2}}$ respectively. This coupling of iterations is called *Newton-Schulz Iteration* when $l = 0, m = 1$, i.e.

$$\begin{aligned} \mathbf{P}_k =& \frac{1}{2}\mathbf{P}_{k-1}(3\mathbf{I} - \mathbf{Q}_{k-1}\mathbf{P}_{k-1}) \\ \mathbf{Q}_k =& \frac{1}{2}(3\mathbf{I} - \mathbf{Q}_{k-1}\mathbf{P}_{k-1})\mathbf{Q}_{k-1}. \end{aligned} \tag{3.27}$$

Compared with the ES-CGM method, this method computes an approximate square root of the global stiffness matrix which may lead to lower accuracy. However, such an iterative approach is more effective as only matrix product is required. Therefore, it allows our cutting framework to effectively solve the equations system.

3.6 Complex Cutting in 3D Environment

Practical cutting applications, such as a virtual surgery simulator and damaging scene in video games always require the interaction between users and the simulated model. The simulation always allow users to define the shape of the cut incision. However, the more flexible and complex cuts that are allowed the more challenges a cutting method will meet. In such situations, the simulation has to ensure high performance in the cutting models that allows complex cut operations. Moreover it can become more difficult to handle in the 3D environment in which the users can touch the cutting model anywhere with the simulated blade. The cutting method should also have the ability to allow multiple cuts while keeping the simulated system stable. A poor cutting strategy can cause unexpected holes or connections in complex situations such as cutting on the curved surface model.

In this section, we will present a strategy to handle complicated cuts in a 3D environment. First we will introduce an efficient approach to represent a smooth arbitrarily-shaped cut whose shape is pre-defined by the user. Furthermore, we will describe our solutions for handling more complicated cut situations, including partial cuts, re-cut or cross cuts at the same area. Finally we propose how our cutting models can provide high-performance sub-meshes, allowing repeated cuts while maintaining the accuracy for collision detection and dynamic physical behaviours.

3.6.1 Smooth Arbitrary-Shaped Cuts

The difficulties. Opening an incision is difficult when the shape of the cut path is not straight. After refining the intersection triangles the intersection mass-nodes are duplicated in preparation for the incision opening. The sub-triangles generated need to be rearranged on both sides of the incision. Compared with straightforward intersection detection it is difficult to determine on which side of the cut plane should a sub-triangle be placed. In a progressive cutting process the material model is opened when the simulated blade cut through the mesh. In this manner, each group of the refined sub-triangles is open in turn. However in a non-progressive cutting process the cut occurs after the users give the arbitrarily-shaped cut path so that it is not easy to separate the sub-triangles with the whole cut path. As depicted in figure 3.19, the blue triangle is on the positive side of the piecewise estimated cut plane ab, but on the negative side of $a'b'$.

Our solution. We separate the generated triangulation for each intersected triangular element based on the cut plane which is connected by the intersection points of each intersected triangular element. For instance, the zoomed part of figure 3.19 illustrates an intersected element that will be refined into a triangulation. All the elements in this triangulation are grouped into two depending on whether they are on the positive or negative sides of the piecewise cut plane (the red dotted line). In this manner, neither need the extra information provided by a locally modified mesh nor the complex curved cut surface. Therefore this method is efficient to handle arbitrarily-shaped cuts at a local level.

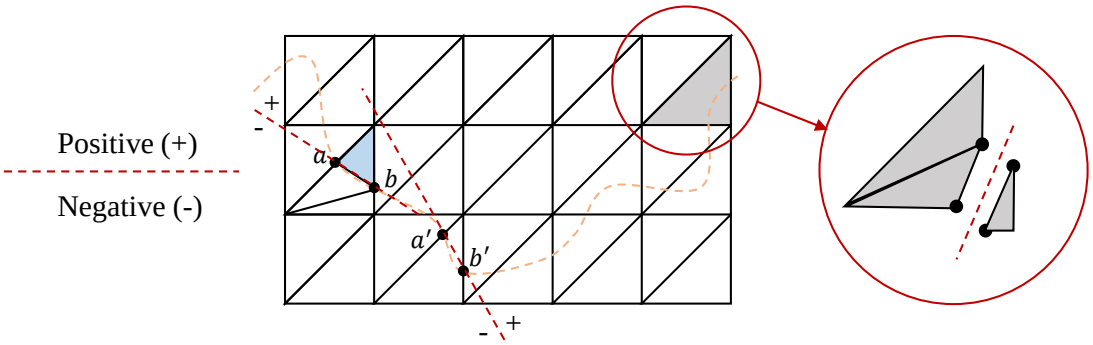


Fig. 3.19 Estimating the arbitrary-shaped cut edge by using the piecewise linear cuts within each intersected triangle. Determining on which side each sub-triangle places on the corresponding linear cut path and separating the material model.

3.6.2 Partial Cut

The difficulties. The difficulty of representing a partial cut within a single element appears when the cut edge is required to provide realistic physical behaviours and correct collision response. A partial cut within a triangular element indicates the cut is terminated inside the element. In such a situation the estimated linear cut path cannot be identified as there is only single intersection point. On the other hand, the realistic physical attributes should be re-projected properly to the cut edge. So-called "meshfree" methods, such as the XFEM and VNA avoid the mesh discontinuity by decoupling the computational model and rendering model so that the cut can be represented by rendering technologies without any modification to the topological structure of the underlying computational model. Highly-detailed cuts can then be generated graphically hence avoiding the refinement of the computational model. However in this manner the physical attributes at the edge cannot correctly represented.

Our solution. We use a unique discretisation model for the computation, rendering and collision detection. When a cut does not completely separate a single element, we utilise the end point of the cut path that terminates inside the element to construct the piecewise cut path. After refining the intersected element with the single intersection point and the end point of the cut path the sub-triangles are grouped into two by the rules proposed in section 3.6.1. Therefore a partial cut can be efficiently represented without special treatment. We initially restrict a piecewise straight cut within an element as shown on the left in figure 3.20. If any of generated sub-elements are along the cut and recognised as ill-shaped, a new vertex will be added at the place that holds the maximum curvature of the corresponding curve segment to support the non-linear subdivision within the element.

3.6.3 Re-cut and Cross Cuts

The difficulties. The re-cuts and cross-cuts required in the model allow repeated cuts at the same area that been modified in the previous cut process. Analogously in cross-cuts the element at the cross point will meet multiple cuts. A method that allows such re-cuts must avoid decreasing the

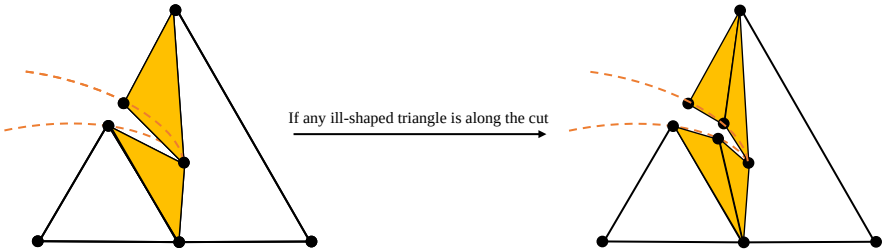


Fig. 3.20 Linear cuts are initially allowed for those which terminate within a triangular element. If any of the sub-elements along with the cut need to subdivide, a new vertex will be added at the place that holds the maximum curvature.

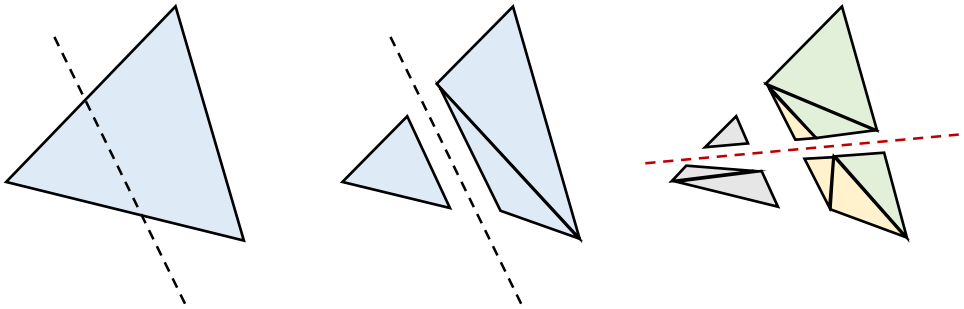


Fig. 3.21 An example of re-cut (also recognise as cross-cut). After the original blue triangle (left) has refined into triangulation, each blue sub-triangle (middle) are valid for the second cut (right).

number of the elements in the material model. Just removing the whole spring (in mass-spring system) or element will result in no mesh remaining after several cuts. Also, the re-cuts and crossing cuts assume each edge in the mesh is topological independent for the correct intersection detection. This is hard to achieve in graphical applications, where the cut edge are visually represented by increasing the level of detail in the rendering model (Koschier et al. 2017). On the other hand such graphical-based methods employ duplication (VNA) or an enrichment function on the basis functions (XFEM) struggle to project the correct physical attributes onto the generated elements as they are assigned these estimated physical properties from the previous solution. Also, it is complicated and time-consuming to introduce new discontinuous basis functions in those graphical-based methods.

Our approach. In our cutting method, both the computational and rendering models are refined from the previous cutting process. Each of the generated sub-triangles will be projected with the full physical and topological information. Figure 3.21 provides an example of re-cut. The triangle at the cross point support re-cuts when applying a cross-cut. The original triangle (left) is refined and separated by the first time cut which is denoted by the black dotted line (middle). For preparing the multiple cuts in the same area each sub-triangles are projected with full physical and topological information. In this manner, the repeat cuts can apply to any area of the mesh, even the new triangles (right). Hence the accurate intersection detection is guaranteed as each edge in the mesh is explicitly known.

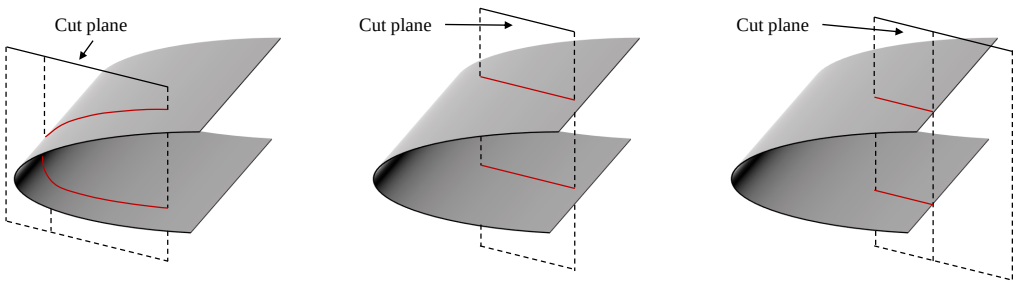


Fig. 3.22 Applying a cut on an non-plane surface model may result single incision (left) or multiple incisions (middle, right).

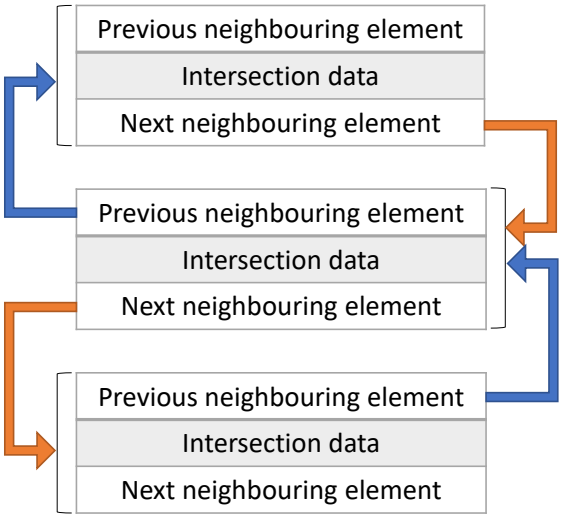


Fig. 3.23 The **Group Edge Map** system. In each item of the list, all intersected elements are linked by a pointer.

3.6.4 Overlapping Cuts

The difficulties. In the 3D environment, the cut is no longer a line but a plane. It may intersect at multiple edges of the model. The difficulty is to detect such complex intersection and handle the refinement simultaneously. Busaryev et al. (2013) applied a layered model in which constraint springs connect each layer. The multiple cuts are generated by recognising which model they belong to. However such a strategy is invalid in a single surface model. As shown in figure 3.22, a single cut applied on a non-planar model can cause a single cut incision (left) or multiple incisions (right). Moreover it can be time-consuming to group intersected triangles in a separated layered model in which each layer is independent of any other layer.

Our approach. To efficiently identify and group all intersecting triangular elements we establish a dynamic edge map. Each original triangle and the generated ones due to the refinement have a unique index number which is stored as the key in the map. The nodal indices of each edge in the mesh will be stored as the value of the corresponding triangle index. The edge map will be modified when adding or removing any triangle due to the cuts. In this manner we can easily categorise all of the intersected triangles by examining whether they are adjacent and share the same intersection edge. Therefore multiple cuts can be resolved without special consideration.

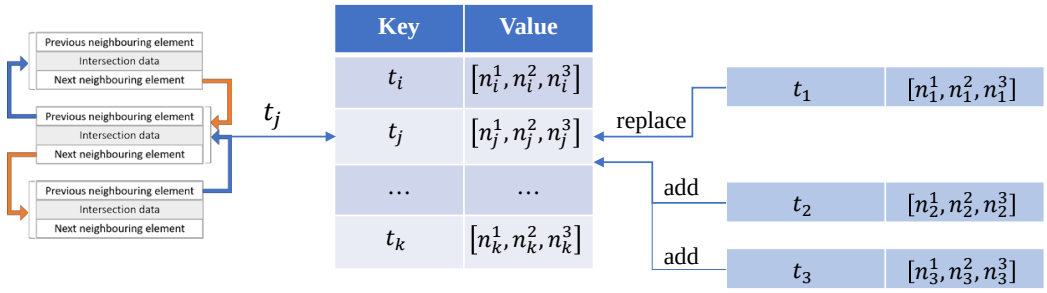


Fig. 3.24 Updating the dynamic **Look-at map.** with the information of new sub-elements.

Dynamic Edge Map. There are two significant components in our **Dynamic Edge Map** system, i.e., the **Group Edge Map** and **Look-at Map**. After applying a cut to the simulated material, each intersected triangular element will be put in a group in which it at least shares one edge with any other elements in this group. For each group, the indices of elements and their association are stored by a Standard Template Library (STL) list of the c++ Standard Library. As shown in figure 3.23, each intersection data structure collects the index of the intersected triangular element and the indices of the vertices for all intersected edges. Each intersection data stores a pointer which points to its neighbours which also have been cut. Each group includes all the intersected triangular elements which are end to end. These group will be deleted after the cuts are reproduced.

Let i be the number of intersected elements which will be allocated into n groups. In each group, the ACDM refinement runs with the *Look-at Map* for each intersected element. Therefore, the computation complexity of solving all the group is

$$O(n) = O(n_1) + O(n_2) + ... + O(n_i), \tag{3.28}$$

where $n = n_1 + n_2 + ... + n_i$, and n indicates that n cuts have been generated from one time cutting. In the **Look-at Map**, the *Key* is the index of an intersected element and the *Value* is the indices of its vertices. This map is pre-constructed before the simulation starts.

Figure 3.24 gives an example of how the **Dynamic Edge Map** system updates to integrate new sub-elements. First, the indices of the intersected element (in this example it is t_j) are taken to find the corresponding *Value*. The computation complexity of finding a *Value* by its *Key* in a STL list is $O(logn)$, where n indicates the number of records in the list. Afterwards, the intersected element will be refined by the ACDM method and its record in the list will be replaced by one of the generated sub-elements. The information of the rest of the sub-elements will be inserted immediately.

