Thin deformable tissues modeling for simulation of arthroscopic knee surgery

Weng, Bin

2018

Weng, B. (2018). Thin deformable tissues modeling for simulation of arthroscopic knee surgery. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/89074

https://doi.org/10.32657/10220/46045

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

THIN DEFORMABLE TISSUES MODELING FOR SIMULATION OF ARTHROSCOPIC KNEE SURGERY

BIN WENG

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2018

THIN DEFORMABLE TISSUES MODELING FOR SIMULATION OF ARTHROSCOPIC KNEE SURGERY

BIN WENG

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University in partial fulfilment of the requirement for the degree of Doctor of Philosophy

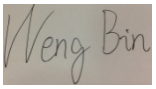
2018

Statement of Originality

I hereby certify that the content of this thesis is the result of work done by myself and has not been submitted for a higher degree to any other University or Institution.

26-Jan-2018

Date



Signature

Acknowledgements

First of all, I would like to thank my supervisor Associate Professor Alexei Sourin for his invaluable instructions. I am grateful for his help throughout my graduate study in Nanyang Technological University. He helped me to understand basic ideas, and to correct theoretical findings. Without his effects, ideas and comments, the project could not reach this point.

I would also like to thank Nanyang Technological University and School of Computer Science and Engineering for providing me the opportunity and financial support to further my study and research. Special thanks go to Fraunhofer Singapore for providing me the first class equipment and technical supports.

Last but not least, I would like to thank my friends, Jian Cui, Xingzi Zhang, and Dr. Shahzad Rasool, for their precious support. I would also like to thank Dr. Fareed H. Y. Kagda for his valuable input and cooperation. Furthermore, I wish to thank my parents, their selfless concern and encouragement are the best support during my postgraduate study.

Abstract

Minimally invasive surgery is common in present hospitals since it has many advantages over traditional open surgery. However, the skills which it requires are very different from those developed in doing traditional surgery, and they have to be learnt in a different way. A survey of the current training methods reveals that using virtual reality simulators is a promising direction for training minimally invasive surgery, and it requires the developers to improve accuracy and efficiency of human tissue simulations. A further survey of existing deformable models for real time surgical-training simulations leads to the conclusion that there is room for research for developing new thin tissue models in which one of the dimensions is significantly smaller than the other two.

A new generic model is proposed for deformation and cutting of thin deformable tissues in real time. The tissues may develop wrinkles when pressed with virtual probes, as well as they can be trimmed and cut with different virtual surgical instruments using desktop haptic devices.

The model is then applied to three tissues for simulating virtual arthroscopic surgery: meniscus, ligament and cartilage. The main attention was paid to the meniscus to be realistically deformed while developing wrinkles when tested with the virtual probe, as well as to be cut and trimmed at its edge with various virtual instruments. Ligaments were simulated for displaying their realistic deformation when examined with the surgical hook and cartilage was

mostly used to modeling its geometry and elastic properties when examined with haptic devices. A side-by-side comparison with actual surgical videos verifies the simulation accuracy. To demonstrate its flexibility, the model was also applied to other objects such as a piece of meat and a thin paper sheet.

To validate the obtained results, a virtual reality knee arthroscopic prototype simulator is developed. The proposed models are combined with a scene graph structure and implemented using SOFA. The performances for the implementation of different settings of collision meshes are compared and discussed. The usability and feasibility of the proposed methods are further verified by the real time simulations of various surgical procedures, which include meniscus examination, punching out torn tissues, removing tissues pieces by pieces, and contouring the edge of meniscus.

Contents

List of Figures

Glossary of Terms

Term	Description	Citation
2D	Two dimensional	
3D	Three dimensional	
ACL	Anterior cruciate ligament	[1]
CPU	Central Processing Unit	
DOF	Degree of freedom	
FEM	Finite element method	[2]
FPS	Frames per second	
GPU	Graphics Processing Unit	
GHz	Gigahertz	
GB	Gigabyte	
in vivo	Performed or taking place in a living organism	[3]
KA	Knee arthroscopy	[1]
LCL	Lateral collateral ligament	[1]
MB	Megabyte	
MCL	Medial collateral ligament	[1]
MIP	Minimally invasive procedure	[1]

MIS	Minimally invasive surgery	[1]
MOR	Model order reduction	[4]
MRI	Magnetic resonance imaging	
PCL	Posterior cruciate ligament	[1]
Poisson’s ratio	The negative of the ratio of transverse strain to axial strain	
RAM	Random-access memory	
SOFA	Simulation Open Framework Architecture	[5]
VR	Virtual reality	
Young’s modulus	The ratio of stress to strain	

Chapter 1 Introduction

This chapter is an introduction to this thesis. In Section 1.1, the motivation of this thesis is given. Section 1.2 briefly describes the objective of the project. In Section 1.3, the proposed new methods are briefly explained. Section 1.4 states the organization of this thesis.

1.1 Motivation

Minimally invasive procedure (MIP) or minimally invasive surgery (MIS) is a category of surgical procedures in which the long thin camera and instruments are inserted into patients' body via a key-hole incision (about 1/8 inch) on the skin instead of cutting and opening the skin in the traditional surgery. During the general process, surgeons watch the surgical scene recorded by the camera via a video monitor, and at the same time move the inserted tool and camera to do the necessary surgical operation scene within the body. It is a very popular surgical procedure which has replaced many open surgeries in modern hospitals.

MIP requires skills different from those in traditional surgeries. Even a surgeon experienced in open surgeries has to spend extensive time learning these special skills, not to mention the medical school students. The trainees need to stand behind the surgeon and watch real surgery for a long time (at least one year), and practice on cadavers, animals or some artificial mock-ups to acquire the necessary skills before they are allowed to do the surgery on real patients. However, traditional learning method is very inefficient, thus putting

the patients on high risks. Therefore, a surgical simulator can really help. The indirect way of watching the monitor while operating on patients in MIP is very suitable for computer simulation, because the patients can be replaced by some plastic models and surgical scene on the monitor can be generated by computers. The purpose of surgical simulators is not to replace the traditional training methods, but to supplement them.

Since 1990, virtual reality (VR) surgical training simulation has constantly attracted many research efforts. Although many research-based and commercial VR surgical simulators are available, it is still a hot research topic. A survey on the existing arthroscopy VR simulators reveals that there are still gaps between simulations and real surgery. Current difficulty mainly comes from the modeling of human organs, which is the core of the software system for surgery-training simulators. The existing models of deformable objects (e.g. organs and tissues) simulation in computer graphics are reviewed. The thesis looks at simulation of thin deformable tissues in arthroscopic knee surgery, which include menisci, ligaments and cartilages—these are the tissues in which one of the dimensions is significantly smaller than the other two. The key research issues of the real-time simulation of local detailed deformation and efficient cutting of the tissues are motivated. For the deformation simulation, a hypothesis is proposed, to embed points into coarse simulation mesh to allow for the detailed local deformation of thin deformable objects. For the cutting simulation, a hypothesis is proposed to add the visual-collision binding to allow

for consistent and interactive cutting of the thin deformable objects with different resolutions.

1.2 Objective

The purpose of this thesis is to analyze the bottleneck of the-state-of-the-art VR surgical simulators for knee arthroscopy, especially with respect to meniscus modeling, and to develop new methods to improve the simulation.

The first objective is to propose a novel method to enhance the existing simulation with local detailed deformation. The meniscus simulated by this method is able to produce wrinkles like local deformation on its thinner edge, which is similar to its behavior in the real surgery.

The second objective is to put forward a new method to integrate high-resolution visual models and low-resolution collision models into cutting simulations. This method can simulate the operations of punching out the torn tissues and contouring the edge of meniscus. It also can be used for modeling deformations of ligaments and physical properties of cartilages.

The third objective is to demonstrate through a series of experiments on a prototype knee arthroscopy VR simulator that the proposed methods could improve the current VR surgical simulation. This simulator could support the simulation of the examination and lesion handling of the meniscus.

1.3 Approaches and research novelties

To achieve the above objectives, several new approaches are proposed. A brief introduction to the proposed approaches is given here while more details can be found in the rest of this thesis.

1. Embedding points into the coarse simulation mesh. Coarse simulation mesh cannot generate detailed local deformation, but it still contains the clue. To utilize this clue, the points have to be embedded into the coarse simulation mesh to monitor the state of the deformation. According to the complicated interactions of tool-tissue and tissue-tissue in the simulation, reference configuration for the local deformation has to be dynamically defined. It is anticipated that based on this configuration, local wrinkle like deformation can be then generated and controlled using functions.
2. Binding visual mesh to collision mesh. When the resolutions of visual and collision meshes are different, it is difficult to do the cutting consistently. It is expected that splitting into preprocessing and simulation phases may solve this issue efficiently. In the preprocessing phase, high resolution visual mesh is to be bound to low resolution collision mesh based on geodesic path calculation. In the simulation phase, the cutting has to be introduced into collision mesh, and the visual mesh has to be cut with local 2D coordinates. The simulation mesh and all the mappings are then have to be updated to adapt to the cut.
3. Using different resolutions for visual, collision and simulation in a VR surgical simulator. A multi-resolution based simulator has to be presented where high resolution triangle meshes have to be used for visualization of the surgical scene, while low resolution triangle

meshes have to be used for the collision detection and response, and low resolution hexahedral meshes have to be used for the deformation of the thin deformable tissues. The force will be then generated from the collision mesh and mapped to the hexahedral mesh. The deformation of the hexahedral mesh has to be driven by liner co-rotational FEM, and then mapped to the visual and collision mesh respectively.

Based on the above approaches, this thesis will produce the following research novelties:

1. An efficient method to enhance coarse simulation of thin deformable objects with local detailed deformation. This method will be applied to the simulation of probing a meniscus, and verified by the side-by-side comparison with real surgical videos.
2. A framework for the cutting simulation that supports different resolutions of visual, collision and simulation mesh. This framework will be applied to several examples, including the cutting of a steak and a thin sheet, as well as the meniscus lesion handling in the VR simulator. Statistic data of the performance will be presented in detail and the comparison with other cutting methods will be made.
3. As a development novelty and for a further way of the research novelty validation a prototype VR knee arthroscopy surgery training simulator will be developed to support the basic operations of the procedure: (1) the examination of meniscus, ligament and cartilage; (2)

punching out the torn tissues of meniscus; (3) removing torn tissues piece by piece; (4) contouring the edges of meniscus. The compatibility of the proposed methods with the existing methods will be verified on the open source platform SOFA. The comparison with the real surgical videos will be made through similar viewing angles.

1.4 Thesis Organization

Chapter 2 reviews the existing VR simulators and describes the gaps between the requirements of knee arthroscopy and the existing simulators. And then, related works are further reviewed, including the existing simulation methods of deformable objects, collision detection and response, as well as the cutting simulation. Based on these surveys, Chapter 3 proposes the hypothesis of this thesis. A method for detailed deformation of thin deformable tissues is proposed in Chapter 4. Chapter 5 presents a method to support both shaving and cutting of thin deformable objects. A prototype VR knee arthroscopic simulator is described in Chapter 6 together with the applications of the proposed model to virtual menisci. The conclusion and future works are in Chapter 7.

Chapter 2 Deformable Objects Modeling for Virtual Knee Arthroscopy Surgical Simulation

Numerous works have been done for the simulation of deformable objects (organs, tissues) for virtual surgical simulation. This chapter only reviews the works that are closely related to the project. Section 2.1 presents the basic knowledge of arthroscopic surgery, which includes the tissues in the knee and the typical surgical procedures on them. Section 2.2 reviews the existing VR-based surgical training. Section 2.3 reviews the simulation of deformable objects. Section 2.4 briefly reviews the collision detection and response of deformable objects. A survey of cutting simulation is given in Section 2.5. The room for research is discussed in Section 2.6. A summary is given in Section 2.7.

2.1 Minimally invasive arthroscopy

There are different types of MIPs such as Laparoscopy, Arthroscopy, Endoscopy, etc. [1, 3]. The knee arthroscopy and its simulation will be discussed in this thesis, while the research results can be extended (or applied) to other MIPs.

2.1.1 Surgical areas in the knee

To better understand the knee arthroscopy procedure, it is necessary to have some basic knowledge about the knee structure. There are some standard medical textbooks and online resource available (e.g. [1, 3]). Here only a brief introduction is given.

2.1.1.1 Joints and bones

The front, side and back views of the knee [6] are illustrated in Figure 2-1. Two major bones in the knee are tibia and femur. Tibia is on the lower part of the knee (the shinbone), while femur is located at the upper part of the knee (the thighbone). The top of the tibia meets the end of the femur and the combination of them constitutes the knee joint.

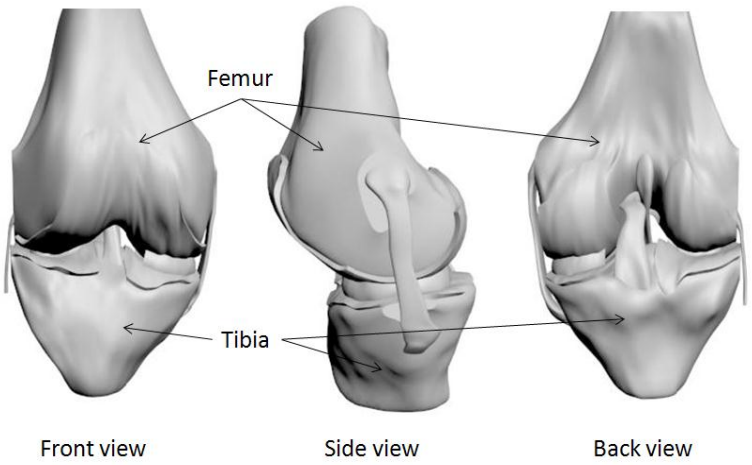


Figure 2-1 Front view, side view and back view (https://simtk.org/projects/openknee).

2.1.1.2 Soft tissues

There are several soft tissues within the knee, which can be categorized into articular cartilages, ligaments and menisci. Generally, soft tissues are deformable. The elastic properties of soft tissues within the knee remain challenging [7-9]. This sub-section gives a brief description of the elasticity of the soft tissues that is related to the surgical simulation. According to the description from the collaborative surgeons in this project, in the healthy tissues,

the feeling of cartilage is like a pencil eraser. It is also learnt that meniscus is a little softer than the cartilage and the ligament is softer than the above two tissues. When it comes to the injured tissues, their feeling will become less firm than that of healthy ones.

Articular cartilages are the soft tissues that cover all the ends of bones and prevent their direct contact. Within the knee, the shocks between the bones are absorbed by articular cartilages that cover the ends of tibia and femur (Figure 2-2). Articular cartilages are white tissues about 0.25 inches in average. The slippery property of articular cartilage prevents the damages when bones are sliding against each other.