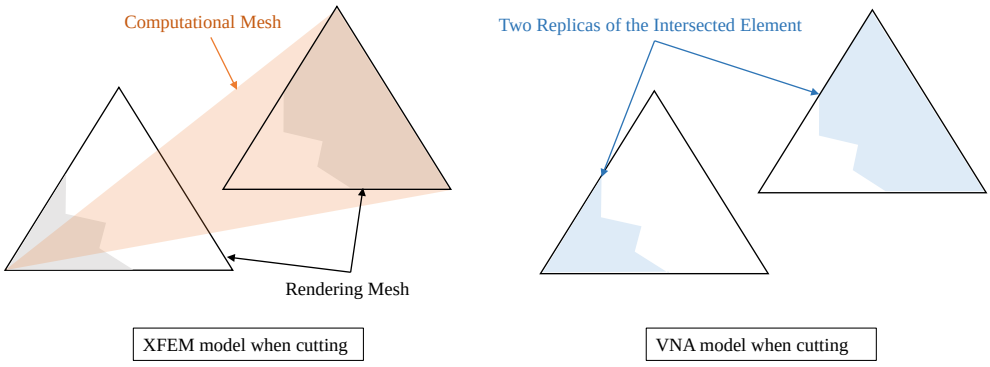


Fig. 3.25 Graphical-based cutting methods. In XFEM (left) the behavior of cut edges is assigned proportionally from the underlying computation model. In VNA (right) the graphically visible cut edges has improper physical attributes in duplicates.

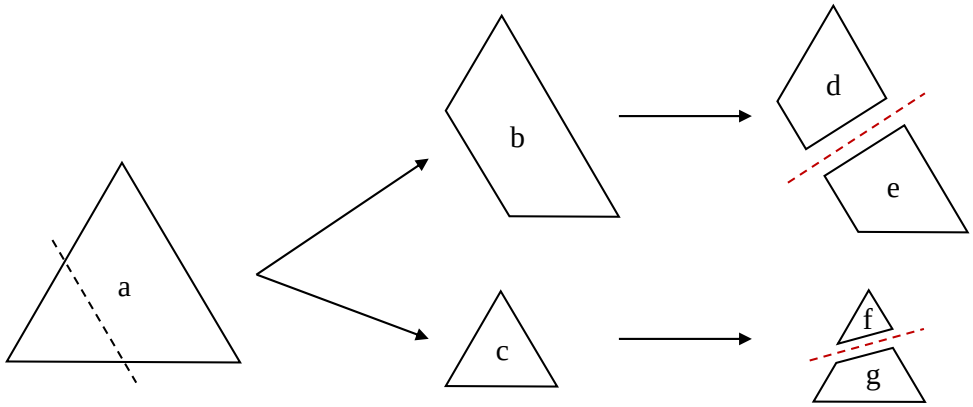


Fig. 3.26 Our mesh-based approach generate sub-meshes whose edges are explicitly known allowing re-cuts and accurate intersection detection.

3.6.5 Cut Out a Partial Mesh

The difficulties. The difficulty of cutting out a partial mesh is to ensure its independent topological and physical properties and behaviours. The separated meshes should be independent ensuring accurate intersection detection and continuous physical behaviours in the following cut processes. It is hard to achieve when employing graphical-based cutting methods. As shown in figure 3.25, XFEM (left) maintains the model for computation (the red triangle) and enriches the basis functions depends on the number of parts that have been cut. In this method the deformations and the physical attributes of the cut edges are partially assigned from those in the computational mesh element which it belongs to. In contrast the VNA method (right) generates replicas of the intersected element and makes the visible part of each replica dependent on the placement of the cut plane. However both approaches are limited in applications that require repeated cuts and correct intersection detection. This is primarily because in both approaches the cut edges are not explicitly known for the intersection detection.

Our approach. We aim to provide a high-performance mesh that allows unlimited cuts with highly-detailed physical behaviours. In contrast to the graphical-based methods we separate the

original elements and maintain the independence of each part cut out. Figure 3.26 describes our solution for providing high-performance sub-meshes under two cuts. The original mesh a is separated into two sub-meshes, b and c, by the first cut (denoted by the black dot line). In the following cuts each of the meshes, b and c, are identified as an original mesh that can be separated by a second cut (illustrated by the two red dashed lines). The generated sub-meshes b, e, f, g have the cut edges which support accurate intersection detection and physical behaviours.

3.7 Conclusion

When compared with the use of volumetric modes surface model cutting is essential for in practical applications such as virtual surgery. We achieved a high-performance method in the area of mesh cutting of 3D surfaces modelled by the physically-based finite element method. The main contributions of this work can be summarised as followed. Firstly, it allows smooth user-defined cut trajectories with arbitrary shapes. The flexible cubic Bézier curve efficiently approximates the exact cut trajectory with a smooth estimation but avoids the substantial computational time due to the instant cut detection. However highly-detailed cuts or complicated cut situations always require a substantial computation cost that is unaffordable for real-time simulations. Moreover the flexible non-uniform refinement can easily generate ill-shaped elements which also can influence the computational efficiency of the simulated system. We have provided the geometrical and numerical solutions for handling the ill-shaped elements thus ensuring a high-performance cut representation which can be used in a real-time environment.

Chapter 4. Experiment Plan and Setting

4.1 Experiments Plan

4.1.1 Accurate Pre-defined Cuts Reproduction

Our goal is to reproduce detailed cuts whose shape can be pre-defined by users at a real-time rate, i.e., at least 30 FPS during the cut. We compare the cuts reproduced by the proposed framework with the state-of-the-art (Manteaux et al. 2015). Simultaneously, we also show the lowest FPS before, during and after the cut when the different numbers of new vertices which range from 76 to 194 are created.

4.1.2 The Influence of Aspect Ratio and Area Ratio

In Section 3.2.2, we analysed how the size and shape of a triangular element influences the mesh quality in FEM-based modelling from a theoretical perspective. The *Aspect Ratio* and *Area Ratio* of the new elements are chosen to measure the quality of a triangular element. We evaluate the impacts of different values of the *Aspect Ratio* and *Area Ratio* on the condition number of the global stiffness matrix and the time spent on solving the system of finite element equations. The range of *Aspect Ratios* tested is from 2.05 to 29.05. The *Area Ratio* ranges from 1.11% to 29.95%.

4.1.3 Stiffness Matrix Optimisation

In Section 3.5, we proposed two optimisation methods, including AS-CGM and ES-CGM, to keep the global stiffness matrix well-conditioned after the cuts. We evaluate the performance of the two optimisation methods in context. We first choose a preferable value for the exponent p for the p-norm by comparing its optimisation efforts on the condition number of the global stiffness matrix and the corresponding time needed for the CG solver. Afterwards, we determine the performance of AS-CGM and ES-CGM when solving multiple cuts in which the number of entities of the global stiffness matrix is up to 10,687. Finally, we compare the AS-CGM and ES-CGM methods with our baseline method (Xin et al. 2018).

4.2 Experimental Environment Setting

All experiments were run on an Alienware laptop with four 8-core Intel(R) Core(TM) i7-6700HQ Processors at 2.60GHz. Each core is equipped with 256KB L1 cache, 1.0MB L2 cache, and 6.0MB shared L3 cache per socket. The machine has a 16.0GB SODIMM 2133MHz memory and two graphics processing units, including Intel(R) HD Graphics 530 and NVIDIA GeForce GTX 970M.

4.3 Related Libraries and Toolkits

The simulation framework is built in C++ with Visual Studio IDE. The application programming interface (API) is OpenGL 4.4. The basic linear algebra algorithms, such as matrix-matrix products, matrix-vector products, and sparse matrix decomposition are computed by the Eigen 3.3.7 library (Guennebaud et al. 2010). The CG solver, the ACDM subdivision, AS-CGM, and ES-CGM are written by the author.

4.3.1 Physics Engine Class Diagram

The physics engine is built from scratch by the author. The class diagram (figure 4.1) is directly generated from Visual Studio for the framework project.

In the ***PhysicsEngine*** class, the *Update* function consists of general runtime modules, including:

- Broadphase collision detection. This step identifies possible collisions between objects with constructing world space partitioning systems.
- Narrowphase collision detection. This step takes the list provided by the broadphase collision detection and accurately collides all objects with building collision manifolds.
- Solve Constraints & Collisions. This step solves all the velocity constraints in the physics system, including both collision constraints and misc world constraints, such as air resistance. The proposed matrix optimisation methods are integrated at this stage to optimise the global stiffness matrix before it is used to update the system.
- Update physics objects. This step moves all physics objects through time and updates the positions.

4.3.2 Material Properties

The shear modulus differentiates the material stiffness of various tissues. The Young's modulus E measures such a material property by

$$E = \frac{\sigma}{\varepsilon}, \tag{4.1}$$

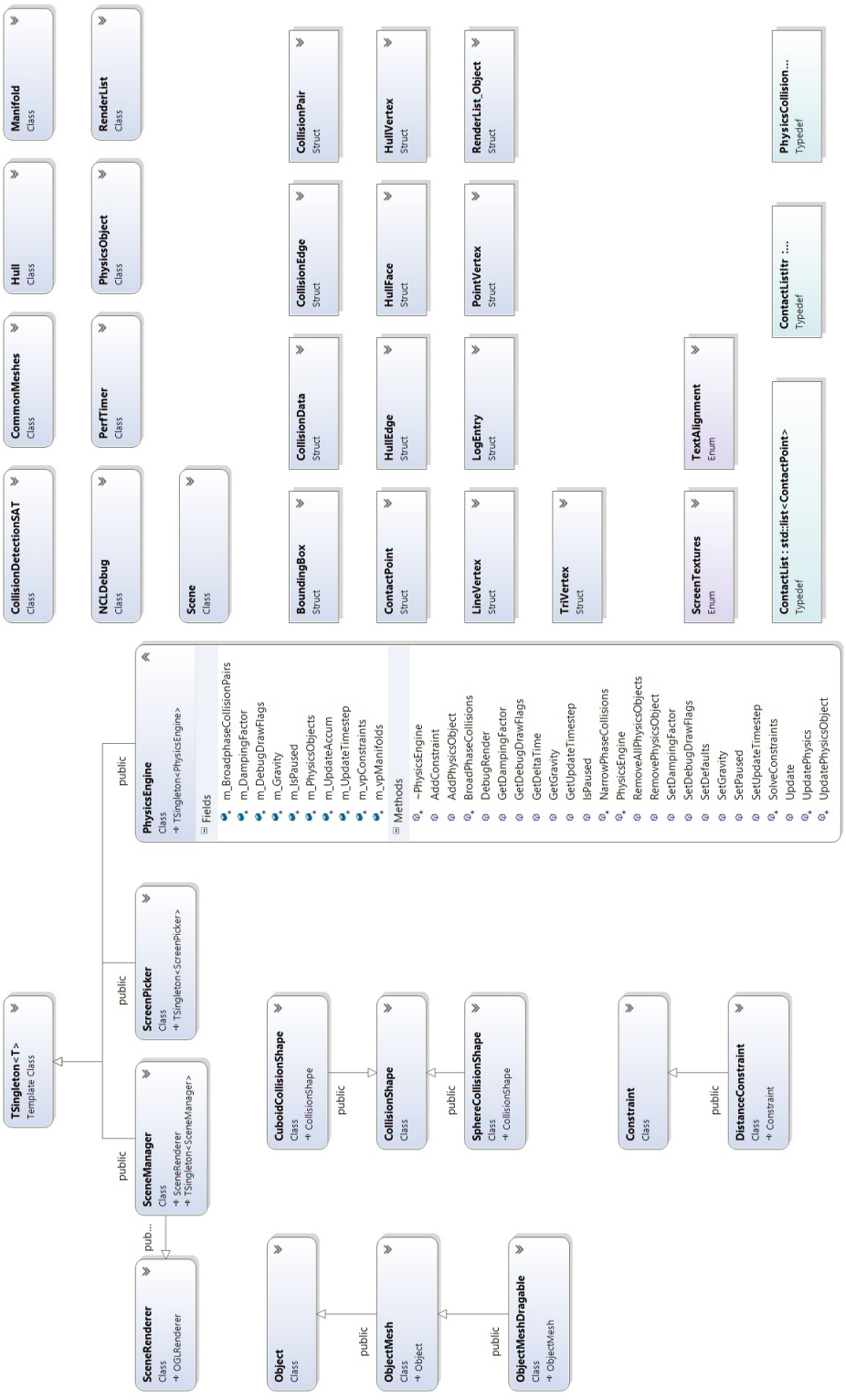


Fig. 4.1 The physics engine class diagram of the proposed framework.

Young's Modulus	Shear Modulus	Mass Density	Poisson Ratio
3.1 Gpa	0.51 Gpa	1.3 g/cm^3	0.63

Table 4.1 Material properties for the simulated wool surface material.

where σ is the stress and ε is the strain. The value of shear modulus defined in our simulation and other material properties are list in table 4.1.

Chapter 5. Results and Evaluation

In this chapter, we will evaluate the performance of our method in a variety of contexts, including the visual results for the representation of the cut path, mesh quality, computational time, the level of details around the cut and the visual results for the complex cutting situations in the 3D environment. The shape of each cut is defined by a cubic B*é*zier curve where the four control points are defined by four mouse clicks from the user. We first determine the impact of both the *Aspect Ratio* and *Area Ratio* of the new elements on the performance of the simulation, aiming to pick the preferable value for the framework. We compare the visuals of the pre-defined cuts and the lowest FPS during the cuts with the state-of-the-art method (Manteaux et al. 2015). We evaluate the computational performance of our method by implementing such a method into the cutting of the dynamic elastic 2D material models that have different numbers of original triangular elements. We repeat each cutting ten times and average the values to obtain the data that is used for the comparison and analysis.

5.1 The Influence of Ill-Shaped Triangular Elements

Our goal is to eliminate the impact of the ill-shaped triangular elements on the performance of the cutting simulation. The performance of a physically-based real-time cutting simulation is largely decided by the condition number of the global stiffness matrix, the time spent on the CG solver and the lowest FPS during the cuts.

In this section, we observe the values of two factors, *Aspect Ratio* and *Area Ratio*, and their impact on the performance of the cutting simulation in a variety of cutting scenarios.

- The *Aspect Ratio* determines whether a triangular element has a large angle or small angle.
- The *Area Ratio* illustrates how large a triangular element is compared with the element which is initially constructed and has not been cut.

Each cut is applied on the simulated surface material which initially has 2,500 vertices and 4,802 triangular elements. No optimisation method is implemented. Figure 5.1 shows the patterns used to observe the impact of different values of *Aspect Ratio* and *Area Ratio* on the performance of the simulation. For each pattern, we randomly cut 10 times to obtain the average *Aspect Ratio* and *Area Ratio* of all new elements and store them in table 2.1. Table 5.1 shows the number of intersected elements in each cut. The condition numbers of the global stiffness matrix and the lowest FPS during reproduction of each pattern follow.

Pattern	5.1 - (a)	5.1 - (b)	5.1 - (c)	5.1 - (d)
#Intersected Elements	151±3	152±6	151±4	152±2

Table 5.1 The number of intersected elements in the patterns shown in figure 5.1.

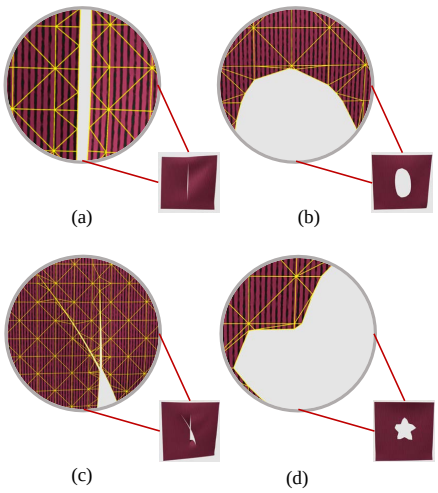


Fig. 5.1 Four cutting patterns used to determine the performance of the simulation. For each pattern, the cut is repeated ten times.

Pattern	#Aspect Ratio	#Area Ratio	#Condition Number	#CG Solver (ms)	#Lowest FPS (during)
5.1 - (a)	4.26±1.16	0.16±0.17	$(1.37\pm0.18)\times10^5$	68.21	13.39
5.1 - (b)	4.64±1.38	0.37±0.23	$(5.08\pm0.70)\times10^5$	74.27	12.48
5.1 - (c)	14.08±7.12	0.06±0.04	$(5.65\pm0.38)\times10^5$	114.04	8.30
5.1 - (d)	14.90±6.44	0.33±0.26	$(1.42\pm0.32)\times10^6$	193.50	4.99

Table 5.2 The impacts of *Aspect Ratio* and *Area Ratio* on a variety of cutting patterns. For each cut, the number of intersected elements ranges from 144 to 166.

As shown in table 5.2, during the reproduction of each pattern, pattern (a) obtained the highest FPS, in which both the values of *Aspect Ratio* and *Area Ratio* are relatively low. Compared with pattern (a), pattern (b) which has the higher *Area Ratio* requires a longer time for the CG solver. Moreover, it can be concluded by comparing patterns (b) and (d) that a higher *Aspect Ratio* slows down the process of the CG solver. Both high *Aspect Ratio* and *Area Ratio* result in the condition number of the global stiffness matrix being high. However, the impact of *Area Ratio* is not as crucial as the *Aspect Ratio*.

In summary, both high *Aspect Ratio* and *Area Ratio* of the new elements result in the increase of condition number of the global stiffness matrix. A global stiffness matrix which has a high condition number slows down the CG solver process; therefore, the FPS during the cut is decreased. Besides, since none of the patterns can be achieved at a real-time rate, i.e., 30 FPS, achieving a cut in real-time meets a bottleneck when the condition number of the global stiffness matrix is high which is more than 5.00×10^5 in the experiments.

5.2 The Criteria for *Aspect Ratio* and *Area Ratio*

In Chapter 3, we proposed a new subdivision scheme which aims to optimise the mesh quality by keeping both the *Aspect Ratio* and *Area Ratio* of the new elements low. In this section, we identify the preferable value of *Aspect Ratio* and *Area Ratio* as the criteria when operating our subdivision scheme.

As shown in table 5.2, compared with other patterns that were tested a flower-like pattern (figure 5.1 - (d)) is most difficult to achieve in real-time. In this section, flower-like patterns are used to determine the impact of different values of *Aspect Ratio* and *Area Ratio* on the performance of the proposed ACDM subdivision.

5.2.1 Experiment Settings

We apply a punch cut on a simulated surface which initially has 2,500 vertices and 4,802 triangular elements. The cut is repeated 20 times with generating punches with random shapes. For each time, the new subdivision is operated until the average *Aspect Ratio* of the mesh achieves a specific value. Both the average *Aspect Ratio* and the corresponding average *Area Ratio* are shown in figure 5.5. Moreover, other factors are also detected, including

- the increase of new elements.
- the lowest FPS during each cut.
- the time spent on the CG solver.
- the change of the condition number of the global stiffness matrix.

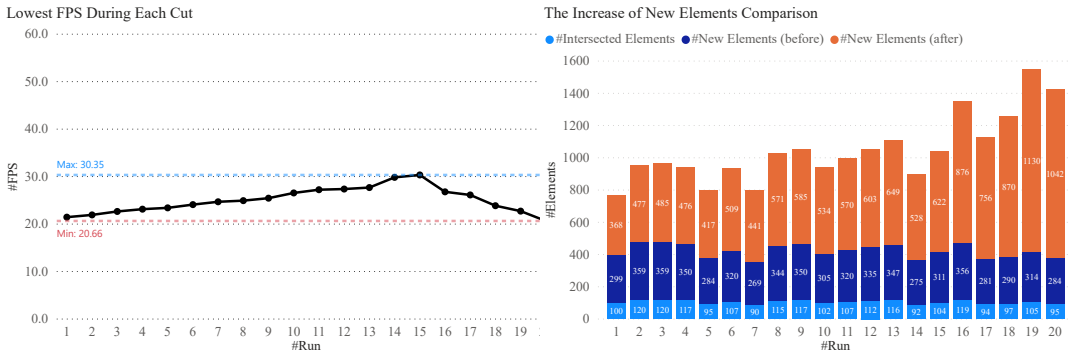


Fig. 5.2 The lowest FPS during each cut and the corresponding new elements increase before and after implementing the new subdivision method.

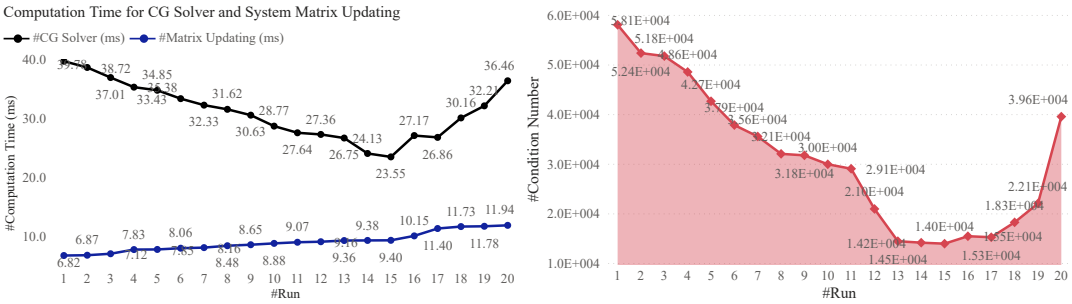


Fig. 5.3 The time spent on updating the global stiffness matrix and the resulting condition number.

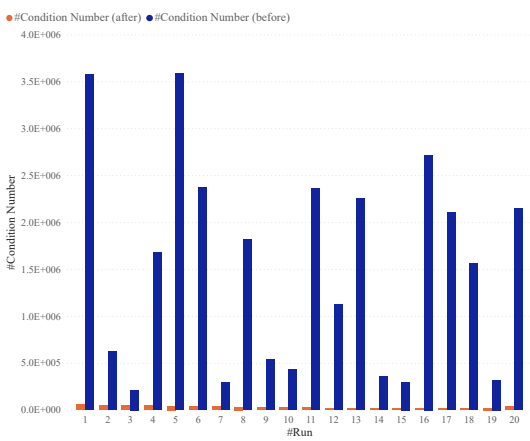
5.2.2 Results and Analysis

For a fair comparison, each cut has a similar size which is shown on the right side of figure 5.2. This figure also provides the number of new elements when no optimisation method is implemented.

Time Efficiency

The time efficiency of a cutting method reflects how fast the cut can be reproduced. Therefore, we determine the lowest FPS during the reproduction for each cut which is shown on the left side of figure 5.2. In general, our subdivision method performs stably between 20.66 FPS and 30.35 FPS which is on a par with the minimum FPS expectation for achieving a cut in real-time, i.e. 30 FPS. Specifically, the lowest FPS continuously increases between runs 1 and 15; while it begins to go down quickly from run 15.

The FPS reflects the computation time spent on updating the simulation after applying a cut. The time for updating the global stiffness matrix and the CG solver accounts for most of the total computation time. Figure 5.3 collects the computational time required for updating the global stiffness matrix and the CG solver. As shown in the figure, the time spent on updating the global stiffness matrix increases repeatedly and goes more quickly from run 16. Conversely, the time for the CG solver process decreases nearly linearly before run 15 and increase unstably afterwards.



#Run	#Condition Number (after)	#Condition Number (before)
1	5.81E+004	3.58E+006
2	5.24E+004	6.23E+005
3	5.18E+004	2.15E+005
4	4.86E+004	1.68E+006
5	4.27E+004	3.59E+006
6	3.79E+004	2.37E+006
7	3.56E+004	2.95E+005
8	3.21E+004	1.82E+006
9	3.18E+004	5.40E+005
10	3.00E+004	4.35E+005
11	2.91E+004	2.36E+006
12	2.10E+004	1.13E+006
13	1.45E+004	2.26E+006
14	1.42E+004	3.54E+005
15	1.40E+004	2.99E+005
16	1.55E+004	2.72E+006
17	1.53E+004	2.11E+006
18	1.83E+004	1.56E+006
19	2.21E+004	3.20E+005
20	3.96E+004	2.15E+006

Fig. 5.4 Compare the condition number of the modified system before and after applying our subdivision.

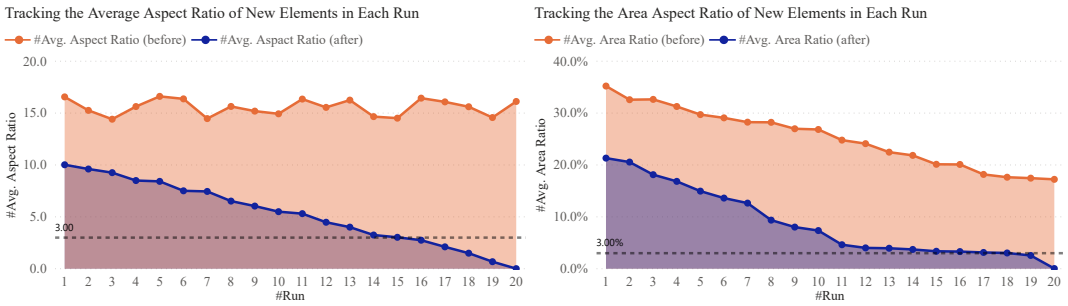


Fig. 5.5 The average *Aspect Ratio* and *Area Ratio* in each run. The average *Aspect Ratio* of new triangular elements. In each cut, the subdivision operates until the *Aspect Ratio* meets at different levels.

The observation above indicates that the time efficiency of the proposed subdivision largely depends on the size of the initial mesh and modification of the global stiffness matrix. Moreover, to make the average *Aspect Ratio* lower requires a higher level of subdivision. However, the new subdivision can undoubtedly ensure the simulation can be run at a real-time rate during a cut in run 15. The next part of this section will take a closer look at the setting of the average *Aspect Ratio* and the corresponding average *Area Ratio* in each run. The goal here is to choose the preferable criteria value to achieve the best performance by using the new subdivision.

Mesh Quality Optimisation

The goal of the proposed new subdivision is to maintain a high mesh quality by reducing the condition number of the global stiffness matrix.

The right side of figure 5.3 shows the condition number of the global stiffness matrix after implementing the new subdivision in each cut. As shown in the figure, the condition number eventually decreases from run 1 to 12 and remains steadily low between runs 13 and 17. However, it rapidly goes back to being high from run 18. Figure 5.4 lists the values of the condition number in each run and compares it with the values before implementing the proposed subdivision. Figure 5.5 provides the average *Aspect Ratio* constrained for each run and the corresponding average *Area Ratio*. As shown in the figures, the average *Aspect Ratios* between

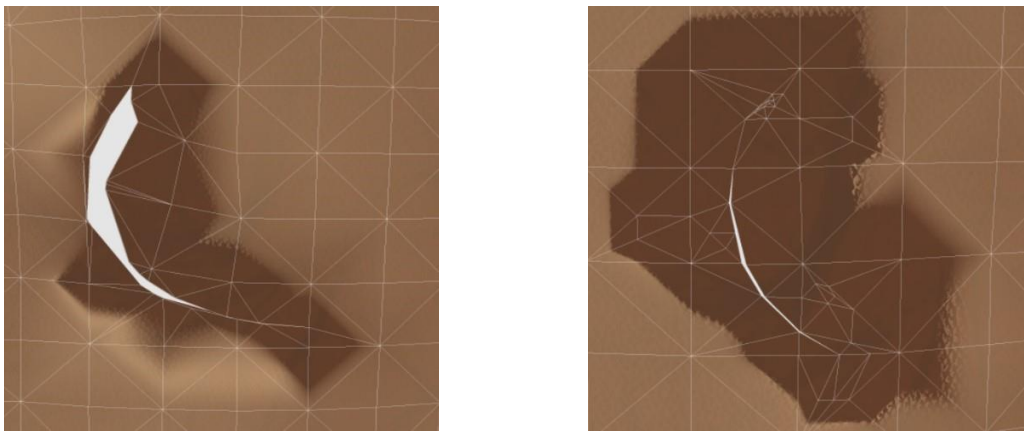


Fig. 5.6 The deformation around the cut before (left) and after (right) applying the proposed refinement method.

run 13 and 17 are around 3, where the corresponding average *Area Ratios* from run 13 to 18 keep at stable around 3%.

Figure 5.6 compares the dynamics of the cut generated before and after applying the proposed refinement method. As shown on the left, when integrating the dynamic physically-based behaviours the ill-shaped triangular elements that is generated can easily lead to visual errors. Conversely the proposed refinement method can represent a convincing cut edge with correct physically-based deformations.

In summary, the new subdivision method can significantly decrease the condition number of the global stiffness matrix after applying a cut. For each cut, the average number of intersected elements and new elements is 100 and 522, respectively. Moreover, it provides the new subdivision with the best optimisation performance when determining whether the *Aspect Ratio* of a new element is higher than 3 and the *Area Ratio* is larger than 3%. Besides, the new subdivision guarantees that the simulation during the reproduction of a cut can run at a real-time rate when the average *Aspect Ratio* is lower than three and the average *Area Ratio* is higher than 3%.

5.3 The Improvements on *Aspect Ratio* and *Area Ratio*

In Section 5.2, the proposed new subdivision has shown its capability to significantly decrease the condition number of the global stiffness matrix. In this section, we determine the optimisation efforts of the proposed subdivision by applying a single cut, intersecting cuts and multiple cuts which have a wide range of sizes.

Initially, we construct an elastic surface material with isosceles right triangular elements in which the *Aspect Ratio* is approximately 1.41. There are initially 2,500 vertices and 4,802 triangular elements. The ACDM subdivision starts after the *Conditional Delaunay Triangulation*

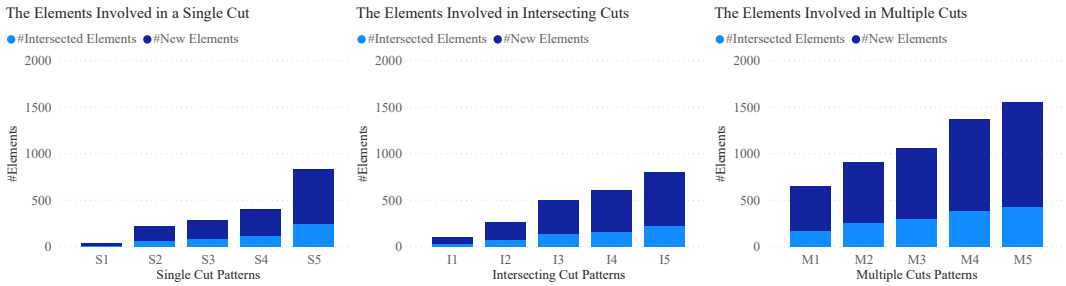


Fig. 5.7 The number of intersected elements and the new elements created by implementing the ACDM method.