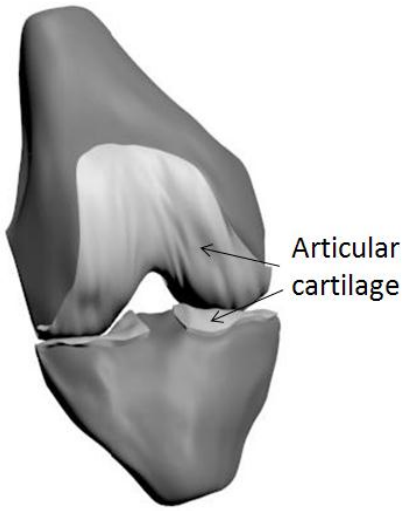


Figure 2-2 Articular cartilage within the knee (https://simtk.org/projects/openknee).

Many constitutive models have been proposed for the elastic property of cartilages. These models include, single-phase viscoelastic models [10], poroelastic and biphasic models [11], fiber-reinforced nonlinear models [12],

and triphasic models [13]. Besides, the properties of cartilages in different situations were also studied, such as influence from other tissues [14], electromechanical properties [15, 16], patient-specific constitutive form [17], tissues contact in walking [18], the contribution of constitutive components [19], and the properties in knee cartilage loss [20].

Ligaments are the soft tissues inside the knee which include the lateral collateral ligament (LCL), anterior cruciate ligament (ACL), posterior cruciate ligament (PCL), and medial collateral ligament (MCL) (Figure 2-3). LCL and MCL are on the side of the knee. The function of MCL and LCL is to control the side-by-side moving of the knee. ACL and PCL are in the middle of the knee joint. The function of ACL and PCL is to prevent the knee of moving too far in the front and back direction. All the ligaments guarantee the stability of the knee.

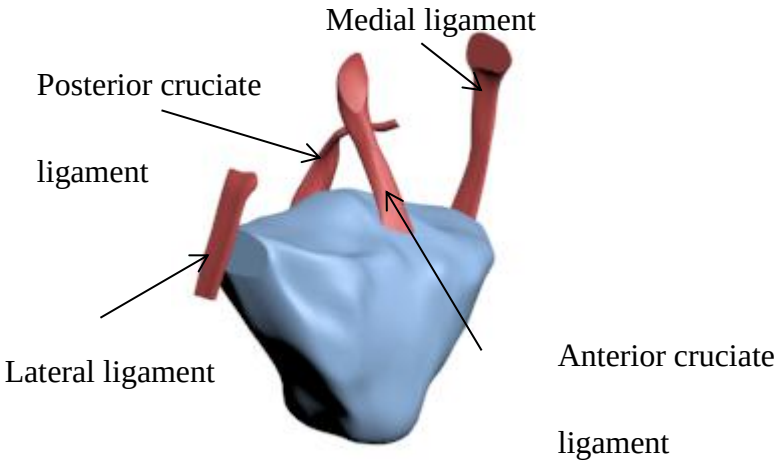


Figure 2-3 Ligaments within the knee (https://simtk.org/projects/openknee).

Most of the proposed models for ligament are based on the behavior of collagen, since the mechanical behavior of ligament is mostly due to the collagen fibers. Nonlinear behavior was considered in [21, 22]. Hyper elasticity and strain energy were proposed in [23] and viscoelasticity was discussed in [24]. Besides, extensive research efforts have been made, such as loading during jump landing [25], measure strain and forces *in vivo* [26], and biomechanics during simulations [27].

Menisci are special types of ligaments lying between tibia and femur (Figure 2-4). The shape of menisci is crescent like, and they are thinner inside but thicker outside. The functions of menisci are: (1) distribution of the force between tibia and femur and (2) together with ligaments, stabilization of the knee joint. During sports, the whole body weight exerts on the menisci, and this is the reason why they are so frequently torn by athletes.

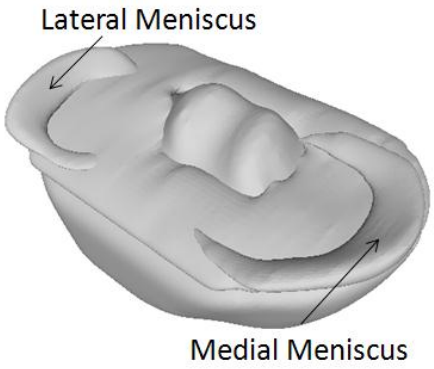


Figure 2-4 Menisci.

Single-phase nonlinear elastic was considered in [28]. Axis-symmetry and isotropic behavior were considered in [29]. The fiber reinforcement was

discussed in [30] and biphasic linearity was used in [31]. Besides, many research efforts have been made on the meniscus, such as mechanical response in indentation experiments [7], contact mechanics using a sensor [32], micromechanical properties of extracellular matrix [33], loading in ACL injured knee [8], and structure and function relationships of tissues [34].

2.1.1.3 Discussion on soft tissues

Cartilages, ligaments and menisci are thin deformable tissues, i.e. their thinnest dimension is significantly smaller than the other two dimensions. Extensive research works have been done on the properties of these tissues. Among the existing computational models, finite element method is the most popular [7, 35-38]. However, most research works were based on the data from animals or cadavers [39], since it is unethical to sample data from human tissues *in vivo* (in the living organism). The biomechanical properties in animals and cadavers are generally different from the counterparts in living human body [9]. Furthermore, the properties of tissues vary from person to person [40]. Therefore, the biomechanical properties of cartilage, ligament and meniscus of human *in vivo* are still open research topics. For surgical training, the simulation of human tissues *in vivo* is generally required, since the real surgeries are performed on living human body. A conclusion could be made even without a survey on the existing real time simulation methods, which is that the complete validation of these methods is impossible presently.

2.1.2 Typical knee arthroscopy procedure

The typical knee arthroscopy procedures include the insertion of the camera and tool, meniscus inspection and partial meniscectomy. Only a brief description is presented and more details can be seen in [1]. In a general procedure of minimally invasive knee arthroscopy (KA), surgeons make two small incisions on the skin of the knee and insert a camera and a tool through these keyholes into the knee. The scene inside the knee will be displayed on the monitor. Surgeons watch the monitor while operating on the camera and tool. Usually, the procedure begins with the examination of the knee to see where the damage is. Then surgeons make the decision on how to repair the torn tissues according to their conditions.

2.2 VR-based arthroscopy simulators

Traditional training approaches include using of cadavers, animals, mock-ups, multimedia tools, and the training on real patients [41]. The purpose of VR-based surgical simulators is to supplement the existing training methods [42]. The VR simulation of surgery has been studied for more than thirty years. The existing arthroscopic simulators can be classified into two categories: research-based and commercial-based. They are reviewed in the following two subsections.

2.2.1 Research-based VR arthroscopic surgery-training simulators

In 1995, Ziegler et al. [43] introduced a VR knee arthroscopic simulator which was the result from the cooperation of computer graphics scientists and

traumatologists. They listed the requirements for arthroscopic simulators and constructed 3D knee mesh model from MRI data. Mesh decimation was applied to the constructed model to increase the system's performance. The simulator was run on a workstation connected with a tracking system. It was tested by more than 40 participants. Although positive assessments were given to the simulator, no collision, deformation and force feedback were provided by it. In 1997, Gibson et al. [44] discussed the real-time rendering of voxel-based volumetric models and used 3D ChainMail algorithm [45] to simulate the deformation of soft tissues. A PHANToM desktop haptic device from SensAble was used for interaction. In paper [46], a simulator called VE-KATS was described. In this system, an accustomed haptic feedback device was utilized and an electromagnetic tracking system was used to track the instruments. The deformation of tissues was simulated by finite element method. A collision depth map was developed for the collision handling. In the simulator developed by Megali et al. [47], different modules were presented. These modules included anatomical knowledge, procedural knowledge and navigation training. However, no collision detection and deformation methods were discussed. There are many limitations of the early works above. The simulators in [15][16] worked on workstation. Only the simulator in [16] provided the deformation of soft tissue which was yet unrealistic. Also the interactions provided by these early simulators were limited.

In the simulator proposed by Heng et al. [48] in 2004, a tailor-made haptic device was used. The deformation and cutting of soft tissues were simulated by

a hybrid finite element method (FEM). They generated surface mesh by the triangulation method proposed in [49]. Tetrahedral mesh was produced by the method proposed in [50]. In their hybrid FEM, they separated the organs into two regions: non-operational and operational. The non-operational region was simulated by linear and topology-fixed FEM. The operational region was simulated by complex FEM which supporteds topological change and non-linear deformation. These two regions were connected by additional boundary conditions. The cutting operation was only allowed in the operational region. The cutting was performed on tetrahedral mesh, and the operation was identified by five general cases. The crack-free tetrahedral subdivision was separated into eight categories. A simplification method was applied to improve the mesh quality after cutting. This simulator combined haptic feedback, collision detection, soft tissue deformation, and cutting, however, specific procedures like shaving of tissues and ACL reconstruction were not supported.

In 2009, based on the framework proposed in [51, 52], Lu et al. presented knee arthroscopic simulator that supported the training of partial meniscectomy [53]. They used mass-spring method to simulate the deformation of menisci. This system also provided different training choices, covering basic portal selection, triangulation of the camera and the tools, and the procedure of torn meniscus curing. Although different levels of training were provided, realistic visual feedback provided by the system was still limited.

In 2013, several simulators were proposed. A simulator was proposed by Lyu et at. [54] to utilize the experience of surgeons. They recorded the

trajectory of the inspection procedure from experienced surgeons and used B-spline curves to filter the noises. The trainees were then guided by the recorded trajectory using the force generated from the difference between current position and the recorded position. In this way, the learning curve of novice trainees could be shortened. Rasool et al. [55] discussed the hand-eye coordination training in the virtual knee arthroscopic simulator, where they detailed the hand-eye coordination in both 3D and image-based environments. In the 3D environment, the incision point was selected on a 2D image. The 3D scene in the knee was displayed when the camera was inserted. In the image-based environment, the surgical scene was defined by panoramic images. These images were constructed by stitching the snapshots from real arthroscopic videos. Wang et al. [56] proposed a simulator that not only supported the basic operation of probing and cutting, but also the incision, puncturing and drilling. Based on these operations, the arthroscopic ACL reconstruction was simulated. A linear finite element method was used to simulate the deformation of skins and menisci. A position-based method was coupled with generalized cylinder to simulate the rope-like objects such as tendons and ligaments. The simulation of tunnel construction and bone drilling were also presented.

The simulations of shoulder and wrist arthroscopy were also studied. In the shoulder arthroscopic simulator developed in [57], the considerations from expert surgeons were described. There were three modules for this simulator, covering multimedia area, simulation kernel and interaction system. In the wrist

arthroscopic simulator presented in [58], the objects within the wrist joint were generated from CT images. A linear programming method was utilized for the collision detection. Besides, an accustomed haptic feedback device was implemented.

To summarize, the core of developing VR surgical simulators is human organs modeling. The methods used for the simulation of deformable tissues affect the performance of the whole system heavily. Existing methods in the simulators discussed above include mass-spring method, chain mail method, finite element method and position-based method. A direction that deserves further investigation is the models for the simulation of deformable organs in VR surgical simulators.

2.2.2 Commercial VR arthroscopic simulators

In addition to the research-based VR simulators, several commercial arthroscopic simulators are also available around the world. However, the technical details of these simulators are not published. The information can only be collected and approximated from websites, product brochures and videos provided by the respective companies.

Simendo Pro arthroscopy from SIMENDO [59] has a lower leg mock-up with an inserted instrument, and it provides for several training courses. In the course of “Creating space”, in order to create a suitable workspace within the knee, the trainee needs to move the lower leg mock-up. In the course of “6 Boxes”, the basic skills of using angled camera are trained by inspecting 6 open boxes with different positions and angles. In the course of “Knee navigation”,

the training of inspecting the knee joint is guided by the predefined trajectory. In the course of “Knee inspection”, the camera-tool triangulation is trained by touching the specified positions within the knee. In the course of “Knee manipulation”, the manipulation of soft tissue is simulated by pulling some preset cylinders that are close to meniscus via a hook. This company provides an online scoring platform for trainees, teachers and supervisors. The face validity and partial construct validity was reported by Tuijthof et al. [60]. However, no interaction with organs was provided. Besides, basic procedures like partial meniscectomy were not supported.

ArthroSim from ToLTech [61] is the result of cooperation of the Arthroscopy Association of North America (AANA), the American Board of Orthopedic Surgeons (ABOS), the American Academy of Orthopedic Surgeons (AAOS) and Touch of Life Technologies (ToLTech). It has two monitors and lower leg mock-up, as well as an arthroscope and a tool. The right monitor is used to display the surgical scene within the knee. The left monitor is used to display all the rest information such as the instructions of surgery, educational videos from real surgery, scoring module and a 3D knee model. The arthroscope and tool are specially designed haptic devices that are very similar to the original ones in the real surgery. This simulator supports the simulation of the whole knee inspection, covering the inspection of menisci, PCL, ACL, articular surface, etc. The haptic and visual interaction between tools and tissues is supported and the finite element method is used for the deformation of soft tissues. A training curriculum is provided by the AAOS, which utilizes text,

images, movies and animations. Both knee and shoulder simulations are provided in this system. However, many useful procedures are still not currently supported, such as shaving the rim of meniscus and partial meniscectomy.

ARTHRO Mentor from Simbionix (has been acquired by 3D SYSTEMS company) [62] is a simulator that provides both knee and shoulder modules. It includes a video monitor, a full leg mock-up, two haptic devices and several tools. For the knee module, the diagnostic tasks support the inspection of healthy and pathological joint, several tears (longitudinal tear, horizontal tear, flap tear, radial tear, etc.), while the procedural tasks include ACL repair, meniscectomy, micro fracture and loose body removal. For the shoulder module, the diagnostic tasks include the glenhumeral and subacromial examination, cartilage lesion, etc, while the procedural cases support micro facture, suture handling, loose body removal, etc. The effectiveness of this simulator in the education of junior residents was confirmed by several professors from NorthShore University and University of Chicago.

ArthroS from VirtaMed [63] is the result of cooperation of ETH Zurich and Balgrist University Hospital. It includes a video monitor, a knee mock-up and several tools. ArthroS provides the training of basic skills, diagnostic arthroscopy and therapeutic arthroscopy. In the module of basic skills, it provides 9 courses for the training of diagnostic tour, handling of instruments, detecting and eliminating of lesions. In the module of diagnostic arthroscopy, it provides the training of 8 cases with different difficulties to let the trainees practice the whole knee inspection and the handling of instruments. In the

module of surgical knee arthroscopy, it provides the training of the cutting and shaving of soft tissues, as well as the grasping and removing of chips. For the training of shoulder arthroscopy, ArthroS also provides 10 cases for basic skills training, 8 patients for diagnostic training and 8 patients for therapeutic training. ArthroS is the only simulator that supports cutting of the menisci. The number of validation studies for this simulator is growing for the past two years [64-67]. A comparison study concluded that ArthroS has better overall face validity than ARTHRO Mentor and ArthroSim [67].

Table 2-1 summarizes the above commercial arthroscopic simulators in terms of the human organ models. ArthroSim and ARTHRO Mentor support active haptic feedback. Only SIMENDO arthroscopy does not support the deformation of soft tissues. Only ArthroS provides cutting and shaving of soft tissues. It is clear that the soft tissue simulations provided in these simulators are limited.

Table 2-1 Summarization of the commercial simulators

	Haptic feedback	deformation	cutting	shaving
SIMENDO arthroscopy	NO	NO	NO	NO
ArthroSim	Active	YES	NO	NO
ARTHRO Mentor	Active	YES	NO	NO
ArthroS	Passive	YES	YES	YES

2.2.3 Discussion on VR-based simulators

The validation of the existing VR-based arthroscopic simulators is an on-going research topic [68, 69]. The current results have already shown that the VR-based surgery-training simulator is a promising direction [68]. However, most of the validations were conducted on the basic training like hand-eye coordination. Many procedures have not been validated (e.g., contouring the edge of meniscus), since the simulated procedures provided by the existing simulators are limited.

According to the discussion in [42, 43, 68] and the opinions from collaborative arthroscopic surgical doctors, an ideal surgical simulator should provide the following:

1. Realistic and efficient models of tissues within the knee.
2. Interactive handling of camera and instruments.

3. Real time simulation of all kinds of procedures, including probing, cutting and suturing of soft tissues.
4. A good user interface that allows for choosing of courses of different levels.
5. Objective assessment in detail for trainees, such that the trainees can find out what they missed in the previous training, and what they need to improve in the next training.
6. Training of abnormal surgical cases.

Only limited parts of the requirements have been achieved so far. The accuracy and efficiency of current VR-based simulators need to be improved and more study is required. Therefore, a conclusion can be made that there is room for research for the VR-based surgery-training simulators. However, the purpose of this project is not to solve all the issues, but to focus on the improvements in the simulation of soft tissues (primarily meniscus, as well as ligament and cartilage). To achieve this goal, a survey of the existing models for the real-time simulation of deformable objects was performed and reported in the following sections.

2.3 The simulation of deformable objects in computer graphics

Numerous works have been done for the simulation of deformable objects (organs, tissues). This chapter only reviews the works that are closely related to the project. The simulation of deformable objects in real time is a challenging research topic in computer graphics. Extensive methods have been proposed for different applications. Comprehensive surveys on the physically-based methods

in computer graphics could be found in [70-73]. Recently, a review for the deformable models in surgical simulation was presented in [74]. Besides, introduction courses could be found in [75, 76]. Generally, the methods for simulating deformable objects could be applied to the simulation of thin deformable tissues. This section gives an overview of the simulation methods.

With reference to Figure 2-5, deformation methods can be classified into geometrically- and the physically-based. *Geometrically-based* deformation does not consider the physical properties of the objects. This category includes shape modification of NURBS [77, 78], T-spline [79-81], free form deformation (FFD) [82, 83] and surface based on differential properties [84, 85]. The models in this category are generally not suitable for the surgical simulation, since they are not physically plausible. In contrast, this project focuses on the *physically-based* simulation, which can be further classified into basic models and enriching models. The basic deformation methods can be divided into two categories: discretely-based and continuum-based. *Discretely-based* methods define forces, constraints or connectivity directly on the discrete representation of objects. This category includes Mass-Spring methods (MSM), chain mail methods (CMM), mass-tensor methods (MTM), and position based dynamics (PBD). *Continuum-based* methods utilize the laws of continuum mechanics and solve the governing differential equation in each time step. This category includes finite element methods (FEM), finite difference methods (FDM), finite volume methods (FVM), boundary element methods (BEM), extended finite element methods (XFEM), frame based methods (FBM), meshless (or meshfree)

methods, etc. For the real-time simulation, the degree of freedoms (DOFs) used in basic methods are limited. Therefore, different enriching methods were proposed. This category includes the methods of data-driven, model order reduction and procedural enriching.

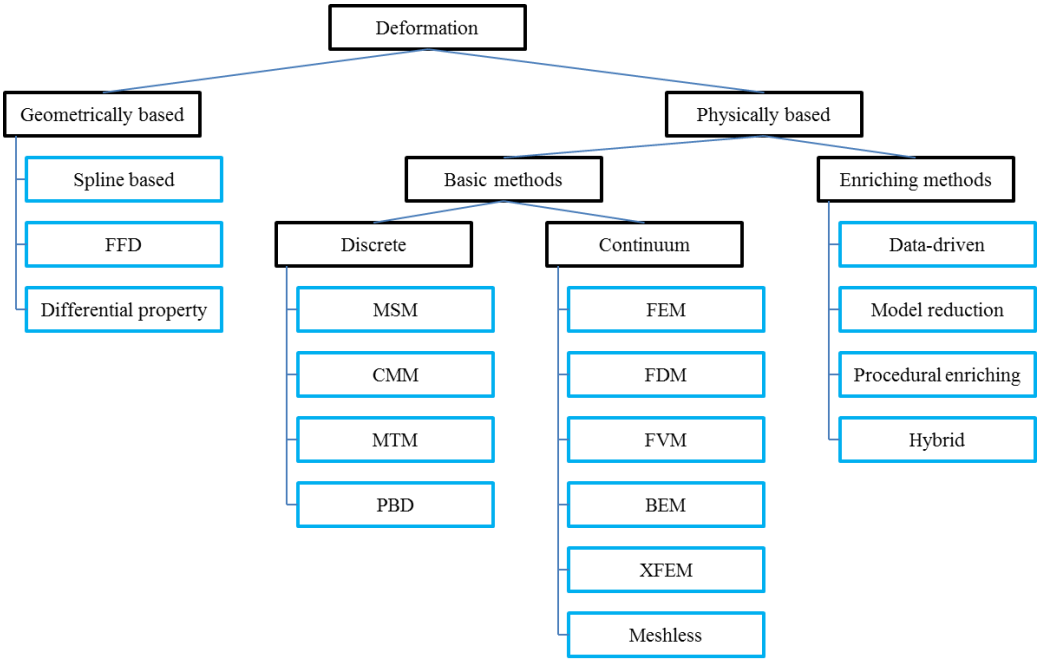


Figure 2-5 Overview of the existing deformation methods.

2.3.1 Physically based simulation methods

There are two challenges for the physically-based simulation: to be realistic and to work in real-time. Generally, these two aspects affect each other. The improvement in one aspect usually leads to the detriment of the other. Considering the representation of the object domain, simulation methods can be divided into the discretely-based and the continuum-based.

2.3.1.1 Discretely based methods

In the category of the discretely-based methods, forces, masses or constraints are defined on the discrete geometric representation of objects. The most popular methods in this category are MSM and PBD.

MSM assigns masses to the points and defines forces by the elastic springs that link these points [86, 87]. During the simulation, when the external forces are introduced, the new positions of these points are calculated by balancing the forces in each time step. MSM is efficient and simple to implement. Research efforts have been made on the simulation of heterogeneity [88], nonlinear behavior [89], multi-layered structure [90], and local deformation [91]. MSM has been applied to many surgical simulations, such as laparoscopic surgery [89] and orthopedic surgery [90]. However, MSM suffers from the drawbacks of limited accuracy, convergent issue and dependency of geometrical structure.

PBD defines constraints on points and calculates new positions by projecting constraints using a Gauss-Seidel solver [92-94]. This method is particularly suitable for the interactive simulation, since it is simple, fast, robust, stable and controllable. Many constraints have been proposed in this framework, including bending, stretching, self-collision, volume conservation, density, etc. PBD has been applied to the development of many applications, such as cloth simulation [95, 96], and fluid simulation [97, 98]. Although there are a few papers that applied PBD to surgical simulation [99-101], generally this method is physically wrong and not designed for the applications in which accuracy is strongly required.

The discretely-based methods are generally fast and simple to implement. However, the methods in this category have some common drawbacks, e.g., limited accuracy and convergence issue.

2.3.1.2 Continuum based methods

In the category of continuum-based methods, deformable objects are considered as continuum medium. The simulation of mechanical behaviors relies on the continuum mechanics of solid and the constitutive laws. The most popular method in this category is FEM.

In the FEM simulation, the governing equations and constitutive laws are approximated by finite elements (such as tetrahedra and hexahedra), and the positions are updated by solving a large system of equations which assemble the material properties [70, 75]. Extensive research efforts have been made for the application of FEM in computer graphics, such as matrix condensation [102], strain limiting [103], non-linear [89, 104], and GPU implementation [105, 106]. For surgical simulation, the most popular method would be the Co-rotational FEM [40, 71, 107-111], which applies FEM in the rotated local frames with linear materials. Accuracy and efficiency are generally contradicted to each other. Real-time simulation usually requires coarse finite element mesh, which would generally lose the accuracy. Bui et al [110, 112] proposed an error-driven approach to control discretization error on coarse Co-rotational linear FEM simulation by template based local h-refinement. However, the modeling error and numerical error are not discussed in this method, thus the total error is still not controlled.

Continuum based methods are generally more accurate than discretely based methods. However, the accuracy can only be achieved by a large number of DOFs. For the real time application, coarse resolution has to be used, thus subtle deformation such as wrinkles is generally lost; the total error could be intolerable and therefore the simulation result is still only visually plausible.

2.3.2 Enhanced methods

Accurate simulation is usually achieved by fine resolution simulation mesh, with which the real time constraint is difficult to meet. Real time is generally achieved by coarse resolution simulation mesh, with which the fine local deformation that is essential for the physically plausible simulation is lost. In this sub-section, various enriching methods to enhance the efficiency or the accuracy of the simulation are briefly reviewed.

2.3.2.1 Data driven methods

The core idea of data driven methods is in using the captured real data to enhance the low resolution simulation data based on a reasonable relation between them [113, 114]. Data driven methods have been successfully applied in the animation of many objects, such as cloth [115], facial [116], and body [117]. For the surgical simulation, although there are some methods to capture the material properties indirectly [114, 118], it is generally unethical to sample human organ data *in vivo*. Seiler et al. [119] proposed to use high resolution nonlinear simulation results as example data and to enhance low resolution linear simulation based on a nonlinear mapping. However, in the arbitrary

cutting simulation, a deformable object could be separated into infinite parts, and thus requires theoretically unlimited pre-stored data.

2.3.2.2 Model order reduction

The application of model order reduction (MOR) in engineering has a long history [4]. The core idea is to select good sub basis that captures the major behavior and project the deformation. This usually reduces the degree of freedom from about 5~10 thousand to 12~30, thus real time simulation could be achieved. However, the local deformation is usually lost due to the limited deformation modes. Multi-domain [120] and hierarchy [121] methods are proposed to enhance the local deformation. However, the true local behavior is not achieved unless very fine sub-domain is applied, since these techniques are based on the subdivision of the original domain. A method for the local deformation was proposed by Harmon et al. [122], however only linear local deformation was addressed. Besides, many research efforts, such as collision handling [123], secondary dynamics [124], optimal approximation of the manifold [125], and incremental eigenvalue calculation [126], have been made. The methods of subspace integration are promising methods since visual and haptic real time can be achieved for soft tissues deformation. For the simulation of cutting (fracturing), Kerfriden et al. [127, 128] proposed a local/global technique to couple the cracked and intact domains. However, full simulation is still required for the fractured (topologically changed) domains. Furthermore, it is difficult to simulate the detailed deformation (such as little wrinkles on human faces) with MOR due to the lack of DOFs.

2.3.2.3 Procedurally enhanced methods

The basic idea of procedurally enhanced methods is to enrich the coarse deformation with the details that are generated from the state of the coarse simulation. In [129], based on the stretch tensor on the coarse simulation mesh, realistic wrinkles are generated to augment the coarse deformation. Muller et al. [96] defined a wrinkle mesh attached to the original mesh with some constraints, then evolved the wrinkle mesh by a static solver, and used Bezier interpolation to avoid the shortcoming of piecewise linear shape. In paper [130], based on harmonic functions, local wrinkles for the draping of cloth were generated by solving simple elliptical equations. The advantages of procedural details generation method are: (1) The local details can be controlled by some parameters; (2) It is not difficult to apply this method to the application with topological changes. Procedural methods were mainly applied for the detailed deformations such as: cloth [131, 132], faces [133, 134], and many other surface of objects [135]. The major disadvantage is that the deformation generated by procedures may look artificial. Although the above methods could generate visually plausible detailed deformation, the quantitative errors may still be large.

2.4 Collision detection and response of deformable objects

Collision detection and response is essential for many applications, a good review can be found in [136]. The most popular methods are bounding volume hierarchy [137], signed distance field [138], and spatial hashing [139]. For most of the collision handling methods, the performance could be improved if the

number of collision primitives (DOFs) decreases. Spillmann et al. [140] proposed a spatially reduced framework, in which a high resolution triangles mesh was used for rendering, a coarse resolution triangle mesh with radii was used for the collision handling, and a coarse volumetric mesh was used for the simulation. The robustness and efficiency for the collision handling of this framework have been proven by extensive example. However the cutting simulation of this framework was left as a future work, thus the application of this framework was limited.

2.5 Cutting simulation of deformable objects.

An excellent review for the cutting simulation can be found in [136]. There are three models for a deformable body: visual, collision and simulation models. The existing cutting methods can be categorized according to the relation of these models.

2.5.1 Cutting of conformal mesh

Many cutting frameworks use one conformal mesh for visualization, collision and simulation. Tetrahedral meshes are commonly used in this category. The simulation of deformation is run on tetrahedral meshes while the outside faces of the meshes are used for visualization. The collision can be handled by the boundary triangles or the tetrahedral meshes. The cutting operation is performed on the tetrahedral meshes, while the visual collision mesh is updated accordingly. Many methods have been proposed, including element removal [141], element refinement [142, 143], face-splits [144-146], node snapping [147], and their combinations [148, 149]. The advantages of conformal mesh

are that the cutting needs only to be considered on the tetrahedral mesh, and thus the complexity of updating the embedding between different meshes can be avoided. The drawbacks of using conformal tetrahedral meshes are that the resolution of visualization and collision has to be aligned with the resolution of simulation. Increasing the resolution of any of the three models directly leads to the resolution increments of the other two models.