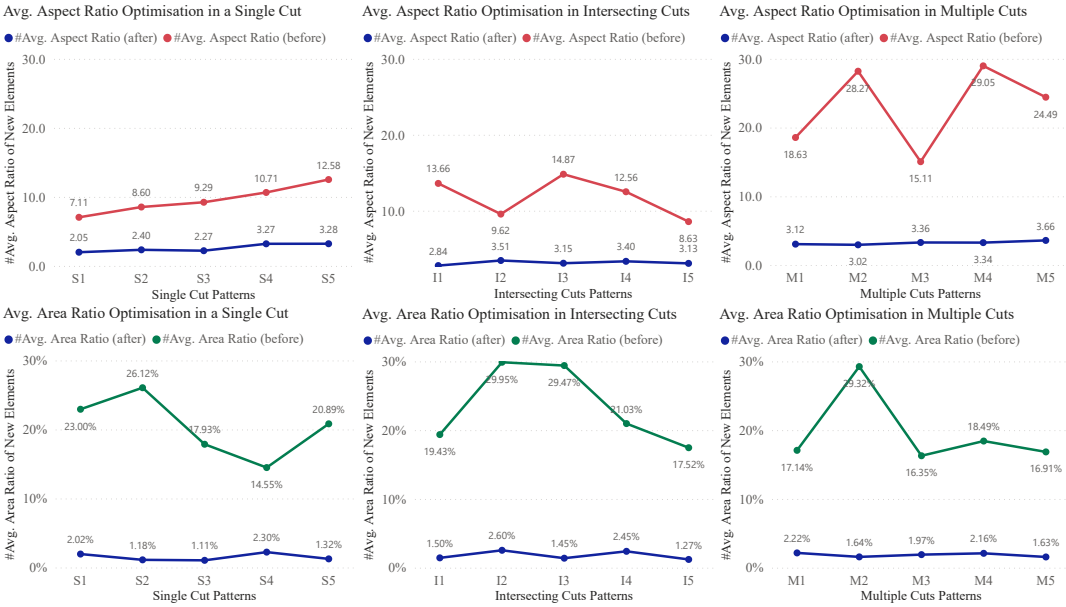


Fig. 5.8 The optimisation performance of the proposed subdivision method on the average *Aspect Ratio* and *Area Ratio* in the modified model mesh in a variety of cutting scenarios.

is generated. Any triangular element for which *Aspect Ratio* is higher than 3 and the *Area Ratio* is larger than 3% will be further refined by the ACDM method.

Figure 5.9 shows the visuals of cuts used for collecting the data shown in Figures 5.7 and 5.8. Figure 5.7 provides the number of intersected elements and new elements after applying each cut. In particular, the number of new elements does not include the intersected elements.

Figure 5.8 illustrates the average *Aspect Ratio* and *Area Ratio* in the mesh before and after employing the ACDM method. Generally, the ACDM method can secure an average *Aspect Ratio* lower than 4 and the average *Area Ratio* fluctuates between 1.11% to 2.60%. Specifically, compared with a single cut, both intersecting cuts and multiple cuts can more easily result in a high *Aspect Ratio*. Moreover, compared with applying no optimisation method, the ACDM method can effectively decrease the average *Aspect Ratio* to only 15% and 36% when reproducing intersecting cuts and multiple cuts, respectively.

The average *Area Ratio* varies for each cutting pattern. In particular, the average *Area Ratio* is smaller when there are more cursive cuts. As shown in the second row of figure 5.8, the ACDM

Cutting scenario	Reference	Grid Size	#Frame		#Rendering vertices		#Collision nodes		#Integration nodes		Lowest FPS		
			initial	final	initial	final	initial	final	initial	final	before	during	after
Single cut	(Manteaux et al. 2015)	40×40	5	5	81	2111	200	200	200	200	60.00	14.40	60.00
	Ours	**-**	**-**	**-**	**225**	**302**	**225**	**302**	**200**	**302**	**80.09**	**60.89**	**77.98**
Intersecting cuts	(Manteaux et al. 2015)	68×68	5	5	4225	4889	200	200	200	200	45.00	7.20	35.70
	Ours	**-**	**-**	**-**	**225**	**358**	**225**	**358**	**225**	**358**	**80.10**	**50.64**	**57.63**
Punch	(Manteaux et al. 2015)	50×50	5	12	4225	8253	200	200	200	200	60.00	6.80	45.00
	Ours	**-**	**-**	**-**	**225**	**338**	**225**	**338**	**225**	**338**	**80.10**	**46.74**	**47.56**

Table 5.3 The comparison of our method with state-of-the-art method (Manteaux et al. 2015).

method can significantly decrease the average *Area Ratio* to less than 3% in all the three cutting scenarios.

5.4 Compare with State-of-the-art

Manteaux and his colleagues propose a method to reproduce detailed pre-defined cuts at an interactive rate. The method constructs the simulated surface material by employing different meshes for rendering, deformation, and collision. In particular, the resolution of the visualisation mesh is much higher than the other meshes. Instead, we use one mesh for collision, rendering and deformation. Our strategy is to increase the minimum number of elements without losing details. Furthermore, we keep the global stiffness matrix well-conditioned to ensure physical accuracy and time efficiency.

In this section, we compare our method with Manteaux and his colleagues' method (Manteaux et al. 2015) in terms of two significant factors which are:

- The visuals of complex pre-defined cuts in a variety of cutting scenarios.
- The lowest FPS before, during and after the cuts.

For our method, all the data used in this section is obtained by averaging six experimental results. The visuals of the cuts and the number of vertices in the mesh will be expanded and discussed horizontally in the next section.

graphicx

Table 5.3 shows the comparison of our method with Manteaux and his colleagues' method in a variety of cutting scenarios. In their method, the frames are duplicated to increase the resolution of the rendering mesh. The grids are used to sample the collision nodes. Conversely, we use one mesh for deformation, rendering and collision. We run our simulation with a simulated surface material which has the same number of vertices as the collision and the integration vertices in Manteaux and his colleagues' method.

Our strategy is significantly different from Manteaux and his colleagues' method, but the lowest FPS can reflect the performance of the two strategies during the simulation (table 5.3). As shown in the table, Manteaux and his colleagues obtain the best performance when handling a single cut. However, in the same cutting scenario, the lowest FPS during the cut obtained by

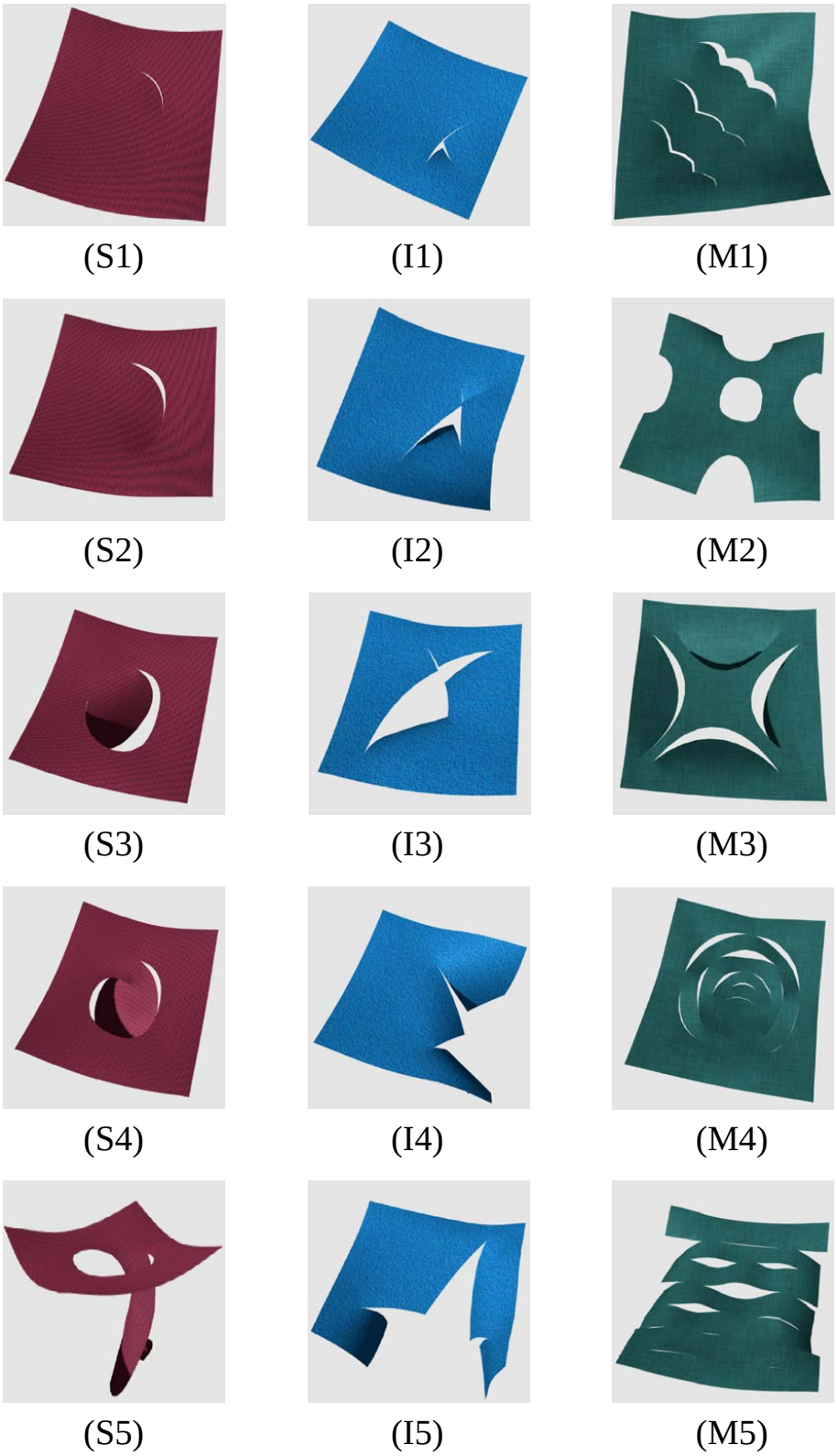


Fig. 5.9 The cut patterns used to determine the improvement of average *Aspect Ratio* and *Area Ratio* by using the proposed subdivision scheme in the single cut, intersecting cuts and multiple cuts scenarios respectively.

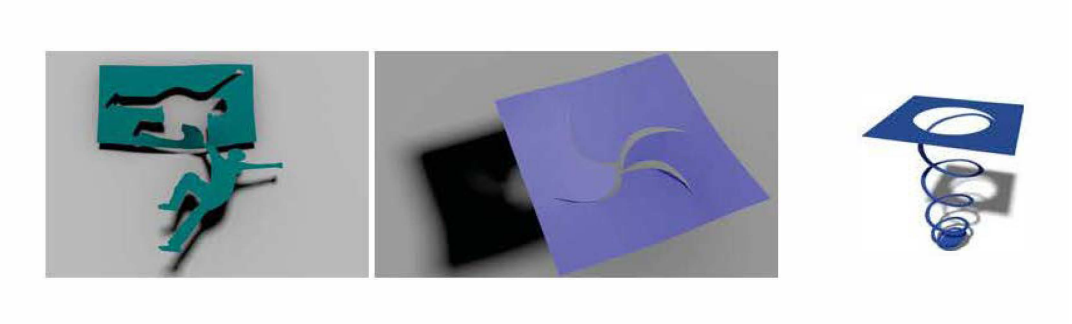


Fig. 5.10 The screenshot of the cuts provided by the state-of-art (Manteaux et al. 2015).

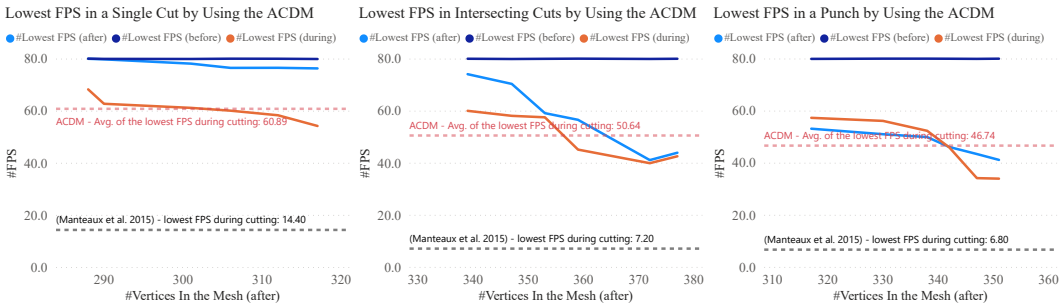


Fig. 5.11 Lowest FPS obtained by implementing the ACDM subdivision and the state-of-the-art method (Manteaux et al. 2015).

using our method is nearly seven times higher than their method. Moreover, after reproducing a single cut, our method can almost run at the initial FPS level. Manteaux and his colleagues' method finds it most difficult to reproduce a punch at an interactive rate. However, our method runs with 46.74 FPS during the punch and 47.56 FPS after the punch.

5.5 Detailed Pre-defined Cut

In this section we expand the results for averaging the data which is shown in table 5.3. Figure 5.10 provides the visuals of cuts reproduced by the state-of-the-art work (Manteaux et al. 2015) for reference.

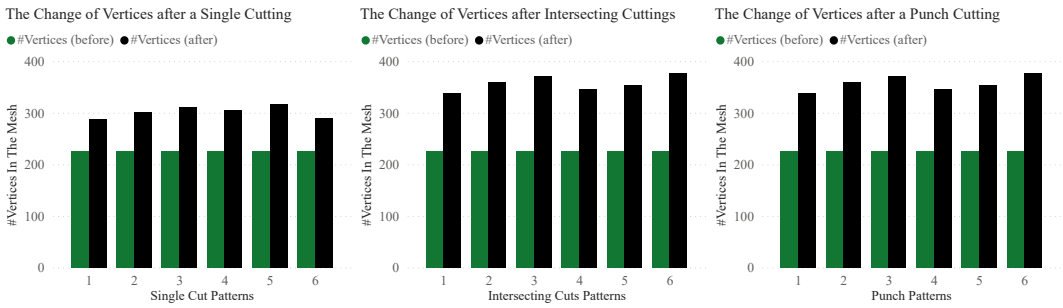


Fig. 5.12 The change of vertices in the mesh after applying a single cut, intersecting cuts, and a punch by using the ACDM subdivision. The six patterns in each cutting scenario are shown from Figures 5.13 to 5.15.

5.5.1 Single Cut Reproduction

Figure 5.13 illustrates the six patterns after applying a single cut to the simulated material by using the ACDM subdivision. Analysis of the corresponding lowest FPS is shown to the left of figure 5.11, the ACDM subdivision can reproduce a single cut while remaining on average 70 FPS during the cut. In Manteaux and his colleagues' experiments, 14.40 FPS is the highest FPS that they provided. In figure 5.12, reproducing pattern 5 which has more sharp cut edges requires us to increase the largest number of vertices compared with other single cut patterns. It indicates that sharp cuts can more easily produce ill-shaped elements than smooth cuts.

5.5.2 Intersecting Cuts Reproduction

Figure 5.14 shows the six intersecting cut patterns generated by the ACDM subdivision. Compared with other intersecting cut patterns, pattern 1 obtains the highest FPS during and after the cuts which are 60.12 FPS and 74.20 FPS, respectively. Manteaux and his colleagues can also reproduce this cutting pattern; however, the FPS they can maintain is only one-ninth of ours. The increase of the vertices in the mesh is illustrated in the middle of figure 5.12. Reproducing patterns 3 and 6 needs to create a similar number of vertices which are higher than those for other intersecting cut patterns.

5.5.3 Punch Reproduction

Simulating a punch needs to cut out parts of the mesh. Figure 5.15 gives the six punch patterns reproduced by using the ACDM subdivision. Compared with reproducing a single cut, reproducing a punch and intersecting cuts resulted in a lower FPS during and after the cuts. However, during a punch cut the lowest FPS is not significantly different compared to those during intersecting cuts which are 50.64 FPS and 46.74 FPS, respectively.

In summarise, the ACDM reproduces the intersecting cuts and a punch by increasing a similar number of vertices. However, the cuts which have more cursive or sharp edges, such as patterns 4 and 6, need the introduction of more vertices.

5.6 Stiffness Matrix Optimisations

In Chapter 3, we proposed two matrix optimisation methods, including ES-CGM and AS-CGM, to further optimise the global stiffness matrix. In this section, we first pick the preferable value for exponent p in the p-norm. Afterwards, we evaluate the optimisation performance of the two methods when solving the global stiffness matrices in which the number of elements ranges from 7,557 to 10,687. In our experiments, the pre-conditioned Gauss-Seidel iterative method is employed to solve the equations system. We show the superiority of the AS-CGM and ES-CGM methods by comparing our baseline method (Xin et al. 2018).