2.5.2 Cutting of multi-resolution mesh

The multi-resolution simulation framework has been adopted by many researchers. Tetrahedral meshes were used as simulation meshes in [150-152]. The virtual node method proposed in [150] generated a well-shaped tetrahedron for each material branch. Although arbitrary cut [151] and Graphics Processing Unit (GPU)-accelerated simulation [152] can be supported by this method, the connectivity analysis of tetrahedra is not very efficient.

The idea of composite finite elements (CFEs) was introduced into the computer graphics community by Nesme et al. [153]. In this simulation framework, a high resolution visual mesh is embedded into a fine hexahedral mesh, which is also embedded into a coarse hexahedral mesh. Branch analysis is performed so that for each material component a hexahedron is duplicated and assigned. Jeřábková et al. [154] simulated the cutting by element removal of the fine hexahedron and the dynamic branch analysis. Although interactive cutting rate could be achieved, the visual quality and mechanical accuracy were limited due to the element removal operation.

Based on the octree structure, Dick et al. [155] connected coarse hexahedral mesh to fine hexahedral mesh. The cutting was simulated by disconnecting the links between fine hexahedral elements. Wu et al. [156] improved the generation of visual mesh by the dual contour method. They also developed a method for the collision detection and response for this model [139]. The performance was further improved by Jia et al. [109] with Central Processing Unit (CPU)-GPU mixed implementation. Therefore, this model supports not only interactive cutting, but also efficient collision handling. However, both the cutting operation and the collision detection were relied upon the octree structure. The collision vertices could be a down-sampled from the visual vertices to improve the efficiency of simulation. However, the collision handling can only relied on the collision points, which may not approximate the visual mesh well.

Seiler et al. [157] proposed a threefold model for the simulation of deformable objects. In this model, a high resolution surface mesh was used for rendering, a mid-resolution tetrahedral mesh was used for collision detection and response, and a coarse resolution hierarchical hexahedral mesh was used for deformation. Later on, they developed a cutting method for this model [158] which was also applied in a data-driven simulation [119]. The cutting operation was simulated by the removal of tetrahedra that were intersected by the blade volume. Efficient cutting simulation was achieved by this method. However, the resolution of the tetrahedral mesh should have been large enough to represent the shape of the cutting blade, due to the none-split operation they

adopted for the cutting. This compromised the ability of tuning the resolution of collision of this model. The tetrahedral mesh was no longer used for the collision detection and response in their cutting simulation, and only served as a material graph. The collision handling was not discussed, and left for a future work.

A few cutting methods in the framework of position based dynamics (PBD) were proposed recently. Pan et al. [100] combined a surface mesh with inner meatballs; while Berndt et al. [99] further discussed the haptic, electrocautery (thermo cautery), and skinning in the cutting simulation. Although efficient cutting has been achieved, PBD is generally not designed for the surgical simulation due to its accuracy issue. Manteaux et al. [159] developed a method to interactively cut a thin sheet in detailed, the deformation of which is driven by the frame based simulation.

2.6 Critical analysis

For the deformation of objects in surgical simulation, discretely-based methods are generally not preferred due to the lack of accuracy. Theoretically, continuum-based methods can meet the accuracy requirement. However it is usually achieved with high resolution of DOFs, which generally leads to a FPS that is below the real time rate. For the enriching methods, although the accuracy and efficiency can be enhanced by data-driven and model order reduction, it is difficult to apply these methods on the topologically changed domains introduced by the arbitrary cutting. The data-driven methods usually require the pre-storage of large simulation data while MOR generally lost the

detailed deformation. The procedural methods are generally not accurate, and have not been applied in the surgical simulation. For the surgical training, most of the existing methods are only visually plausible. At the same time, it is difficult to validate simulation errors of human organ *in vivo* due to ethical issues.

For the cutting simulation of deformable objects, the conformal models bind together the resolution of visual, collision, and simulation models, thus cannot adapt to different requirements from visualization, collision handling and deformation. Furthermore, the conformal models could generate ill-shape simulation mesh which would lead to an unstable system. The multi-resolution cutting simulation allows for coupling of high resolution visual model with low resolution simulation model. Most of the existing methods handle collision directly by the visual model. In this way, the collision handling could easily become the bottleneck of the whole simulation. The efficiency could be improved by down-sampling the visual vertices. However, the down-sampled collision vertices may not approximate the visual mesh very well.

Based on the above survey and analysis, this chapter concludes that there is a room for research in the deformation and cutting simulation. At the same time, due to the complexity of thin deformable tissues (as discussed in Section 2.1.1), it would be too ambitious to propose a model to simulate all their behaviors in real time. This project narrows the research goal to improve the real-time simulation of some behaviors of meniscus: local detailed deformation and multi-resolution cutting. The proposed model should also be able to

simulate deformations of ligaments and collision handling of cartilages. The local deformation under complex tool-tissue and tissue-tissue interaction are the key features for thin deformable tissues. For example, meniscus has the unique behavior that it can show some wrinkles on the thin edge when it is pressed. Only simulation of this behavior could make the meniscus visually plausible with features, since no such behavior could be found on the other tissues in the knee. A new method should be developed to simulate this behavior. The simulated results should be verified by the side-by-side comparison with real surgical videos. For the cutting simulation, coarse collision meshes are preferred to couple with fine visual meshes in the real time system. However, it is difficult to update the collision and visual meshes consistently when the resolutions of them are different. If the cutting is performed on visual and collision meshes separately, it may happen that the high resolution visual mesh has been cut while the low resolution collision mesh has not. As a result, these two meshes become inconsistent. A new cutting method should be developed to overcome this issue. The accuracy and efficiency should be verified by different examples with complex collision handling. Furthermore, as this project studies the thin deformable tissues (such as meniscus, ligament and cartilage) for the simulation of minimally invasive surgery, a prototype surgical simulator should be developed to prove the feasibility of applying the proposed methods in such simulation.

2.7 Summary

This chapter has briefly described the minimally invasive knee arthroscopic surgery, covering the thin deformable tissues within the knee (menisci, ligaments and cartilages) and the typical surgical procedures. A survey of the existing VR-based knee arthroscopic surgery-training simulators leads to a conclusion that although VR-based training is a promising direction, more studies need to be done. This thesis chooses to work on the modeling of thin deformable tissues (primarily meniscus, as well as ligament and cartilage), which is the fundamental part of most simulators. Therefore, this chapter has also reviewed the research topics for the simulation of deformable objects, covering deformation, collision detection and response, and cutting simulation. An overview of the deformation has been given, followed by the brief discussion of each category. The most related collision methods have also been analyzed. The existing cutting methods have been classified and discussed. Based on the knowledge of these existing works, the conclusion has been made on what models are yet to be designed and how their validity can be tested. In the next chapter, the research hypotheses and goals will be discussed.

Chapter 3 Research Hypothesis and Plan

The previous chapter has reviewed the deformation and cutting methods which can be used in VR surgical simulation and concluded that there is a room for research. In this chapter, the hypothesis of this thesis is proposed.

In Section 2.6, a conclusion has been made on the existing simulation methods that the coarse simulation mesh loses the ability to produce detailed deformation.

Based on the above conclusion, the first hypothesis is made that embedding points into coarse simulation mesh will allow for the detailed local deformation to be added to the coarse deformation of thin deformable objects.

It has also been concluded that the existing framework of the cutting simulation cannot support different resolutions of visual and collision mesh.

Based on this conclusion, another hypothesis is made that adding the visual-collision binding will allow for consistent and interactive cutting of the thin deformable volumetric objects with different resolutions.

Thus the research goal of this thesis is to come up with the model for the thin deformable objects that:

1. Can generate local detailed deformation for the tissue. Achieving this goal is described in Chapter 4.
2. Allows for different resolutions of visual, collision and simulation mesh in the cutting simulation. This is described in Chapter 5.

3. Can be integrated into a prototype VR knee arthroscopy surgery training simulator. The implementation of the simulator is described in Chapter 6.

Chapter 4 Function Enhanced Local Detailed Deformation of Thin Tissues

This chapter verifies the proposed hypothesis for the deformation. The setup of the simulation is described in Section 4.1. Section 4.2 describes the coarse FEM simulation of the meniscus. The embedding of the points is illustrated in Section 4.3. The generation of the local detailed deformation is presented in Section 4.4, followed by the experimental results discussed in Section 4.5. Section 4.6 summarizes the whole chapter. The chapter was partially published in [160].

4.1 The Overall Simulation of the Meniscus Examination

As the most challenging task, the meniscus was first simulated and the developed model was further generalized to simulate other tissues. The whole simulation consists of two Geomagic Touch desktop haptic devices (https://www.3dsystems.com/haptics-devices/touch), a 3D printed knee model, and a video monitor for displaying the simulative virtual scene (Figure 4-1).

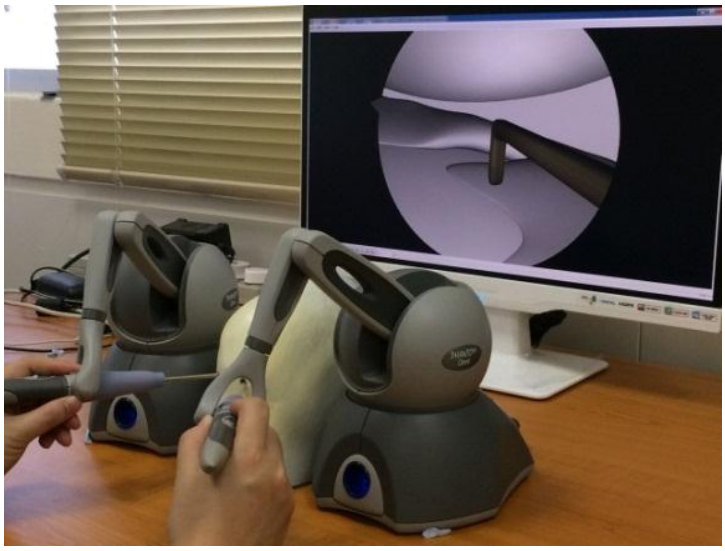


Figure 4-1 The virtual knee arthroscopy simulation setup.

The video of the simulation could be watched at https://youtu.be/NsZvuuEFbPc. There are two small holes made on the knee model, through which the extensions of the two haptic handles are inserted.

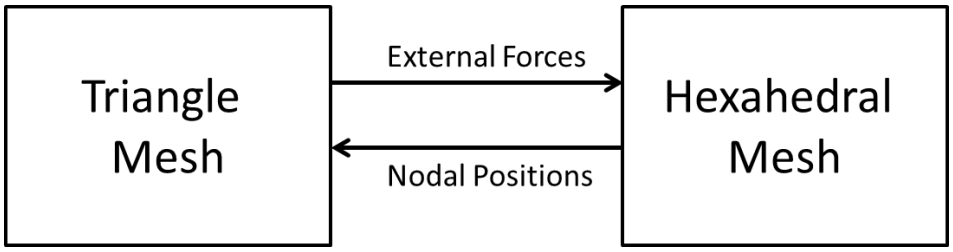


Figure 4-2 Barycentric mapping between (surface) triangle mesh and (physical) hexahedral mesh.

The simulation is conducted on a coarse hexahedral mesh with the co-rotational FEM and an implicit solver (will be described in Section 4.2). With reference to Figure 4-2, a high-resolution triangle mesh for the surface of

meniscus is embedded into the coarse hexahedral mesh by its barycentric coordinates. The relation between the triangle vertices and hexahedral mesh can be represented by the following formula:

$$\mathbf{s} = \mathbf{H}\mathbf{x} \tag{4-1}$$

where $\mathbf{s}$ represents the position vectors of triangle mesh, $\mathbf{x}$ stands for the position vectors of hexahedral mesh, and $\mathbf{H}$ is formed by assembling the corresponding barycentric coordinate. Generally, DOFs of the triangle mesh is much larger than that of the hexahedral mesh. The external force generated from the collision response is mapped to the hexahedral mesh, and the deformation is mapped from the hexahedron mesh back to the surface mesh, all through the barycentric coordinates. There is a need to mention that FEM is only one of the choices to produce the coarse real time simulation for the enriching method proposed in this chapter. Other methods such as MSM and PBD may also be utilized, which will be studied in the future works.

The whole simulation can run in real time, but the fine deformation details of meniscus, such as wrinkles, cannot be generated this way due to the coarse hexahedral mesh. To address this issue, this chapter proposes to add wrinkles into coarse simulation by embedding control points. There are two phases for this method: pre-processing phase and simulation phase, which will be described in Section 4.3 and Section 4.4, respectively. In pre-processing phase, points are embedded into the simulation mesh, and the affected regions on the surface mesh are defined. The pre-processing phase is implemented before the

simulation; hence the time is not critical. In simulation phase, the wrinkles are generated according to the deformation state.

4.2 Co-rotational Linear FEM Simulation of the Meniscus

As reviewed in Chapter 2, co-rotational linear FEM is one of the most popular methods for deformable objects in surgical simulation. Therefore this method is chosen as the base for the simulation of meniscus, and is briefly described in this section. More details could be found in [2, 111, 136].

4.2.1 Governing equations for the meniscus

With reference to Figure 4-3, a meniscus can be regarded as an elastic solid, and the governing equations of its dynamic behavior are formulated as follows

$$\rho\ddot{\boldsymbol{u}} - div\,\boldsymbol{\sigma}\big(\boldsymbol{\varepsilon}(\boldsymbol{u})\big) = \boldsymbol{f_b} \quad in\ \Omega, \tag{4-2}$$

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\boldsymbol{\nabla u} + \boldsymbol{\nabla u}^T), \tag{4-3}$$

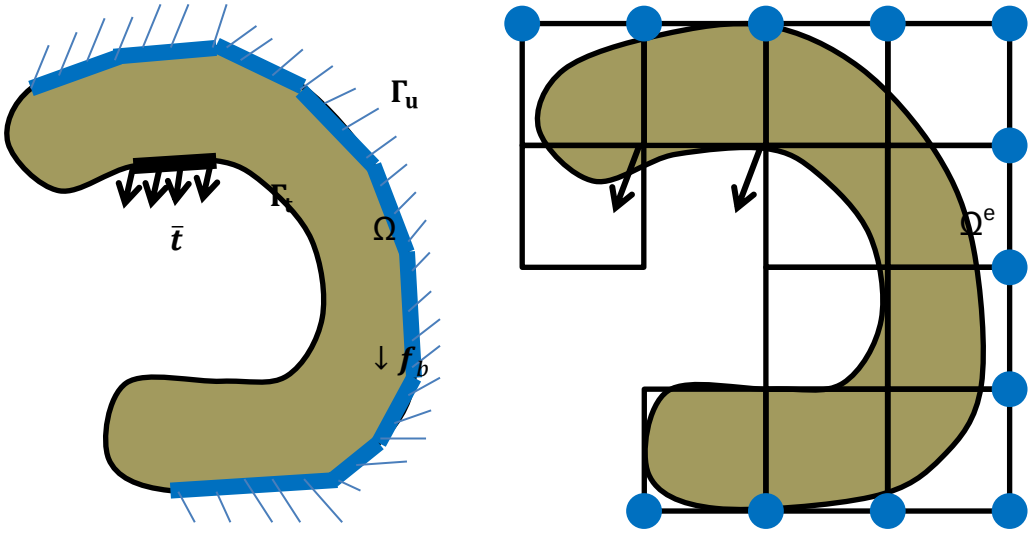
$$\boldsymbol{\sigma} = \mathbf{C}\colon \boldsymbol{\varepsilon}, \tag{4-4}$$

$$\boldsymbol{\sigma}\cdot\mathbf{n} = \bar{\boldsymbol{t}} \ \ on\ \ \Gamma_t, \tag{4-5}$$

$$\mathbf{u} = \bar{\boldsymbol{u}} \ \ on\ \ \Gamma_u, \tag{4-6}$$

where ρ is the mass density, $\mathbf{u}$ is the displacement, $\ddot{\boldsymbol{u}}$ denotes the acceleration, div is the divergent operator, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, $\boldsymbol{\varepsilon}$ is the linear strain tensor, $\boldsymbol{f_b}$ is the body force, $\mathbf{C}$ represents the isotropic linear stress-strain relationship and is related to the Young's modulus and the Poisson's ratio, $\bar{\boldsymbol{t}}$ is the traction on the boundary Γ_t, and $\bar{\boldsymbol{u}}$ is the prescribed displacement on boundary Γ_u. Equation (4-2) describes the dynamic behavior of the meniscus, Equation (4-3) specifies the linear strain (the kinematics),

Equation (4-4) describes the linear material model, Equation (4-5) represents the traction boundary conditions, and Equation (4-6) describes the displacement boundary conditions.



(a) (b)

Figure 4-3 Continuum body of a meniscus and its FEM discretization. (a) Meniscus body Ω subjected to a body force $\mathbf{f_b}$, a traction $\bar{\mathbf{t}}$ on its boundary $\mathbf{\Gamma_t}$ and a prescribed displacement $\bar{\mathbf{u}}$ on the boundary $\mathbf{\Gamma_u}$ (the blue edge) (b) 2D illustration of FEM discretization, the domain Ω is approximated by hexahedral elements Ω^e. $\mathbf{f_b}$, $\bar{\mathbf{t}}$, $\mathbf{\Gamma_t}$ and $\mathbf{\Gamma_u}$ are discretized repectively.

The weak form of the equation (4-2) is obtained by multiplying the equation with an arbitrary test function δv and integrating over the domain Ω:

$$\int_\Omega \delta v \cdot \rho \cdot \ddot{\boldsymbol{u}} dx + \int_\Omega \boldsymbol{\varepsilon}(\delta v) : \boldsymbol{\sigma} dx = \int_\Omega \delta v \cdot \boldsymbol{f}_b dx + \int_{\Gamma_t} \delta v \cdot \bar{\boldsymbol{t}} dx \quad \forall \delta v \qquad (4\text{-}7)$$

Note that equation (4-7) has already considered the boundary condition (4-5).

4.2.2 Finite element discretization

1) Spatial discretization: To solve the above equation (4-7), FEM is used. With reference to Figure 4-3(b), the simulation domain Ω is discretized into a finite set of hexahedral elements. For each element Ω^e, the displacement field is defined by interpolating the displacement of its corner node $\boldsymbol{u}_i^e$ with trilinear shape functions φ_i^e as

$$\boldsymbol{u}_{\Omega^e}(\mathrm{x}) = \sum_{i=1}^{n_e} \varphi_i^e(x)\boldsymbol{u}_i^e = \boldsymbol{\Phi}^e(x)\boldsymbol{u}^e, \tag{4-8}$$

where $n_e = 8$ is the number of nodes of this hexahedral element, $\boldsymbol{\Phi}^e(x)$ is the element shape matrix, and u^e is the displacement vector of element nodes. Based on Equation (4-8), the element strain can be expressed as

$$\boldsymbol{\varepsilon}_{\Omega^e}(x) = \boldsymbol{B}^e(x)\boldsymbol{u}^e, \tag{4-9}$$

For each element, it can be derived from the weak form equation (4-7) that

$$\int_{\Omega^e} \rho(\boldsymbol{\Phi}^e)^T\boldsymbol{\Phi}^e dx\ \ddot{\boldsymbol{u}}^e + \int_{\Omega^e}(\boldsymbol{B}^e)^T\boldsymbol{C}\boldsymbol{B}^e\, dx\, \boldsymbol{u}^e = \int_{\Omega^e}(\boldsymbol{\Phi}^e)^T\boldsymbol{f}_b dx + \int_{\Gamma_t^e}(\boldsymbol{\Phi}^e)^T\bar{\boldsymbol{t}}dx, \tag{4-10}$$

The second term in equation (4-10) is the internal force, which can be written as

$$\boldsymbol{f}^e = \int_{\Omega^e}(\boldsymbol{B}^e)^T\boldsymbol{C}\boldsymbol{B}^e\, dx\, \boldsymbol{u}^e = \boldsymbol{K}^e \cdot \boldsymbol{u}^e = \boldsymbol{K}^e \cdot (\boldsymbol{x}^e - \boldsymbol{x}^{0e}), \tag{4-11}$$

where $\boldsymbol{x}^e$ and $\boldsymbol{x}^{0e}$ are current and initial position vectors of element nodes, respectively. The force calculated by equation (4-11) is not invariant for large rotations. Hence, the co-rotational strain formulation is utilized, and the calculation of internal force becomes

$$\boldsymbol{f}^e = \boldsymbol{R}^e\boldsymbol{K}^e \cdot ((\boldsymbol{R}^e)^T\boldsymbol{x}^e - \boldsymbol{x}^{0e}), \tag{4-12}$$

where $\boldsymbol{R}^e$ is the local element rotation matrix calculated from the current position with respect to the initial position.

Assembling the per-element equation (4-10) and applying Rayleigh damping leads to a global system of ordinary differential equations for the whole meniscus

$$\mathbf{M}\ddot{\boldsymbol{u}} = f(\boldsymbol{x}, \boldsymbol{v}), \tag{4-13}$$

where, $f(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{f}^{ext} - \mathbf{D}\boldsymbol{v} - \boldsymbol{K}\boldsymbol{u}$, $\boldsymbol{x}$, $\boldsymbol{v} = \dot{\boldsymbol{u}}$ and $\ddot{\boldsymbol{u}}$ are position, velocity and acceleration vectors respectively, and $\mathbf{M}$ is mass matrix, $\boldsymbol{K}$ is stiffness matrix, $\mathbf{D}$ is damping matrix calculated as $\mathbf{D} = \alpha\mathbf{M} + \beta\mathbf{K}. \alpha, \beta \geq 0$. Note that, the boundary condition (4-6) is also discretized (see the blue points in Figure 4-3(b)) and enforced by removing the relevant vertices from equation (4-13).

2) Time discretization: To numerically solve the equation (4-13), implicit backward Euler solver [161] is used. Suppose Δt is an acceptable large time step size, define $\Delta\mathbf{x} = \boldsymbol{x}^{t+\Delta t} - \boldsymbol{x}^t$ and $\Delta\mathbf{v} = \boldsymbol{v}^{t+\Delta t} - \boldsymbol{v}^t$, where $\boldsymbol{x}^t$ and $\boldsymbol{v}^t$ are the known position and velocity vectors in the current time step, while $\boldsymbol{x}^{t+\Delta t}$ and $\boldsymbol{v}^{t+\Delta t}$ are the unknown position and velocity vectors in the next time step. According to Equation (4-13), $\Delta\mathbf{x}$ and $\Delta\mathbf{v}$ can be approximated by

$$\Delta\mathbf{x} = \Delta t(\boldsymbol{v}^t + \Delta\mathbf{v}), \tag{4-14}$$

$$\mathbf{M}\Delta\mathbf{v} = f(\boldsymbol{x}^t + \Delta\mathbf{x}, \boldsymbol{v}^t + \Delta\mathbf{v}), \tag{4-15}$$

Applying Taylor series expansion to function f in equation (4-15) and substituting (4-14) into its linearized system, leads to

$$\left(\boldsymbol{M} - \Delta t\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}} - \Delta t^2\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)\Delta\mathbf{v} = \Delta t f(\boldsymbol{x}^t, \boldsymbol{v}^t) + \Delta t^2\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\boldsymbol{v}^t, \tag{4-16}$$

$\Delta\mathbf{v}$ can be obtained by solving linear system (4-16), and then $\Delta\mathbf{x}$ is calculated by equation (4-14). Finally, the position vectors in the next time step $\boldsymbol{x}^{t+\Delta t}$ can be updated.

The above co-rotational linear FEM has already been validated in many previous works [110, 111, 136]. Tetrahedral elements could also be used; however it is easier to handle topological changes with hexahedral elements (cutting will be discussed in the next chapter).

With coarse hexahedral mesh, the simulation can be run in real time, however the detailed deformation is generally lost. To address this issue, a method to add visually plausible wrinkles to the meniscus is presented in the following sections.

4.3 The Embedding of Monitor Points and Region Points

In pre-processing phase, the fundamental idea is that the coarse finite element mesh is incapable of generating wrinkles but rather may contain the clue to the formation of wrinkles. It is observed that the wrinkling of meniscus only appears locally near the inner border (Figure 4-4).

Figure 4-4 Snapshot from an arthroscopic camera view. Red arrow shows the major moving direction of meniscus. Outlined area demonstrates the wrinkles on deformed meniscus.

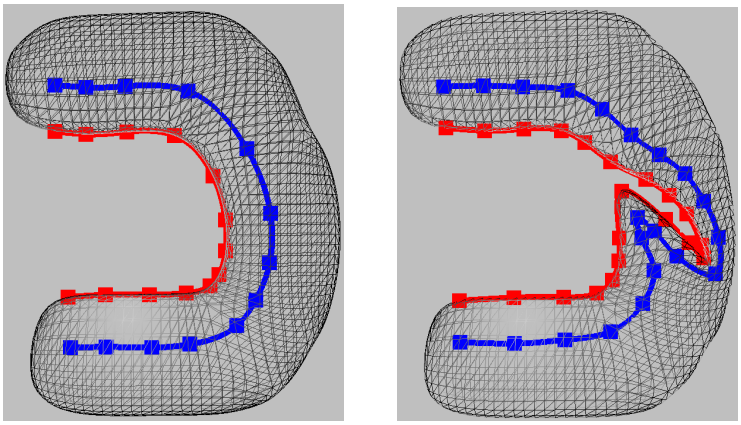
Therefore, in order to monitor and control the local deformation state, the *monitor points* are embedded closely to the inner border of meniscus. But monitor points are not necessarily located on the surface of the meniscus. To define the affected area by these monitor points, the *region points* were also introduced. These points should be positioned inside the meniscus so that the distances between monitor and region points can define the wrinkle area, which is completed as follows.

At step one, a few monitor and region data point pairs were selected along the edge of inner meniscus. With reference to Figure 4-5, the points located on the inner border of meniscus are monitor data points, and the points located inside the meniscus are relevant region data points. At the monitor data points, the thickness was sampled because the wrinkles of meniscus were quite

sensitive to it. The region data points should be located in the middle of the upper and lower surfaces of meniscus (Figure 4-6).

At step two, with reference to Figure 4-5, two smooth curves were generated by interpolating these data points separately. This is achieved by the cubic B-spline curves interpolation [77].

Finally, at step three, the monitor and region point pairs on the B-spline curves are sampled to ensure that the distance between the consecutive monitor points does not exceed a specified upper value, which is necessary for the generation of wrinkles (Figure 4-7).



(a) (b)

Figure 4-5 The generation of curves that interpolate the sampled points. Red points are the monitor data points, and blue points are the region data points. The red curves and blue curves are the interpolated B-spline curves respectfully. (a) Intact meniscus. (b) Torn meniscus.

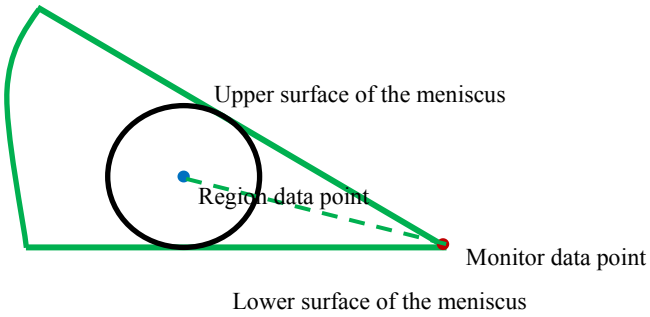


Figure 4-6 Region data points should be placed in the middle of the meniscus. Red points are the monitor data points, and blue points are the region data points.

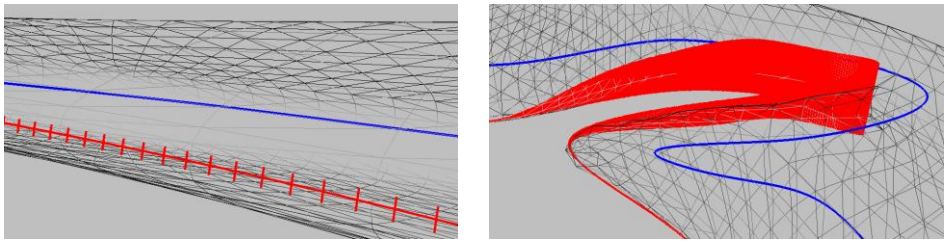
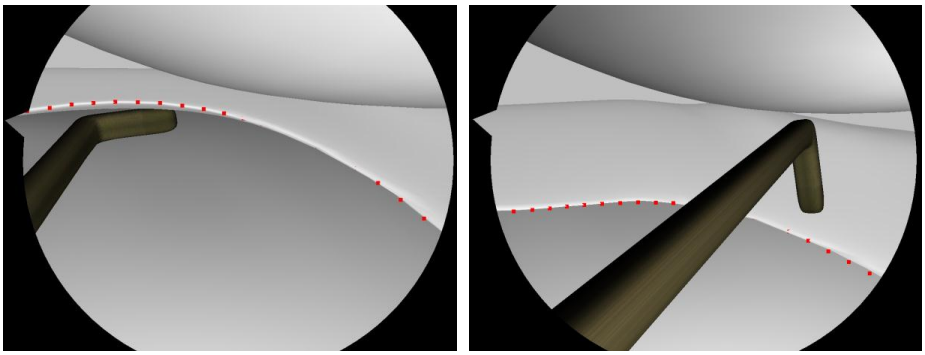


Figure 4-7 Densely sampled points with radii along the inner border of meniscus. (a) Points on an intact meniscus. (b) Points on a torn meniscus.