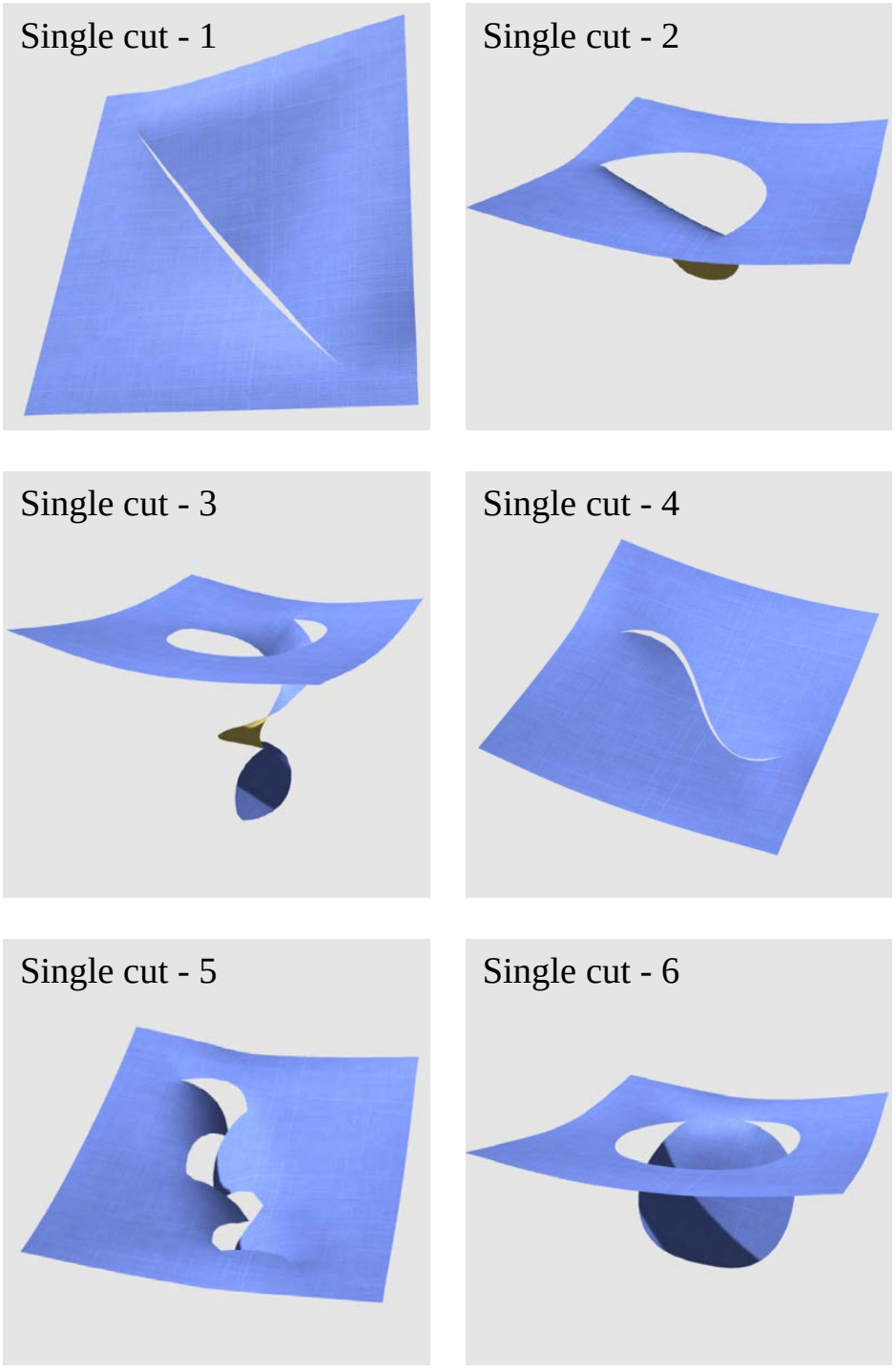


Fig. 5.13 The six patterns after applying a single cut by using the ACDM subdivision.

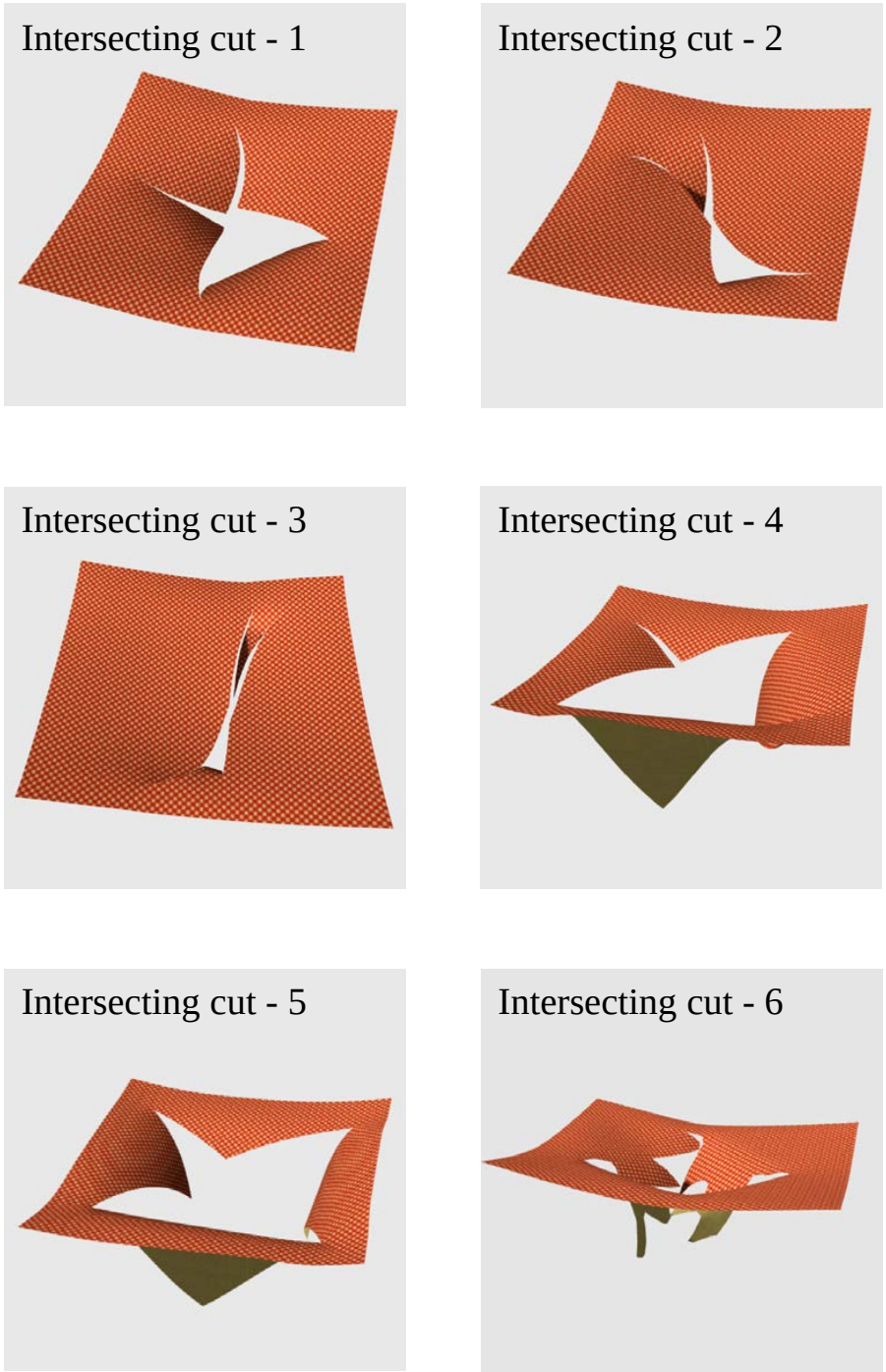


Fig. 5.14 The six intersecting cut patterns generated by using the ADCM subdivision.

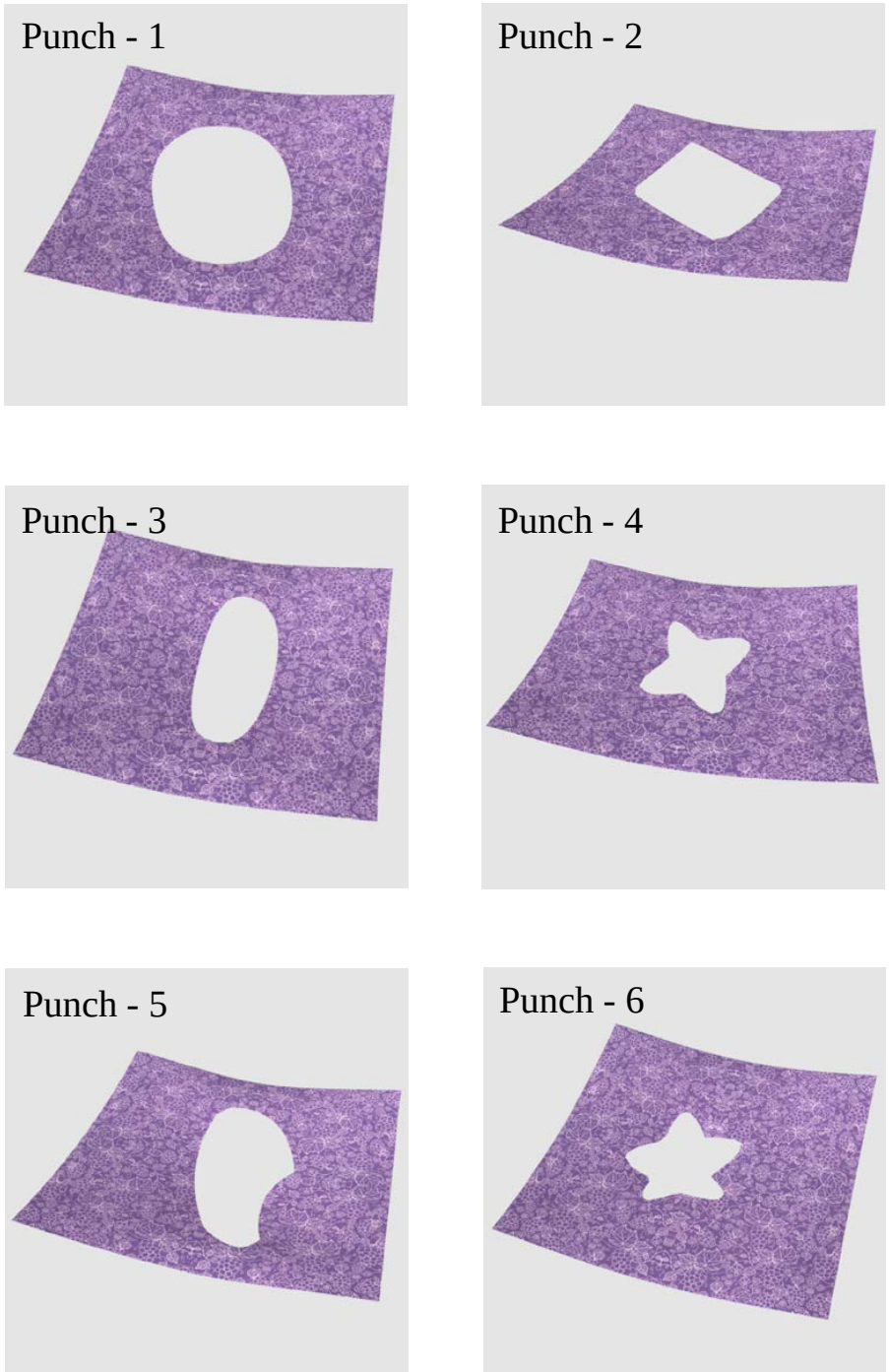


Fig. 5.15 The six punch patterns generated by using the ACDM subdivision.

Methods	#Intersected Elements	#Condition Number (final)	CG Solver (ms)
$L1$-Norm	160±6	(1.87*10^4, 0.91*10^5)	(66.26, 71.64)
log-Norm	156±3	infinite	infinite
p-Norm	157±5	**(8.23*10^1, 1.12*10^2)**	**(41.67, 50.86)**

Table 5.4 The condition number comparison and the time spent on the CG solver.

(a) (b) (c)

Fig. 5.16 The Kirigami patterns reproduced by using the baseline method (Xin et al. 2018) (a), ES-CGM (b), and AS-CGM (c).

5.6.1 Comparison of Exponent p in p-norm

In Section 3.5, we employed the *p-norm* in both AS-CGM and ES-CGM methods. In this section, we determine the preferable value for the exponent p by comparing the resulting condition number of the global stiffness matrix when using other normalisation methods. We randomly apply a single cut each time on the simulated material which initially has 6400 vertices. In our simulation, the degree of freedom (DOF) for a vertex is 3×3. Initially, the size of the global stiffness matrix is $1.92E+004^2$ and the condition number is within 10^1.

Table 5.4 illustrates the condition number of the global stiffness matrix and the corresponding computational time used by the CG solver after applying the cut. $L1$-Norm indicates that no matrix normalisation has been applied to the global stiffness matrix. In this case, the condition number can be up to 0.91×10^5 and it can spend up to 71.24 ms on the CG solver. We also implement the logarithm normalisation (log-norm) which can easily crash the program by resulting in an infinite condition number. Finally, we employ the square root normalisation ($p = 1/2$). As shown in the table, the condition number decreases substantially to only between 8.23×10^1 and 1.12×10^2. Besides, the computational time required by the CG solver is only 46.56 ms.

5.6.2 The Evaluation of the ES-CGM and AS-CGM

Recall the finding in Section 5.2.2, the size of the global stiffness matrix largely impacts the time efficiency of a cutting method. In this section, we compare the time efficiency and robustness of the proposed ES-CGM and AS-CGM methods with our baseline method (Xin et al. 2018) when optimising the global stiffness matrix which has various dimensions. Figure 5.16 shows the

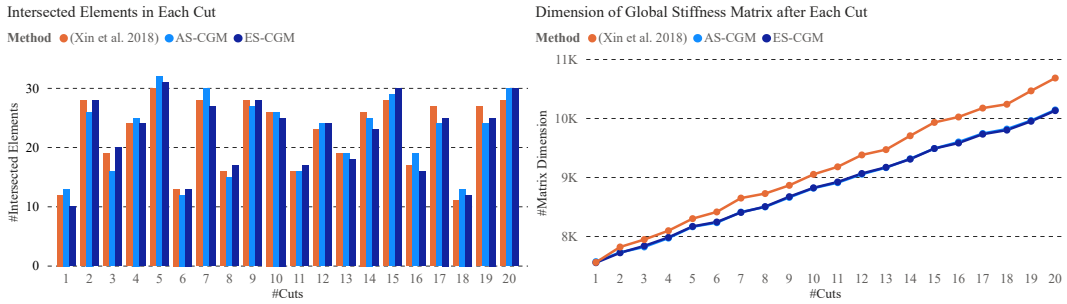


Fig. 5.17 The intersected elements of each cut shown in figure 5.16 and the dimension of the global stiffness matrix after applying each cut by implementing the baseline method (Xin et al. 2018), AS-CGM, and ES-CGM.

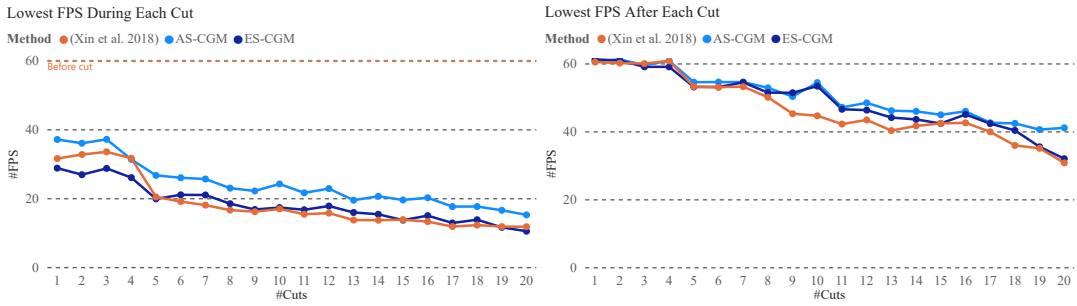


Fig. 5.18 The lowest FPS during and after reproducing each cut by using the baseline method (Xin et al. 2018), AS-CGM, and ES-CGM.

patterns generated by using each method and in each pattern there are 20 cuts. The surface model initially has 2,500 vertices and 4,802 triangular elements.

We compare the performance of the two proposed methods with the baseline method (Xin et al. 2018) after applying each cut. For a fair comparison, each cut is controlled in a similar size. The left-hand side of figure 5.17 illustrates the number of intersected elements of each cut, and the corresponding size of the global stiffness matrix is shown on the right. As shown in the figure, we test the performance of AS-CGM and ES-CGM on solving a wide range of matrices whose elements are between 7,557 and 10,687.

The AS-CGM and ES-CGM methods operate after the user defines the shape of the cut. Therefore, the lowest FPS during the cut evaluates how fast the cut is reproduced. Besides, the FPS after the cut reflects how fast the modified global stiffness matrix can be processed by the system. Figure 5.18 collects the lowest FPS during and after reproducing each cut. The dimension of the global stiffness matrix is up to 8,304 after applying a cut.

Generally speaking, the simulation which employs the AS-CGM method performs the best even though all three methods can keep the FPS higher than 20 before applying the 6^{th} cut. Furthermore, all three methods can achieve a cut at an interactive rate. However, most of the time, only the AS-CGM method can achieve a cut at the real-time rate on average. After the reproduction of each cut, all three methods keep the simulation runs with no less than 30 FPS and there is no significant difference among the three simulations. However, the AS-CGM

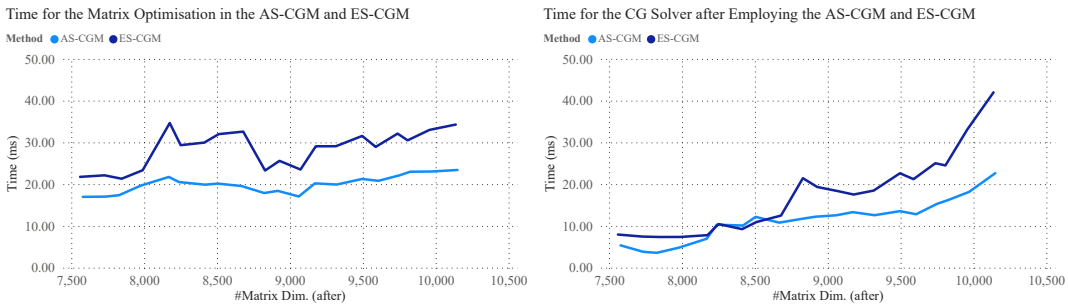


Fig. 5.19 The time needed for the matrix optimisation and the following CG solver in AS-CGM and ES-CGM methods.

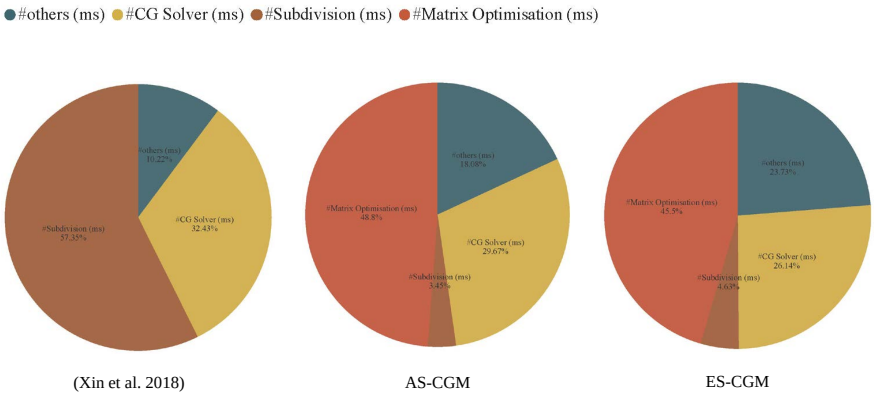


Fig. 5.20 The time distribution during the representation of a single cut in the AS-CGM, ES-CGM, and the baseline method (Xin et al. 2018).

method increases the FPS after the cuts from 30.90 FPS to 41.20 FPS compared with the baseline method (Xin et al. 2018).

Figure 5.19 provides the computation time spent on the matrix optimisation and the CG solver in the two methods, AS-CGM and ES-CGM. As shown in the figure, compared with the ES-CGM method, the AS-CGM method substantially speeds up the matrix optimisation process. Furthermore, for the CG solver, it is more effective to solve the global stiffness matrix which is optimised by the AS-CGM method than the ES-CGM.

Figure 5.20 illustrates the factors that make up the total computational cost and their percentages. In both AS-CGM and ES-CGM methods, an intersected element will be divided into three sub-elements. In this case, unlike those in the baseline method (Xin et al. 2018), the time for the subdivision is not significant. In contrast, most of the computational time is spent on the matrix optimisation process which includes matrix decomposition, matrix normalisation, and matrix re-composition. The other computational costs, such as on rendering, data updating, and matrix-vector calculations, are included in the "other" factor. The large percentage of the matrix optimisation process and the increasing percentage in the "other" factor indicate that speeding the matrix-matrix and matrix-vector calculations may further improve the time efficiency of AS-CGM and ES-CGM methods in the future.

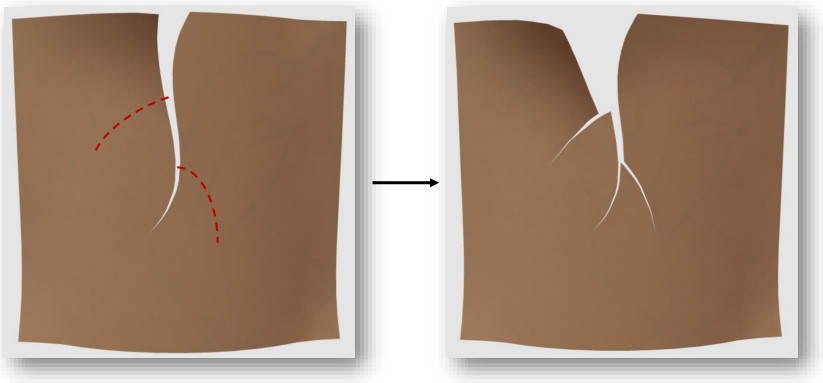


Fig. 5.21 Partial-cuts with the generated cut edges that support re-cuts.

Both AS-CGM and ES-CGM can significantly reduce the computational cost of the CG solver. However, it is worth noticing that the AS-CGM method which computes an approximate square root performs better than the ES-CGM method which calculates the exact square root. We suspect that learning an approximation square root normalisation of the covariance matrix of the global stiffness matrix is more robust than computing the accurate square root. However, it is still an open topic in future research.

5.7 Complex Cuts

In this section, we will show the ability of the proposed cutting method to handle complex cut situations in the 3D environment including partial-cuts, cross-cuts, cut out from an edge, and overlapping-cuts. Also, for practical cutting applications the capability to make a re-cut is a crucial factor to allow unlimited cuts in one simulation procedure. Therefore we integrate the re-cut behaviour in each cut situation to demonstrate the practical capability of the proposed cutting method. The model is constructed using the Co-rotational Finite Element Method (CFEM) with dynamic physically-based behaviours. In order to illustrate that our method can accurately represent the cuts regardless of the level of detail of the material model we apply the complex cut on the mesh which only has 393 original triangular elements.

5.7.1 Partial Cuts

Figure 5.21 illustrates the visual results of cutting a part of the entire mesh from an edge. It also indicates a capability to apply further cuts to the generated cut edges. On the left, the partial-cut results in two smooth cut edges on both sides of the cut plane. The ability to allow re-cuts requires the topologically and computationally independent cut edges generated from previous cuts. Unlike the so-called "mesh-free" cutting technologies, such as Virtual Node Algorithm (VNA) and Extended Finite Element Method (XFEM), which maintain the topological connections of the underlying computational mesh, we refine both the computational and rendering mesh around the cut plane to allow applying further cuts to the generated edges caused from the previous partial cuts. As shown in the left of the figure, after opening the incision

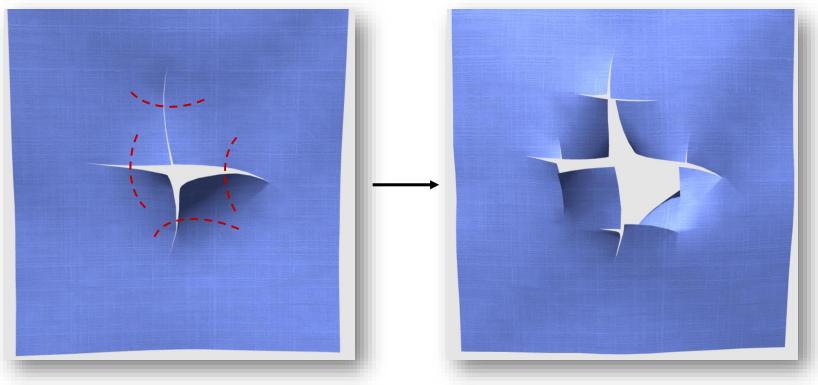


Fig. 5.22 Cross-cuts with the generated cut edges which are allowed to support further cuts.

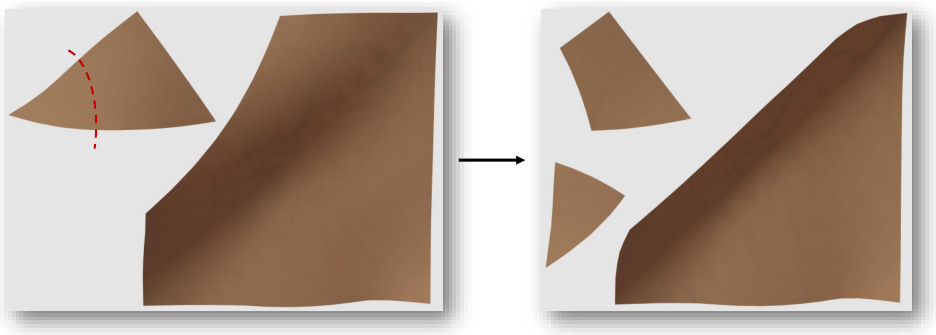


Fig. 5.23 Cut out partial domain with generating the independent sub-domain that allows further cuts.

caused by the previous cut, we give the second time cut (depicted in red dotted line) on each cut edge. On the right of the figure it shows that the two generated cut edges due to the previous partial-cut can support re-cuts anywhere. Also, each of the piecewise cut edges is allowed to support further cuts and accurate collision responds when colliding with other objects.

5.7.2 Cross Cuts

The ability to allow cross cuts reflects the flexibility of a cutting algorithm. It is particularly challenging to manage the refinement of the element at the cross point. Figure 5.22 demonstrates the visual results of cross-cuts using our novel cutting refinement scheme. On the left, the crossing incisions are reproduced after applying two crossed cuts. Also, the resulting edges of the cross incision are allowed to support re-cuts behaviour. As shown on the right, more complicated cross cuts are achieved when applying re-cuts which cross each generated cut edges from the previous cuts. Such technologies will be adequate to simulate practical applications such as kirigami.

5.7.3 Cut Out an Independent Sub-Domain

Our method ensures that any sub-domain cut out entirely is topologically and behaviourally independent. Figure 5.23 shows the visual results when using our cutting method. On the left a

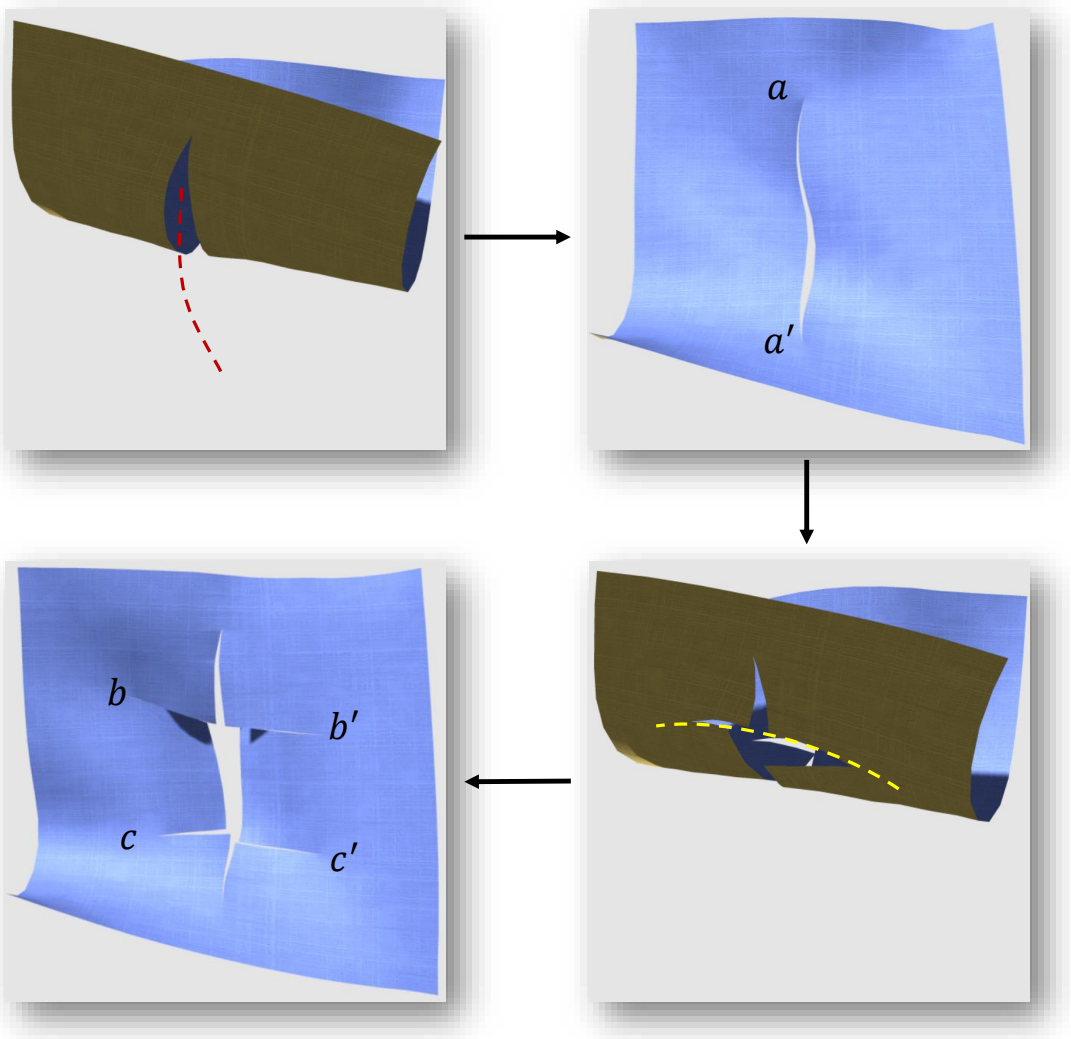


Fig. 5.24 A single incision caused by the overlapping cuts and the re-cuts applied on its cut edges. The incision ab results from the first overlapping cut (the red dotted line), and the incisions bb', cc. are generated after applying the second overlapping cut (depicted by the yellow dotted line).

corner is cut out from the initial mesh. Afterwards we apply the second cut on the sub-domain (depicted in red dotted line). As shown in the right, the re-cut behaviour is supported in the sub-domain. Also, each part can be treated as an independent mesh that allows further cutting operations.