With reference to Figure 4-8, the monitor points are embedded into the existing hexahedral mesh in the same way as the embedding of surface vertices, as in Formula (4-1). If these points are inside one of the hexahedra, their coordinates will be defined by the linear combination of 8 corner points. Otherwise, the coordinates will be defined by the nearest hexahedron.



(a) (b)

Figure 4-8 Monitor points (red) are embedded into the hexahedron mesh, thus following the movement of the deformation of meniscus. (a) Bend up the meniscus. (b) Press down the meniscus.

With reference to Figure 4-9, the affected region on the surface is defined by both the monitor and region points. Let $\boldsymbol{P}_i^{m0}$ be the rest positions of monitor points and $\boldsymbol{P}_i^{r0}$ be the rest positions of region points. Then, the region line is expressed as $\boldsymbol{L}_i^r = \boldsymbol{P}_i^{r0} - \boldsymbol{P}_i^{m0}$. Let $\boldsymbol{P}_j^s$ be the surface points and $\boldsymbol{P}_j^p$ the projected points from the surface points to region line. Let $\boldsymbol{L}_i^p = \boldsymbol{P}_j^p - \boldsymbol{P}_i^{m0}$. If the length of $\boldsymbol{L}_i^p$ is less than that of $\boldsymbol{L}_i^r$, the surface point $\boldsymbol{P}_j^s$ is marked as affected by $\boldsymbol{P}_i^{m0}$. A large sphere centered at each monitor point is employed to filter out the remote surface points. Each surface point can be affected by several monitor points, and there can be overlapping affected areas.

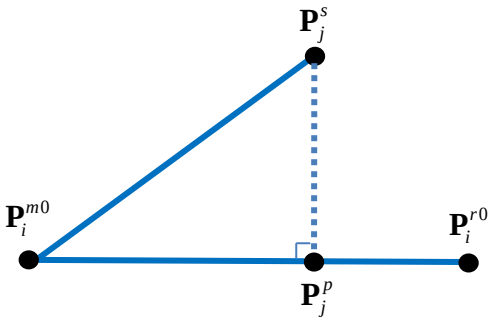


Figure 4-9 The projection of surface points onto the region line.

4.4 Simulation phase

In simulation phase, based on the state of monitor points, the reference configuration of wrinkles is defined. Then, the wrinkles are generated by a function-based method.

4.4.1 Definition of the reference configuration of wrinkles according to the embedded monitor points

Following [119], the contacts are simplified at only one point since the contact area between instrument and meniscus is very small, on which the force is concentrated. The reference configuration and wrinkling area are defined according to the deformation state indicated by the embedded monitor points. More specifically, the state just before the appearance of wrinkles is required to be captured, which is then stored as the reference state for modeling the wrinkles. To monitor the deformation state, the monitor points both at the current and previous time steps are needed. Suppose $\boldsymbol{P}_i^{mc}, i = 1, \cdots, n$ are the monitor points in current step, and $\boldsymbol{P}_i^{ml}, i = 1, \cdots, n$ are the monitor points in

previous step. With reference to Figure 4-10, at every time step, in light of the instrument-to-tissue contact point, the closest point in $\boldsymbol{P}_i^{mc}, i = 1, \cdots, n$ is calculated, then several points $\boldsymbol{P}_k^{mc}, k = i_s, \cdots, i_e$ around this closest point are selected to monitor the local deformation state.

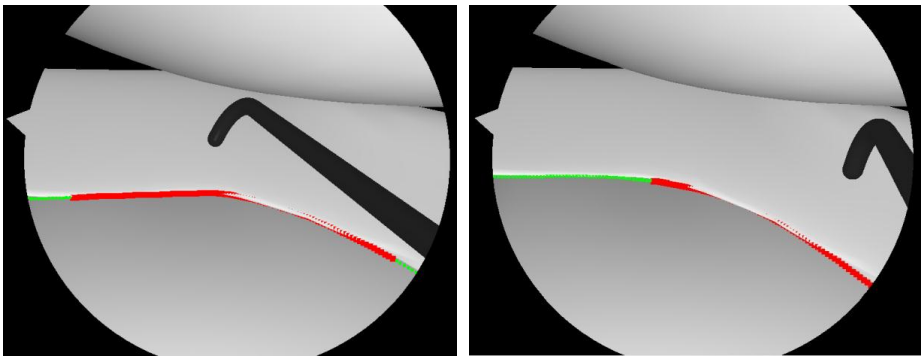


Figure 4-10 In light of the instrument-to-tissue contact positions, different points (red) are selected from the monitor points (green) to monitor the local deformation state.

Let $\boldsymbol{V}_k^c = \boldsymbol{P}_k^{mc} - \boldsymbol{P}_k^{m0}, k = i_s, \cdots, i_e$ be the vectors indicating the direction of current monitor points with respect to their rest positions, and $\boldsymbol{V}_k^l = \boldsymbol{P}_k^{mc} - \boldsymbol{P}_k^{ml}, k = i_s, \cdots, i_e$ be the vectors indicating the direction of current monitor points with respect to the monitor points at previous time step. At every time step, the dot products $\boldsymbol{V}_k^c \cdot \boldsymbol{V}_k^l, k = i_s, \cdots, i_e$ are calculated and the number of cases when $\boldsymbol{V}_k^c \cdot \boldsymbol{V}_k^l < 0$ is counted. If the count is larger than a threshold N_c (a good default value would be 0.5(*e*-*s*)), most of monitor points at the current time step are moving in the opposite direction with respect to the monitor points at previous step. Therefore, the monitor points at previous step are a good choice for the reference configuration of wrinkles (Figure 4-11). Next, the

monitor points in previous step are stored as $\boldsymbol{P}_i^{w0}, i = 1, \cdots, n$. However, the interaction between tissues also ought to be considered, since wrinkles appear when the meniscus is squeezed by the tool above as well as by the tissues below. Thus, a threshold N_s is set such that, the dot product $\boldsymbol{V}_k^c \cdot \boldsymbol{V}_k^l$ is calculated only if the number of collision points is greater than N_s. Besides, a threshold N_e is set for the ending of wrinkle. If the number of collision points is less than N_e, the addition of wrinkles will stop and $\boldsymbol{P}_i^{w0}, i = 1, \cdots, n$ will be cleared.

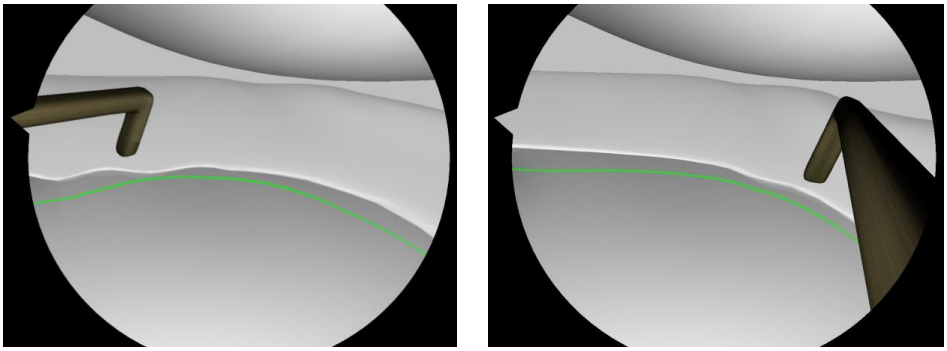


Figure 4-11 Green points show different reference configurations of the wrinkles due to the different instrument-to-tissue and tissue-to-tissue interactions.

The process of finding and removing reference configuration is implemented at each time step after the deformation of hexahedral mesh and before the visualization of surface of the meniscus, which is summarized in the following pseudo code.

if *numberOfCollideWithLowerSurface* $> N_s$ *&& collideWithInstrument=true && needWrinkles=false then*

Select $\mathbf{P}_k^{mc}, k = i_s, \cdots, i_e$ *according to the instrument-meniscus contact point*

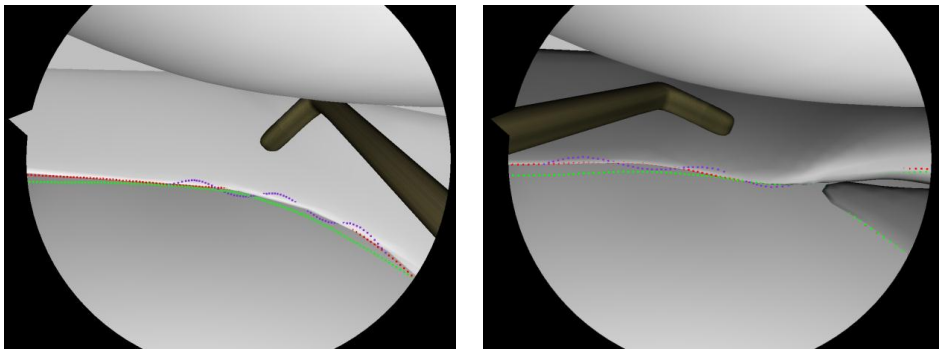
for $k = i_s, \cdots, i_e$

if $\mathbf{V}_k^c \bullet \mathbf{V}_k^l < 0$ *then*

```
        count++
      end if
    end for
      if count >N_c then
        needWrinkles ← true
        P_i^{w0} ← P_i^{ml} ,i = 1,⋯,n
      end if
end if
if numberOfCollideWithLowerSurface < N_e && collideWithInstrument=false then
   needWrinkles ← false
   wrinkleReferencePoints ←NULL
end if
if needWrinkles = true then
   implement the wrinkling algorithm
end if
```

4.4.2 A function-based method to generate the meniscus wrinkles

With reference to Figure 4-12 and Figure 4-13 and based on the reference configuration found in the previous section, the monitor points are fluctuated to generate wrinkles points. After that, the displacements between wrinkle and monitor points are propagated by the aforementioned relation in Section 4.3. A Gaussian function is adopted to control the attenuation of the displacements propagation.



(a) (b)

Figure 4-12 Red monitor points are displaced to the purple points to form wrinkles, before the displacements were propagated to the surface points. (a) Current wrinkle points for normal meniscus. (b) Current wrinkle points for torn meniscus.

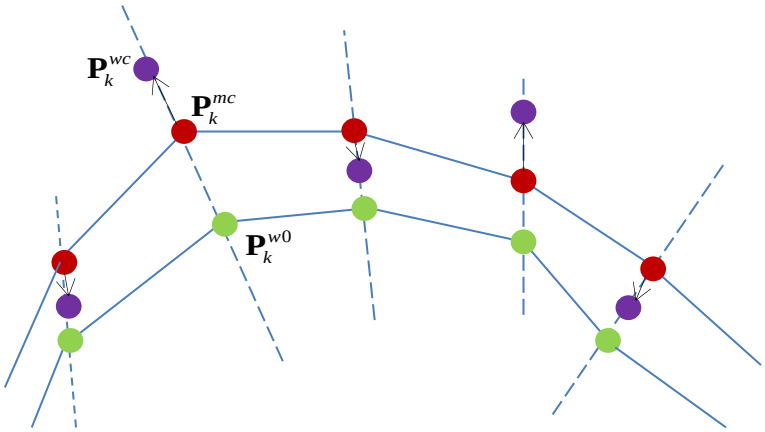


Figure 4-13 Displacements of the current monitor points. Green points are the reference position of wrinkles, red points are the current monitor points, purple points are the current wrinkle points, and black vectors are the displacements.

The displacements are calculated by a simple and efficient function-based method:

$$\mathbf{P}_k^{wc} = \mathbf{P}_k^{w0} + W(k) \cdot \frac{\mathbf{P}_k^{mc} - \mathbf{P}_k^{w0}}{\left\| \mathbf{P}_k^{mc} - \mathbf{P}_k^{w0} \right\|}, k = i_s, \cdots, i_e \qquad (4\text{-}16)$$

where *W(k)=D(k)M(k)* is a variable function for wrinkles, in which *D(k)* is a function controlling direction, and *M(k)* is a function controlling magnitude. At every time step, the current monitor points $\mathbf{P}_i^{mc}, i = 1,\cdots,n$ are firstly calculated by the finite element method and formula (4-1). Secondly, *W(k)* is calculated. Subsequently, the current wrinkle points $\mathbf{P}_k^{wc}, k = i_s,\cdots,i_e$ are calculated by formula (4-16), i.e. by adding *W(k)* along the direction from the wrinkles reference configuration $\mathbf{P}_k^{w0}$ to the current monitor points $\mathbf{P}_k^{mc}$.

For the direction *D(k)*, the accumulated chord length of $\mathbf{P}_k^{w0}, k = i_s,\cdots,i_e$ are calculated as

$$\mathbf{L}_k = \sum_{j=i_s+1}^{k} \left(\mathbf{P}_j^{w0} - \mathbf{P}_{j-1}^{w0}\right), k = i_s+1,\cdots,i_e$$

Then, $\mathbf{P}_k^{w0}, k = i_s,\cdots,i_e$ are parameterized to be within the interval [0, θ^d] by

$$\theta_{i_s+1} = 0, \quad \theta_k = \frac{L_k}{L_{i_e-1}} \times \theta^d, k = i_s+1,\cdots,i_e-1, \quad \theta_{i_e+1} = \theta^d$$

Hence, each $\mathbf{P}_k^{w0}$ is assigned a parameter θ_k while θ^d controls the frequency of wrinkle. A default value of 5π is chosen. Therefore, $D(k) = \sin(\theta_k), k = i_s,\cdots,i_e$.