5.7.4 Overlapping Cuts

Single incision and re-cuts. Figure 5.24 illustrates a single incision caused by the overlapping cuts applied on the curved surface model. As shown in the figure, the non-plane model is cut by the first cut plane (depicted by the red dotted line), one of whose endpoint terminates inside the model. It results in the single incision aa'. Afterwards, the second cut (depicted by the yellow dotted line) is applied on the generated cut edges of the curved model. This results in two incisions (bb', cc') and illustrates one situation, in which the overlapping cuts can cause single incision. Also, the generated cut edges can support further cuts respectively.

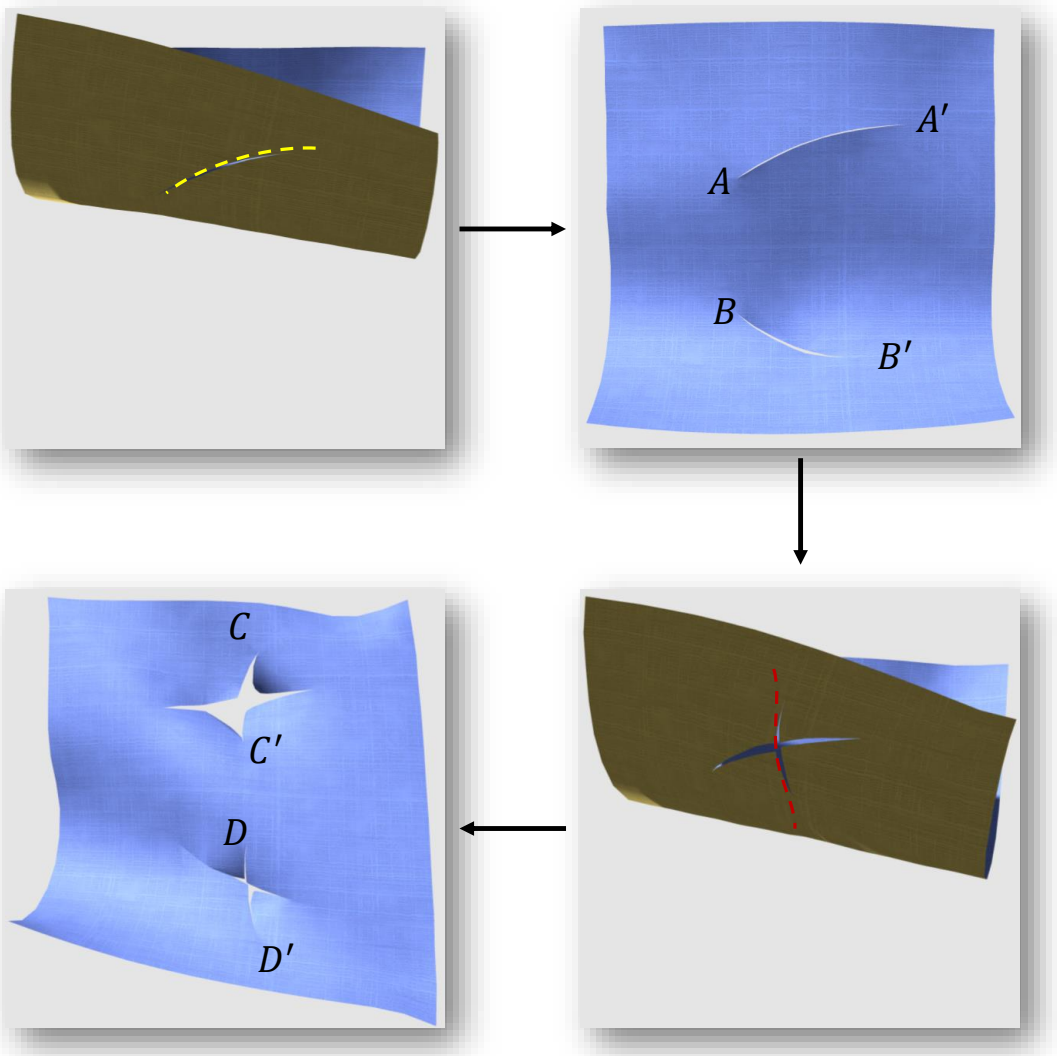


Fig. 5.25 The dual incisions AA' and BB' generated due to the first cut. The second overlapping cuts results in the dual incisions CC' and DD'.

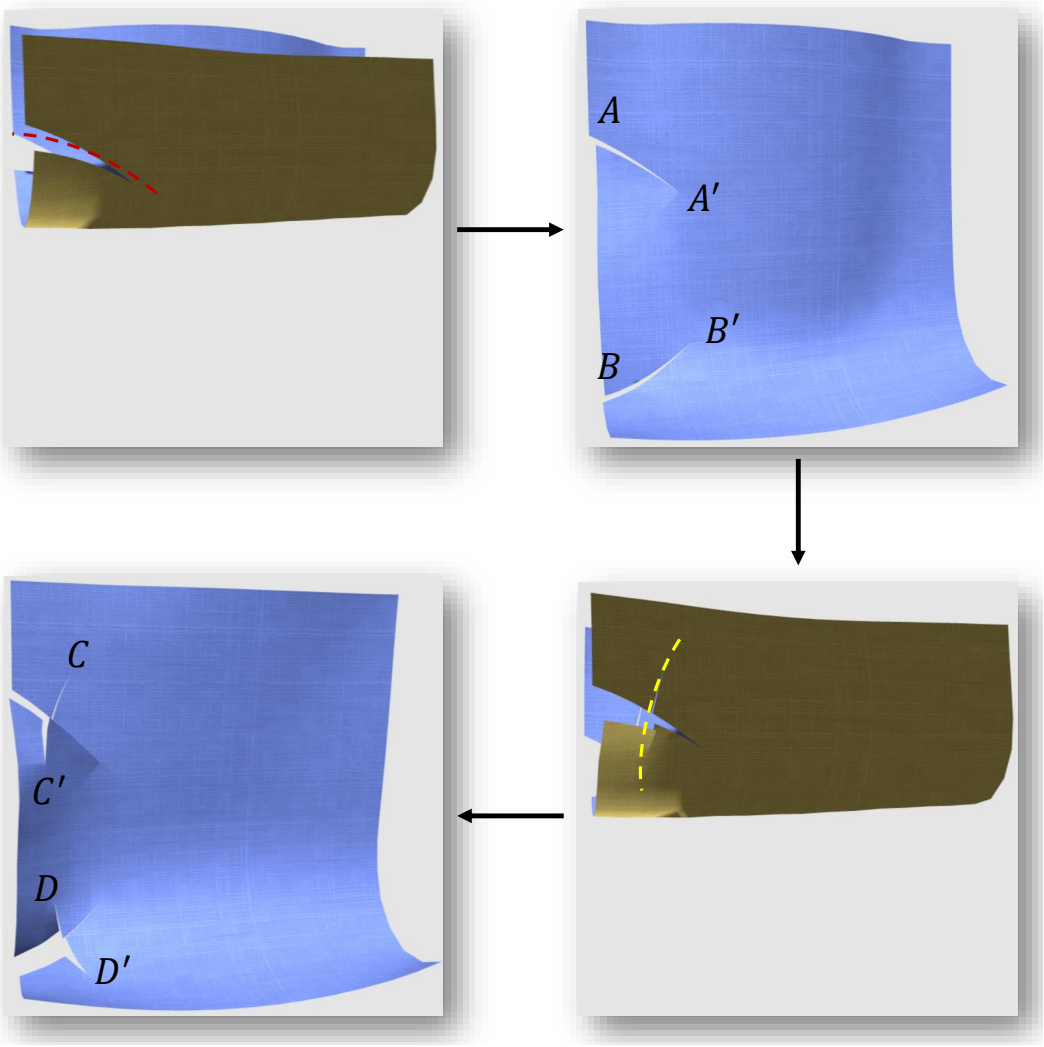


Fig. 5.26 The dual partial incisions AA' and BB' generated due to the first cut that has one endpoint terminated inside the mesh. The second overlapping cuts results in the dual incisions CC' and DD'.

Dual incisions and re-cuts. Figure 5.25 shows one overlapping cut situation in which multiple cuts are reproduced. As shown in the figure, the curved surface model is first cut by the yellow dotted line, causing two cuts AA' and BB'. Furthermore, the second overlapping cut (depicted by the red dotted line) is applied on the cut edges of the two partial incisions. This results in another two partial cuts (also cross-cuts) CC' and DD'. Figure 5.26 describes another overlapping cut situation that also results in multiple cut incisions. Unlike the situation described previously, the first cut plane only has one endpoint terminating within the mesh. Therefore, it results in two partial cuts AA' and BB'. Afterwards, the second cut (depicted by the yellow dotted line) is applied on the edges of the two partial cuts and the incisions CC' and DD' are generated.

5.8 Conclusion

In this chapter we evaluated the Adapted Constrained Delaunay Method (ACDM) and two matrix normalisation methods, i.e. AS-CGM and ES-CGM in terms of the crucial factors for reproducing complex pre-defined cuts in real-time, i.e. the lowest FPS during the cut should not be lower than 30 FPS. First, we compared the cut path representation and the time efficiency with the state-of-the-art method (Manteaux et al. 2015) which can interactively achieve detailed pre-defined cuts on deformable surface models. Afterwards, we illustrated how ill-shaped triangular elements influence the efficiency of the simulation and identify the preferable criteria for both average *Aspect Ratio* and *Area Ratio* when operating the ACDM subdivision. Furthermore, we evaluated the performance of the AS-CGM and ES-CGM methods when optimising the global stiffness matrix with elements ranging from 7,557 to 10,687. To summarise, the results presented in this chapter indicate that:

- In comparison with the state-of-the-art method (Manteaux et al. 2015), our method achieved a competitive visual result of detailed pre-defined cuts while gaining a massive improvement on the lowest FPS during and after the cuts. Specifically, for a single cut, intersecting cuts and a punch, the improvement on the lowest FPS during the cut is 39.94 FPS, 43.44 FPS, and 46.49 FPS, respectively.

- The ACDM subdivision can effectively maintain a high quality of a mesh which has been cut by decreasing the average *Aspect Ratio* and the average *Area Ratio*. Specifically, this subdivision can achieve a cut which includes up to 1,119 intersected elements in real-time by stably maintaining the average *Aspect Ratio* of the elements in the mesh to be less than 3 and *Area Ratio* staying at 3%.

- The ES-CGM and AS-CGM methods can significantly decrease the condition number of the global stiffness matrix after applying the cuts. Compared with the baseline method (Xin et al. 2018), the ES-CGM and AS-CGM methods increase approximately 5 FPS and 10 FPS during and after the cut, respectively.

- Finally we showed the capability of reproducing overlapped complex cuts in 3D environments. We also illustrate the practical applicability of the proposed cutting framework by supporting re-cuts in various cutting scenarios.

Chapter 6. Conclusion

Representing detailed cuts on dynamic deformable materials in real-time is still challenging. This is primarily due to the fact that the cutting method needs to handle the complicated scenarios in a 3D environment while achieving a suitable representation in limited time that is available for real-time simulations. Moreover the dynamic behaviour of the material which has been cut need to be accurate and stable in the resulting deformations in order to reproduce a realistic cutting simulation.

In this thesis the primary goal is to provide a general solution for reproducing accurate cuts on dynamic deformable surface materials in real-time. Both visual results and computational cost from our experimental work indicate that the method proposed in this thesis is capable of being implemented in the real-time cutting simulations of physically-based dynamic surface material models. In this final chapter we first summarise the work that has been presented, implemented and evaluated in this thesis. Afterwards we emphasis the main contributions of this work and finally discuss how it can be extended or developed further.

6.1 Thesis Summary

This thesis has proposed a real-time mesh-based cutting technique which aims to accurately represent complex cuts that are applied to FEM-based deformable thin materials in a 3D environment. This technique is implemented into a dynamic physically-based deformable surface model which is simulated in real-time. The proposed cutting frame further provides a numerical solution for optimising the modified global stiffness matrix, thereby ensuring accurate physical behaviour of the material model. This thesis then discusses the visual results and the computational costs of the proposed method. Such results indicate that the method is capable of supporting the accurate representation of cuts in physically-based real-time simulations.

6.2 Main Contributions

This thesis has researched the mesh-based cutting techniques in physically-based dynamic deformable FEM modelling. Three main contributions have been achieved in this thesis. These contributions are as follows:

- A real-time method to represent the modification of deformable material models due to cuts. This method starts after a user initially defines the shape of the cut by clicking a mouse four times to specify the position of the cut. It utilises an efficient refinement scheme which only increases the number of triangular elements around the cuts in order to accurately represent complex cuts. The proposed method behaves stably in a wide range of cutting scenarios with linear computation complexity , i.e. $O(N)$.

- An optimisation scheme ensuring the physical accuracy of the dynamic deformations of the material models after the cutting has occurred. This scheme first employs an efficient Delaunay triangulation generation algorithm. The Bowyer-Watson algorithm is then used to improve the modified mesh without increasing the number of new nodes and elements. Afterwards a further localised optimisation is applied in order to continuously improve the quality of the modified mesh due to the cuts. The data which has been listed in chapter 5 has shown that this optimisation scheme substantially mitigates the effects of the ill-shaped triangular elements by reducing their *Aspect Ratios* and *Area Ratios*.

- A general solution to handle the complex cuts that result from the complicated cutting scenarios. Such cuts include a single smooth cut which can be arbitrarily-shaped, re-cuts on the place which has been cut previously, cross cuts, cuts through a part of the material model, cut out of a part of the material model and overlapping cuts in which the cut path intersects with the material model in multiple places. Such a solution provides the opportunity to simulate a high-performance physically-based dynamic deformable surface material which supports re-cuts by generating physically accurate cut edges rather than a purely graphical representation of the cut.

6.3 Future Work

While this thesis provides a novel cutting simulation that can be implemented in physically-based FEM modelling there are several directions that can be followed to extend the work that is presented in this thesis.

6.3.1 Cutting on the Composite Material Models

This thesis presented a novel method that is capable of handling complex cuts which are applied to deformable surface material models. However in the real world most of the materials involved such as skin consist of multiple layers. Figure 6.1 shows the skin layers which includes the epidermis, dermis and subcutaneous fat. Implementing our method into such layered materials is challenging as each layer need to be represented appropriately. Such layers have different material structures and properties. When applying a cut such different material layers should have different reactions. This aspect is significant when it comes to representing cuts on a skin-like material model and simulating skin slide.

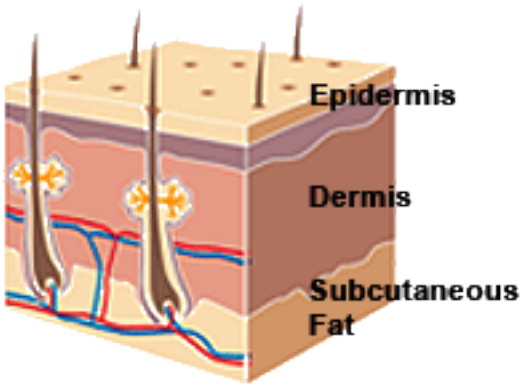


Fig. 6.1 Picture of the skin. The original resource can be found at https://www.aad.org/public/kids/skin/the-layers-of-your-skin.

6.3.2 Real-time Collision Response

This thesis provides a general cutting solution which can be achieved in real-time. In such a situation the extra computational cost required limits the solution time available after applying the cuts. Therefore the collision response of the material model when it is intersected with the cut path is ignored. However a realistic and fast collision response from the material model which has been cut is required in a cutting simulation where the cutting process and user experience is the priority rather than the representation of the cuts. Moreover such user experience can be improved by employing a progressive cutting process in which the cut occurs while the users are actually cutting the material models.