For the magnitude *M(k)*, the displacement of each relevant wrinkle $m_k = \left\|\mathbf{P}_k^{mc} - \mathbf{P}_k^{w0}\right\|, k = i_s,\cdots,i_e$ are calculated. Then, the default magnitudes of wrinkles are chosen as:

$$m^d = \frac{\left(\max_{k=i_s,\cdots,i_e}(m_k) + \min_{k=i_s,\cdots,i_e}(m_k)\right)}{2}.$$

In practice, it is found that an appropriate upper bound for the wrinkle magnitude would make the wrinkles look more realistic. Let m^{up} be the upper bound of wrinkle magnitude. A filter function for the magnitude is defined as:

$$h(x) = \begin{cases} m^{up} & \text{if } x > m^{up} \\ x & \text{if } x \le m^{up} \end{cases}$$

Therefore, the magnitude for wrinkles is $\eta(s \cdot m^d)$, where *s* represents the scale factor with the default value of 1. The thickness of meniscus exerts a heavy influence on the wrinkle magnitude. Let $r_k, k = i_s,\cdots,i_e$ be the relevant thickness of monitor points, the thickness ratio for each monitor point is defined as:

$$t_k = 1 - \frac{r_k - \min_{k=i_s,\cdots,i_e}(r_k)}{\max_{k=i_s,\cdots,i_e}(r_k) - \min_{k=i_s,\cdots,i_e}(r_k)}, k = i_s,\cdots,i_e$$

When the thickness increases, the wrinkle magnitude significantly decreases. In practice, we set the cut-off thickness r^c, and define the following filter function:

$$\delta(k) = \begin{cases} 0 & \text{if } r_k < r^c \\ t_k & \text{if } r_k \ge r^c \end{cases}$$

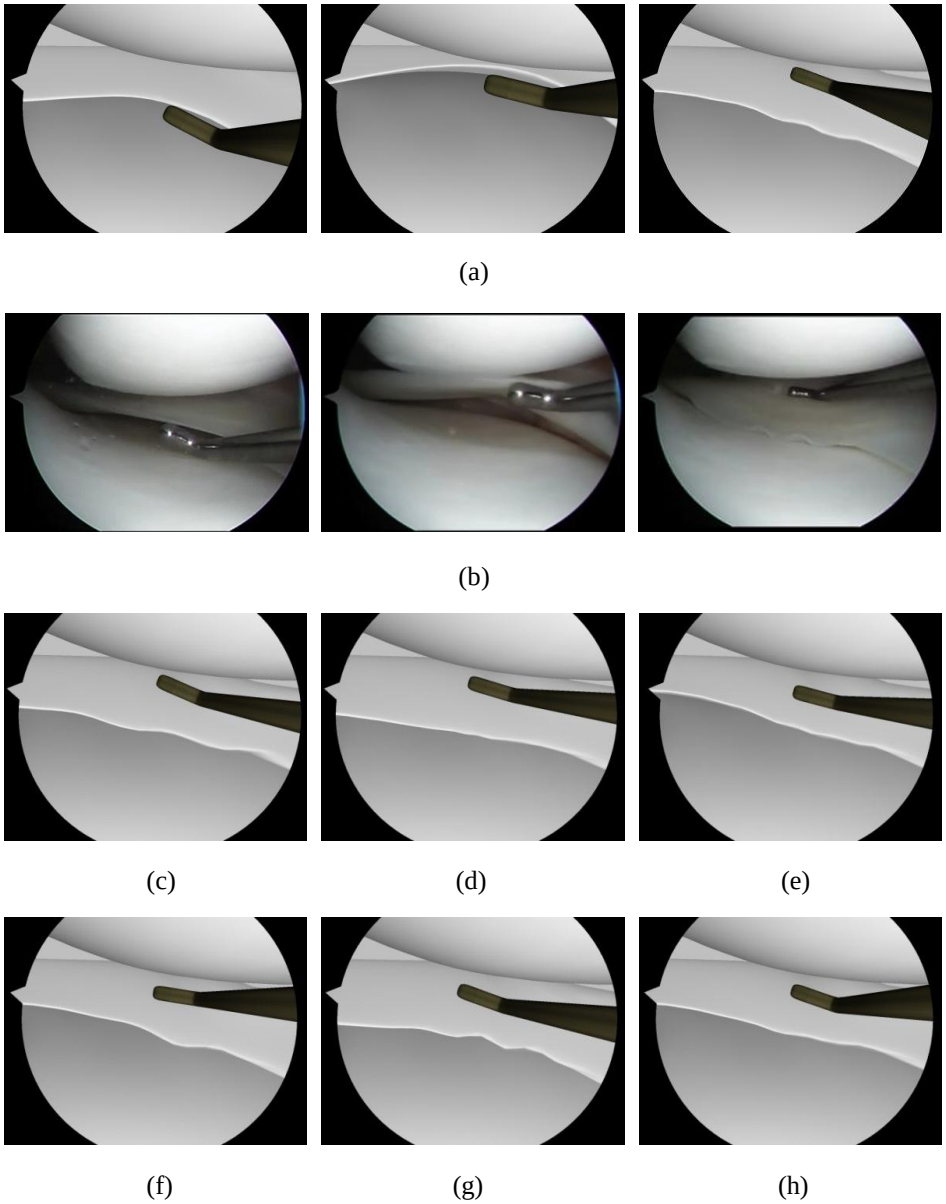
The magnitude of wrinkles can be calculated by:

$$M(k) = \delta(k) \cdot \eta(s \cdot m^d), k = i_s,\cdots,i_e$$

4.5 Experimental results and discussion

The proposed method has been implemented by utilizing the open source platform SOFA (Simulation Open Framework Architecture). For the FEM simulation, the Young's modulus of meniscus is set to 150 Mpa, and the Poisson's ratio is set to 0.49 according to the data in [162]. The whole simulation can be run with approximately 50 FPS on a desktop computer with Intel Xeon dual core 2.27 GHz CPU and 12 GB RAM. The time for generating wrinkles is less than 1 ms. Compared with the data-driven method [119] which requires the pre-storage of 3.33 MB data, this method requires only 12.9 KB of the monitor and region points. With reference to Figure 4-14, the simulated deformations (see Figure 4-14(a)) are compared side by side with the actual deformations (see Figure 4-14(b)). Furthermore, the deformation of wrinkles can be easily tuned by modifying parameters without regenerating the off-line simulation data. Figure 4-14(c) and Figure 4-14(d) demonstrate various wrinkle shapes formed by different parameters in formula (4-16). However, some drawbacks also exist in this method. Although the deformation is visually plausible for the surgical training purpose, the wrinkle positions may not be the same as in real meniscus. Thus this method has a common drawback as the data-driven method proposed in [119], the quantitative errors may be large compared with the real tissues *in vivo*. This is a key difficulty of nearly all enriching methods reviewed in Chapter 2. In addition, although some default values are provided, the parameters for the wrinkles generation may still require some adjustment, which requires some time. There is a need to mention that,

theoretically the proposed function enhanced method is independent from the basic deformation method. Other basic deformation methods may also work well with the proposed function enhanced method, which will be investigated in the future works.



(a)

(b)

(c) (d) (e)

(f) (g) (h)

Figure 4-14 (a) Wrinkles generated by the proposed method. (b) Snapshot from the real surgical video. (c-h) Various wrinkle shapes formed by different parameters: (c) Large e-s value. (d) Small e-s value. (e) $\theta^d = 7\pi$ (f) $\theta^d = 3\pi$ (g) Large magnitude. (h) Small magnitude.

4.6 Summary

In this chapter, a novel method to generate local detailed deformation has been proposed. where the coarse simulation state of low hexahedral mesh can be monitored by the embedded points. The reference configuration of wrinkles was dynamically defined according to the simulation state. The advantage of the proposed method is that the coarse simulation has been enriched with the wrinkles of meniscus by a tunable function. The experimental results have proved the efficiency of this method and verified the proposed hypothesis for the deformation. Compared with existing works, the proposed method requires less storage. The research goal of developing a local detailed deformation model has been accomplished and more details on the implementation will be discussed in Chapter 6.

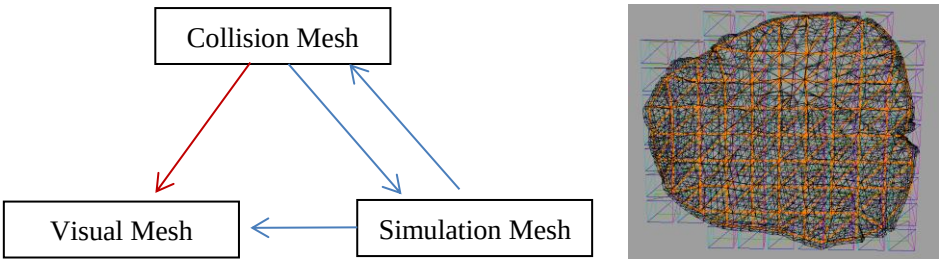
Chapter 5 Interactive Cutting of Thin Deformable Tissues

The proposed hypothesis for the cutting simulation will be verified in this chapter. Section 5.1 describes the framework of the cutting simulation. The generation of the visual-collision binding is presented in Section 5.2, while the updating of the binding during the cutting simulation is illustrated in Section 5.3. The experimental results are presented and discussed in Section 5.4, followed by the summary given in Section 5.5. The chapter was published in [163].

5.1 Overview of the framework of cutting simulation

Spillmann et al [140] proposed an efficient and robust framework for the simulation of thin volumetric deformable objects. However, this framework does not support cutting. This chapter improves this framework by adding the visual-collision binding, such that the extended framework can handle topological changes. With reference to Figure 5-1, the extended framework is composed of three different models for visualization, collision, and simulation. The red arrow between the collision mesh and visual mesh illustrates the visual-collision binding which is this chapter's contribution. A high resolution triangular mesh is used for the rendering, while a coarse triangular mesh with a certain thickness is used to represent the inner part of thin objects. Finally, a coarse hexahedral mesh is used for the simulation. The collision detection and response computation are handled by the collision mesh, while the force is

mapped to the hexahedral mesh. The resolution of the hexahedral mesh should be close to the resolution of the collision mesh, since lower resolution could not resolve all the forces from the collision detection, while a higher resolution would waste the computational time. The physically-based simulation is performed on the hexahedral mesh, and the deformation is mapped to both the collision and the visual meshes. Here the same co-rotational linear FEM as described in Section 4.2 is used. There is a need to mention that the cutting method proposed in this chapter is independent from the simulation methods. Other methods may also be utilized which will be investigated in future works.



(a) (b)

Figure 5-1 Framework of the cutting simulation. (a) Three different meshes and their relations. (b) An example of a steak model in wireframe mode. A low-resolution triangle mesh is used as a collision mesh (orange), a high-resolution triangle mesh is used for visualization (black), while a low-resolution hexahedral mesh is used as a simulation mesh. The hexahedra are visualized by different colors on their faces.

The proposed research hypothesis is that by adding the visual-collision binding (indicated as the red arrow in Figure 5-1), the thin deformable volumetric objects with different resolutions can be cut consistently and interactively. The remainder of this chapter will describe how to build

visual-collision binding, and how the binding interact with the other components in the extend framework for the cutting simulation.

The cutting is handled by two phases: pre-processing and simulation phase. In the pre-processing phase, visual-collision binding is created by using the geodesic paths on the visual mesh and triangle splitting, which will be described in detail in Section 5.2.

In the simulation phase, the cutting surface (blade) is generated from the movement of the cutting tool between the two successive time steps (e.g. the surface generated by sweeping the cutting edge of a scalpel). The cutting process is modelled based on the intersections between the collision mesh and the cutting surface. For most of the applications, the cutting process is triggered immediately at the moment when intersections happen, while in some cases, the cutting process could be started under some conditions (e.g., the scissor's blades are close enough). After the introduction of cutting on the collision mesh, the hexahedral mesh adapts to it by duplicating the elements. The relation between these models is updated, including the embedding between the collision and simulation meshes (collision-simulation embedding), the embedding between the visual and the simulation meshes (visual-simulation embedding), and the visual-collision binding, respectively. The detail of simulation phase will be presented in Section 5.3

5.2 Pre-processing phase

The pre-processing phase is run before the simulation, and the processing time is not critical. In the pre-processing phase, the visual-collision binding is built.

Given a visual mesh of a thin volumetric object, the collision mesh is built interactively using 3dsMax [164]. Automatic generation of reduced collision mesh will be done in the future work. As long as the collision mesh is built, the visual-collision binding will be calculated automatically. To keep the consistency in the cutting operation, a necessary step is to bind the fine visual mesh to the coarse collision mesh. It has to be such that when a triangle on the collision mesh is split by the cutting trajectory, the visual triangles bound to the collision triangle will be cut by the same cutting path. A visual vertex is either bound to the vertex, the edge or the triangle of collision mesh.

5.2.1 Bind visual vertices to collision vertices

For each vertex of the collision mesh, the closest visual vertices are found by searching on the visual mesh (see Figure 5-2). If the vertex is on the border of the collision mesh, the closest vertex on the visual mesh with respect to this vertex is found. If the vertex is inside the collision mesh, the closest vertex along the normal built on this vertex to both sides of the visual mesh is found. Since the number of the visual vertices is larger than the number of the collision vertices, after this step every collision vertices have their bound visual vertices. A necessary constraint here is that one visual vertex cannot be bound to two collision vertices.

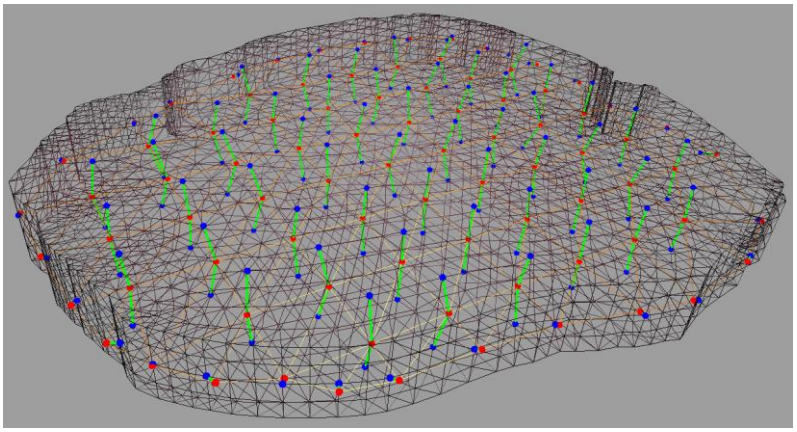
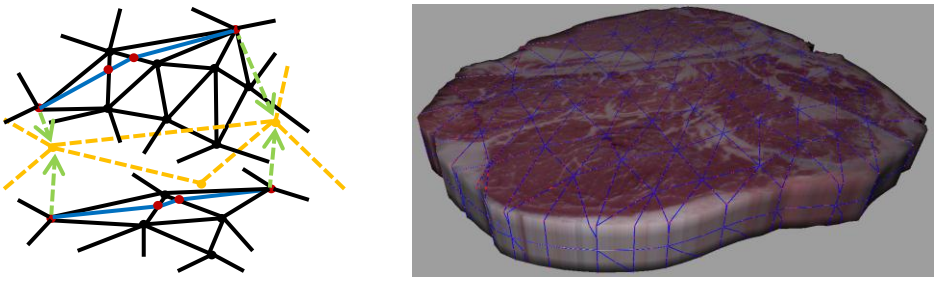


Figure 5-2 Visual vertices are bound to collision vertices. The visual mesh is black, while the orange one is the collision mesh. The red vertices are collision vertices, the blue vertices are visual vertices, and the green lines indicate their connections. Each collision vertex on the border is paired with one visual vertex. Each inside collision vertex is paired with two visual vertices: one is on the positive normal direction while another one is on the negative normal direction.