6.3.3 Non-linear Elements

This thesis employs linear triangular mesh elements which have three levels of freedom. However higher-ordered triangular elements can be achieved by adding nodes at the edges of the elements (also referred to as non-linear elements). In this manner, higher levels of degree can represent the more realistic deformations of the material models. Consequently longer computational times are required to solve such deformations. Moreover the refinement of such non-linear elements is more complicated and time-consuming than with linear elements. Achieving a real-time cutting on such kind of elements is still an open research topic for the future.

References

Anderson, T. L. (2017), *Fracture mechanics: fundamentals and applications*, CRC press.

Babuška, I. & Aziz, A. K. (1976), 'On the angle condition in the finite element method', *SIAM Journal on Numerical Analysis* **13**(2), 214–226.

Baraff, D. & Witkin, A. (1998), Large steps in cloth simulation, *in* 'Proceedings of the 25th annual conference on Computer graphics and interactive techniques', ACM, pp. 43–54.

Beach, R. C. (1991), *An Introduction to the Curves and Surfaces of Computer-Aided Design*, Van Nostrand Reinhold Co.

Belanger, J., Venne, P. & Paquin, J. (2010), 'The what, where and why of real-time simulation', *Planet RT* **1**(1), 25–29.

Belytschko, T. & Black, T. (1999), 'Elastic crack growth in finite elements with minimal remeshing', *International journal for numerical methods in engineering* **45**(5), 601–620.

Bielser, D., Glardon, P., Teschner, M. & Gross, M. (2004), 'A state machine for real-time cutting of tetrahedral meshes', *Graphical Models* **66**(6), 398–417.

Bielser, D., Maiwald, V. A. & Gross, M. H. (1999), Interactive cuts through 3-dimensional soft tissue, *in* 'Computer Graphics Forum', Vol. 18, Wiley Online Library, pp. 31–38.

Bowyer, A. (1981), 'Computing dirichlet tessellations', *The computer journal* **24**(2), 162–166.

Breen, D. E., House, D. H. & Wozny, M. J. (1994), Predicting the drape of woven cloth using interacting particles, *in* 'Proceedings of the 21st annual conference on Computer graphics and interactive techniques', ACM, pp. 365–372.

Bruyns, C. D., Senger, S., Menon, A., Montgomery, K., Wildermuth, S. & Boyle, R. (2002), 'A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools', *Computer Animation and Virtual Worlds* **13**(1), 21–42.

Busaryev, O., Dey, T. K. & Wang, H. (2013), 'Adaptive fracture simulation of multi-layered thin plates', *ACM Transactions on Graphics (TOG)* **32**(4), 52.

Chadwick, P. (2012), *Continuum mechanics: concise theory and problems*, Courier Corporation.

Christian, B., Haynor, D. R. & Hellier, P. (2004), 'Proceedings of medical image computing and computer-assisted intervention-miccai 2004'.

Cotin, S., Delingette, H. & Ayache, N. (2000), 'A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation', *The Visual Computer* **16**(8), 437–452.

Courtecuisse, H., Allard, J., Kerfriden, P., Bordas, S. P., Cotin, S. & Duriez, C. (2014), 'Real-time simulation of contact and cutting of heterogeneous soft-tissues', *Medical image analysis* **18**(2), 394–410.

Cueto, E. & Chinesta, F. (2014), 'Real time simulation for computational surgery: a review', *Advanced Modeling and Simulation in Engineering Sciences* **1**(1), 11.

de L'isle, E. B. & George, P. L. (1995), Optimization of tetrahedral meshes, *in* 'Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations', Springer, pp. 97–127.

Delaunay, B. (1934), 'Sur la sphere vide', *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* **7**(793-800), 1–2.

Delingette, H. & Ayache, N. (2004), 'Soft tissue modeling for surgery simulation', *Handbook of Numerical Analysis* **12**, 453–550.

Dey, T. K., Bajaj, C. L. & Sugihara, K. (1992), 'On good triangulations in three dimensions', *International Journal of Computational Geometry & Applications* **2**(01), 75–95.

Felippa, C. A. (2000), A systematic approach to the element-independent corotational dynamics of finite elements, Technical report, Technical Report CU-CAS-00-03, Center for Aerospace Structures.

Flynn, C. & McCormack, B. A. (2010), 'Simulating the wrinkling and aging of skin with a multi-layer finite element model', *Journal of biomechanics* **43**(3), 442–448.

Freitag, L. A. & Ollivier-Gooch, C. (2000), 'A cost/benefit analysis of simplicial mesh improvement techniques as measured by solution efficiency', *International Journal of Computational Geometry & Applications* **10**(04), 361–382.

Fung, Y.-c. (1977), 'A first course in continuum mechanics', *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1977. 351 p.* .

Gillio, R. G. (1999), 'Virtual surgery system'. US Patent 5,882,206.

Golub, G. H. & Van Loan, C. F. (2012), *Matrix computations*, Vol. 3, JHU Press.

Gregory, J. (2017), *Game engine architecture*, AK Peters/CRC Press.

Guennebaud, G., Jacob, B. et al. (2010), 'Eigen', *URl: http://eigen. tuxfamily. org* .

Higham, N. J. (2008), *Functions of matrices: theory and computation*, Vol. 104, Siam.

Ho-Le, K. (1988), 'Finite element mesh generation methods: a review and classification', *Computer-aided design* **20**(1), 27–38.

Hughes, T. J. (2012), *The finite element method: linear static and dynamic finite element analysis*, Courier Corporation.

James, D. L. & Pai, D. K. (1999), Artdefo: accurate real time deformable objects, *in* 'Proceedings of the 26th annual conference on Computer graphics and interactive techniques', ACM Press/Addison-Wesley Publishing Co., pp. 65–72.

Joy, K. I. (2000), 'Cubic bezier curves', *Visualization and Graphics Research Group, University of California, Davis* .

Kaufmann, P., Martin, S., Botsch, M., Grinspun, E. & Gross, M. (2009), Enrichment textures for detailed cutting of shells, *in* 'ACM Transactions on Graphics (TOG)', Vol. 28, ACM, p. 50.

Koschier, D., Bender, J. & Thuerey, N. (2017), 'Robust extended finite elements for complex cutting of deformables', *ACM Transactions on Graphics (TOG)* **36**(4), 55.

Koschier, D., Lipponer, S. & Bender, J. (2014), Adaptive tetrahedral meshes for brittle fracture simulation, *in* 'Proceedings of the ACM SIGGRAPH/eurographics symposium on computer animation', Eurographics Association, pp. 57–66.

Lawson, C. L. (1972), 'Generation of a triangular grid with applications to contour plotting', *Jet Propul. Lab. Techn. Memo* **299**, 2.

Lawson, C. L. (1977), Software for c 1 surface interpolation, *in* 'Mathematical software', Elsevier, pp. 161–194.

Lee, D.-T. & Lin, A. K. (1986), 'Generalized delaunay triangulation for planar graphs', *Discrete & Computational Geometry* **1**(3), 201–217.

Lim, Y.-J., Jin, W. & De, S. (2007), 'On some recent advances in multimodal surgery simulation: A hybrid approach to surgical cutting and the use of video images for enhanced realism', *Presence: Teleoperators and Virtual Environments* **16**(6), 563–583.

Lindblad, A. & Turkiyyah, G. (2007), A physically-based framework for real-time haptic cutting and interaction with 3d continuum models, *in* 'Proceedings of the 2007 ACM symposium on Solid and physical modeling', ACM, pp. 421–429.

Liu, I.-S. (2013), *Continuum mechanics*, Springer Science & Business Media.

Manteaux, P.-L., Sun, W.-L., Faure, F., Cani, M.-P. & O'Brien, J. F. (2015), Interactive detailed cutting of thin sheets, *in* 'Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games', ACM, pp. 125–132.

Manteaux, P.-L., Wojtan, C., Narain, R., Redon, S., Faure, F. & Cani, M.-P. (2017), Adaptive physically based models in computer graphics, *in* 'Computer Graphics Forum', Vol. 36, Wiley Online Library, pp. 312–337.

McCloy, R. & Stone, R. (2001), 'Virtual reality in surgery', *Bmj* **323**(7318), 912–915.

Meier, U., López, O., Monserrat, C., Juan, M. C. & Alcaniz, M. (2005), 'Real-time deformable models for surgery simulation: a survey', *Computer methods and programs in biomedicine* **77**(3), 183–197.

Molino, N., Bao, Z. & Fedkiw, R. (2004), A virtual node algorithm for changing mesh topology during simulation, *in* 'ACM Transactions on Graphics (TOG)', Vol. 23, ACM, pp. 385–392.

Müller, M., Dorsey, J., McMillan, L., Jagnow, R. & Cutler, B. (2002), Stable real-time deformations, *in* 'Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation', ACM, pp. 49–54.

Müller, M. & Gross, M. (2004), Interactive virtual materials, *in* 'Proceedings of Graphics Interface 2004', Canadian Human-Computer Communications Society, pp. 239–246.

Nam, N. M., Van Kiem, H. & Nam, N. V. (2009), A fast algorithm for constructing constrained delaunay triangulation, *in* '2009 IEEE-RIVF International Conference on Computing and Communication Technologies', IEEE, pp. 1–4.

Narain, R., Samii, A. & O'Brien, J. F. (2012), 'Adaptive anisotropic remeshing for cloth simulation', *ACM transactions on graphics (TOG)* **31**(6), 152.

Nealen, A., Müller, M., Keiser, R., Boxerman, E. & Carlson, M. (2006), Physically based deformable models in computer graphics, *in* 'Computer graphics forum', Vol. 25, Wiley Online Library, pp. 809–836.

Nienhuys, H.-W. & van der Stappen, A. F. (2000), Combining finite element deformation with cutting for surgery simulations, *in* 'EuroGraphics short presentations', pp. 43–52.

Nienhuys, H.-W. & van der Stappen, A. F. (2001), A surgery simulation supporting cuts and finite element deformation, *in* 'International conference on medical image computing and computer-assisted intervention', Springer, pp. 145–152.

Nienhuys, H.-W. & van der Stappen, A. F. (2004), A delaunay approach to interactive cutting in triangulated surfaces, *in* 'Algorithmic Foundations of Robotics V', Springer, pp. 113–129.

O'brien, J. F. & Hodgins, J. K. (1999), Graphical modeling and animation of brittle fracture, *in* 'Proceedings of the 26th annual conference on Computer graphics and interactive techniques', ACM Press/Addison-Wesley Publishing Co., pp. 137–146.

Pfaff, T., Narain, R., De Joya, J. M. & O'Brien, J. F. (2014), 'Adaptive tearing and cracking of thin sheets', *ACM Transactions on Graphics (TOG)* **33**(4), 110.

Saito, J. & Yuen, S. (2017), Efficient and robust skin slide simulation, *in* 'Proceedings of the ACM SIGGRAPH Digital Production Symposium', ACM, p. 10.

Seiler, M., Steinemann, D., Spillmann, J. & Harders, M. (2011), 'Robust interactive cutting based on an adaptive octree simulation mesh', *The Visual Computer* **27**(6-8), 519–529.

Sela, G., Subag, J., Lindblad, A., Albocher, D., Schein, S. & Elber, G. (2007), 'Real-time haptic incision simulation using fem-based discontinuous free-form deformation', *Computer-Aided Design* **39**(8), 685–693.

Serby, D., Harders, M. & Székely, G. (2001), A new approach to cutting into finite element models, *in* 'International Conference on Medical Image Computing and Computer-Assisted Intervention', Springer, pp. 425–433.

Shewchuk, J. (2002), 'What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint)', *University of California at Berkeley* **73**, 137.

Shimada, K. & Gossard, D. C. (1995), Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing, *in* 'Proceedings of the third ACM symposium on Solid modeling and applications', ACM, pp. 409–419.

Sibson, R. (1978), 'Locally equiangular triangulations', *The computer journal* **21**(3), 243–245.

Sloan, S. (1987), 'A fast algorithm for constructing delaunay triangulations in the plane', *Advances in Engineering Software (1978)* **9**(1), 34–55.

Sloan, S. & Houlsby, G. (1984), 'An implementation of watson's algorithm for computing 2-dimensional delaunay triangulations', *Advances in Engineering Software (1978)* **6**(4), 192–197.

Sloan, S. W. (1993), 'A fast algorithm for generating constrained delaunay triangulations', *Computers & Structures* **47**(3), 441–450.

Sokolnikoff, I. S., Specht, R. D. et al. (1956), *Mathematical theory of elasticity*, Vol. 83, McGraw-Hill New York.

Solin, P., Segeth, K. & Dolezel, I. (2003), *Higher-order finite element methods*, Chapman and Hall/CRC.

Steinemann, D., Otaduy, M. A. & Gross, M. (2006), Fast arbitrary splitting of deforming objects, *in* 'Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation', Eurographics Association, pp. 63–72.

Tanaka, H. T., Tsujino, Y., Kamada, T. & Viet, H. Q. H. (2006), Bisection refinement-based real-time adaptive mesh model for deformation and cutting of soft objects, *in* 'Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on', IEEE, pp. 1–8.

Terzopoulos, D. & Fleischer, K. (1988*a*), 'Deformable models', *The visual computer* **4**(6), 306–331.

Terzopoulos, D. & Fleischer, K. (1988*b*), Modeling inelastic deformation: viscolelasticity, plasticity, fracture, *in* 'ACM Siggraph Computer Graphics', Vol. 22, ACM, pp. 269–278.

Terzopoulos, D., Platt, J., Barr, A. & Fleischer, K. (1987), 'Elastically deformable models', *ACM Siggraph Computer Graphics* **21**(4), 205–214.

Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.-P., Faure, F., Magnenat-Thalmann, N., Strasser, W. et al. (2005), Collision detection for deformable objects, *in* 'Computer graphics forum', Vol. 24, Wiley Online Library, pp. 61–81.

Turkiyyah, G. M., Karam, W. B., Ajami, Z. & Nasri, A. (2011), 'Mesh cutting during real-time physical simulation', *Computer-Aided Design* **43**(7), 809–819.

Wang, Y., Jiang, C., Schroeder, C. & Teran, J. (2014), An adaptive virtual node algorithm with robust mesh cutting, *in* 'Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation', Eurographics Association, pp. 77–85.

Warburton, M. & Maddock, S. (2015), 'Physically-based forehead animation including wrinkles', *Computer Animation and Virtual Worlds* **26**(1), 55–68.

Watson, D. (1981), 'Computing the a-dimensional dclaunay triangulation with application to voronoi polytopes', *The Computer Journal* **24**.

Wicke, M., Ritchie, D., Klingner, B. M., Burke, S., Shewchuk, J. R. & O'Brien, J. F. (2010), Dynamic local remeshing for elastoplastic simulation, *in* 'ACM Transactions on graphics (TOG)', Vol. 29, ACM, p. 49.

Wicke, M., Steinemann, D. & Gross, M. (2005), Efficient animation of point-sampled thin shells, *in* 'Computer Graphics Forum', Vol. 24, Wiley Online Library, pp. 667–676.

Wu, J., Westermann, R. & Dick, C. (2015), A survey of physically based simulation of cuts in deformable bodies, *in* 'Computer Graphics Forum', Vol. 34, Wiley Online Library, pp. 161–187.

Xin, T., Marris, P., Mihut, A., Ushaw, G. & Morgan, G. (2018), Accurate real-time complex cutting in finite element modeling, *in* '13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018)', Vol. 1: GRAPP, pp. 183–190.

Zhang, J., Zhong, Y. & Gu, C. (2017), 'Deformable models for surgical simulation: a survey', *IEEE reviews in biomedical engineering* **11**, 143–164.