5.2.2 Calculate geodesic paths between the bound visual vertices

For each edge of the collision mesh, its two end vertices are paired with the visual vertices, respectively. Since each collision edge is a straight line, i.e. the shortest path between two collision vertices, the shortest path between its two paired visual vertices should be found. An ideal choice for this purpose is the geodesic path, since it is the shortest path on the surface between two vertices. The geodesic paths between these vertices on the visual mesh are calculated, according to the edges of the collision mesh (see Figure 5-3). There are many existing methods for the calculation of a geodesic path. The method proposed by Xin et.al [165] is chosen, since it calculates the exact geodesic path that

guarantees the non-intersection between each path. Although all these geodesic paths are not intersected with each other, they may have common points.



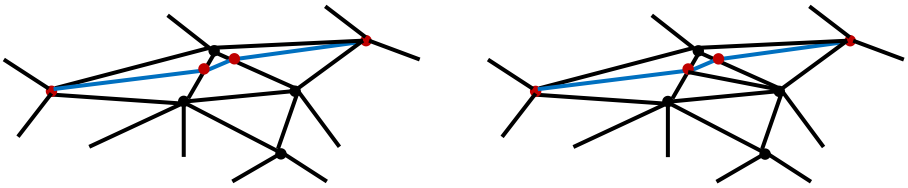
(a) (b)

Figure 5-3 Geodesic paths are calculated between the visual vertices that are bound to collision vertices. (a) The orange lines are the edges on the collision mesh, while blue lines are its relevant geodesic paths. The red points are the intersections between the geodesic paths and the visual mesh. The green arrows indicate the binding direction. Each collision edge is related with one or two geodesic paths. (b) The geodesic paths for all the collision edges are visualized on top of the original visual mesh.

5.2.3 Re-meshing of the visual mesh

With reference to Figure 5-4, new visual vertices along all the geodesic paths are added. Then, the triangles that are intersected by these new visual vertices are split, and the whole visual mesh is re-meshed. The result is a new surface mesh that is geometrically identical to the original visual mesh. All the visual vertices on the paths are bound to the edges on the collision mesh, respectively. The visual mesh is divided into several regions with respect to the triangles on the collision mesh (collision triangles). For all the visual vertices that have not yet been assigned to the collision vertices and edges, they are assigned to the collision triangles. This is achieved by the local Depth First Search (DFS)

algorithm based on the topology of visual mesh. After this step, all the visual vertices are bound to either vertices or edges or triangles on the collision mesh (Figure 5-5).



(a) (b)

Figure 5-4 The visual triangles intersected by the geodesic paths are split. (a) Some of the visual triangles are intersected by the geodesic paths. The light blue lines are geodesic paths, while the red points are the intersections points between the geodesic paths and the visual edges. (b) All the visual triangles intersected by the geodesic paths are split, and a new visual mesh is generated.

5.2.4 Assigning local parameters to the visual vertices

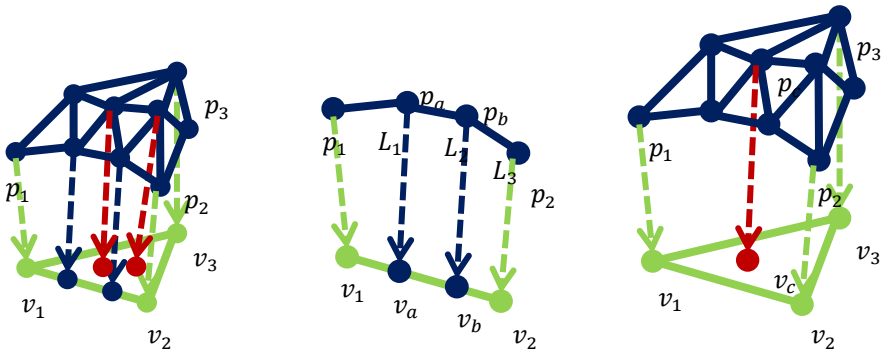
Local parameters are assigned to all the visual vertices. For each visual vertex that is bound to a collision edge, its parameter is assigned by their accumulated length, which can reflect the distances between the neighboring vertices. With reference to Figure 5-5(b), suppose visual vertices p_1 and p_2 are bound to the collision vertices v_1 and v_2, respectively, and p_a and p_b are bound to the collision edge (v_1, v_2). Let L_1, L_2 and L_3 be the lengths of the visual edges (p_1, p_2) in the reference configuration (p_a, p_b) and (p_b, p_2). Then the binding parameters for p_a and p_b are $(\frac{L_1}{L_1+L_2+L_3}, 1-\frac{L_1}{L_1+L_2+L_3})$ and $(\frac{L_1+L_2}{L_1+L_2+L_3}, 1-\frac{L_1+L_2}{L_1+L_2+L_3})$, respectively. With the above parameters, we can calculate the

projected points v_a of p_a as $v_a = \frac{L_1}{L_1+L_2+L_3} * v_1 + \left(1 - \frac{L_1}{L_1+L_2+L_3}\right) * v_2$, and v_b of p_b as $v_b = \frac{L_1+L_2}{L_1+L_2+L_3} * v_1 + \left(1 - \frac{L_1+L_2}{L_1+L_2+L_3}\right) * v_2$. For each visual vertex that is bound to a collision triangle, it is parameterized by the mean value coordinates [166], since this parameterization has a good quality and is widely used in many geometric applications. With reference to Figure 5-5(c), suppose visual vertex p_c is bound to the collision triangle (v_1, v_2, v_3). Let (a,b,c) be the binding parameter of p_c, then the projected point v_c of it can be calculated by

$$v_c = a * v_1 + b * v_2 + c * v_3 \quad (5\text{-}1)$$

Since each visual vertex that is bound to the collision vertex is shared by its one-ring surrounding collision triangles, the parameter of this visual vertex is defined by the specific collision triangle. For examples, in Figure 5-6 the parameter of v_1 in the triangle (v_1, v_2, v_3) is (1,0,0) while the parameter of v_1 in the triangle (v_5, v_6, v_1) is (0,0,1). Similarly, since each visual vertex that is bound to the collision edge is shared by two collision triangles, the parameter of this visual vertex is then defined by the specific collision triangle. With reference to Figure 5-6, suppose *v* is a projected point on the edge (v_1, v_2) with the parameter (a, b). Then for the collision triangle (v_1, v_6, v_2), the parameter of *v* is (a, 0, b), while for the collision triangle (v_1, v_2, v_3), the parameter of *v* is (a, b, 0). Thus, the local parameters of the visual vertices that are bound to the collision edges will be defined in the simulation phase according to the specific collision triangle. Therefore, for each given collision triangle, the coordinates of

the visual vertices bound to it can be uniformly written as in Formula 5-1, no matter what the visual vertices are bound to the collision vertices, edges, or triangles. In the above, only the visual vertices in the positive normal direction (positive visual vertices) of the collision triangles are discussed. It can be handled similarly for all the visual vertices in the negative normal direction (negative visual vertices) of the collision triangles. In Figure 5-7, the parameterization is visualized on top of the collision triangle. All the parameters are needed to be calculated only once at the pre-processing phase. In the simulation phase, the parameters of the new visual vertices can be interpolated or transformed from the parameters of the original visual vertices.



(a) (b) (c)

Figure 5-5 All the visual vertices are bound to the collision triangles. (a) The orange triangle is the collision triangle, while the blue triangles are visual triangles. The arrows indicate the binding direction: green for the visual vertices bound to the collision vertices, blue for the visual vertices bound to the collision edges, and red for the visual vertices bound to the collision triangles. (b) The calculation of the binding parameters for the visual vertices bound to the collision edges. (c) Visual vertex $\boldsymbol{p_c}$ is bound to $\boldsymbol{v_c}$ on the collision triangle.

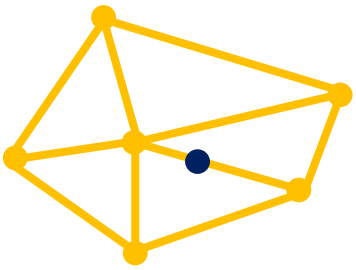
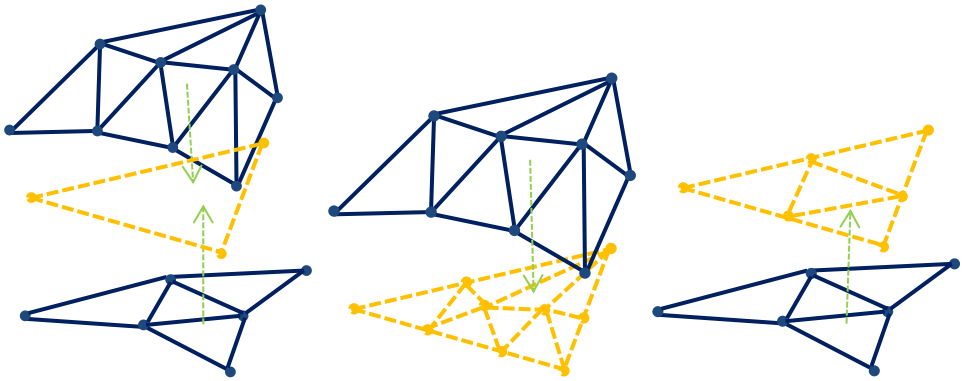


Figure 5-6 The definition of local parameters for the visual vertices bound to collision edges. Visual vertex *v* is bound to collision edge ($\mathbf{v_1}$, $\mathbf{v_2}$), its parameters in collision triangle ($\mathbf{v_1}$, $\mathbf{v_6}$, $\mathbf{v_2}$) and ($\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$) are different.



(a) (b) (c)

Figure 5-7 Local parameterization of the visual vertices. (a) Blue visual vertices are bound to the orange collision triangle, including the visual vertices on the positive and negative normal directions of this triangle. The green arrows indicate the binding direction. (b) Each visual vertex on the positive normal direction is parameterized. The parameterization is visualized by overlapping with the collision triangle. (c) Each visual vertex on the negative normal direction is parameterized.

5.3 Simulation phase

In the simulation phase, the collision and visual meshes are cut locally and the visual-collision binding is updated accordingly. The cutting simulation includes the following steps:

1. *Cutting of collision mesh*. The cutting blade is intersected with the collision mesh. The triangles are split along the path of the blade. The branches of the collision mesh contained in the elements of the simulation mesh are analyzed. The elements are duplicated for each branch. The collision-simulation embedding is updated.

2. *Cutting of visual mesh*. Based on the visual-collision binding, the visual mesh is cut in the rest configuration using local 2D coordinates. Also, the triangle mesh for the cut surface is generated. Then the visual-collision binding is updated.

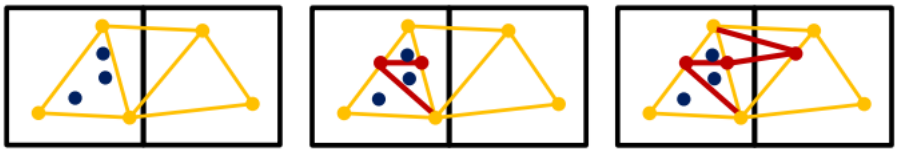
3. *Updating the visual-simulation embedding*. According to the visual-collision binding, the visual-simulation embedding is updated.

5.3.1 Cutting the collision mesh and updating the collision-simulation embedding

The cutting of the collision mesh is performed by the method of elements splitting that is similar to works [144-146]. The intersections between the triangles of swiping blade and the edges of the collision mesh are detected. If two edges of the collision triangle are intersected with the blade, the triangle

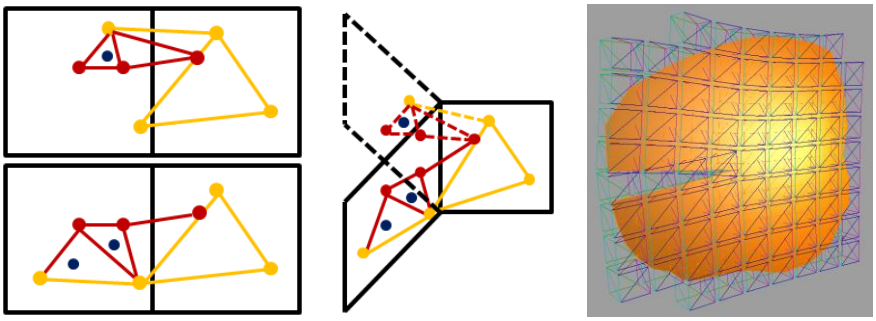
will be split into three triangles (Figure 5-8). A vertices snapping strategy similar to [148] is utilized to avoid ill-shaped triangles for collision mesh.

After cutting of the collision mesh, a method that is similar to the method published in [154, 159] is applied to update the hexahedral mesh to adapt to the cut (Figure 5-9). Elements analysis are performed on the relevant hexahedra. The hexahedral mesh is adapted to the cut by duplicating the elements of the hexahedra, such that for each branch within the original hexahedron, a copy of the hexahedron is made. The hexahedra are connected according to the branch analysis, which is performed based on the topology of the new collision mesh. Since the cutting is locally performed, the branch analysis is also performed locally on a few hexahedra. The advantages of using hexahedral mesh include: (1) It is easier to do the branching and analyze the connection between neighboring elements thanks to its regular structure (as against to the unstructured tetrahedral mesh); (2) Since each new created element is a copy of the original element, the cutting process will not generate ill-shaped elements. Therefore, the stability of the simulation is preserved.



(a) (b) (c)

Figure 5-8 The process of cutting the collision mesh. The hexahedra are illustrated in 2D as the quadrangles with black edges; the orange edges are the collision edges; the orange points are the collision vertices; the red points and lines are the cutting points and paths respectfully; the dark blue points are the visual vertices bound to the collision triangle. (a) The mesh before cut. (b) A triangle is cut by two intersection points and split into three triangles. Since the first hexahedron still has only one branch, the hexahedron remains unchanged. (c) The second triangle is cut. The first hexahedron needs to be duplicated, because it has two branches.



(a) (b) (c)

Figure 5-9 Duplication of the hexahedron. The red points and lines are the new added collision vertices and edges as the result of the cut. (a) Duplication of the hexahedron such that each copy has one branch of the collision vertices. (b) Each copy of the hexahedron is connected with its surrounding hexahedra according to the topology of the collision mesh. (c) Snapshot of simulation of cutting the collision mesh.

5.3.2 Cutting the visual mesh

Based on the visual-collision binding which is built on the pre-processing phase, the cutting of the visual mesh is performed in the rest configuration of the collision mesh (see Figure 5-10 and Figure 5-11). With reference to Figure

5-10(a), suppose p_1, p_2 and p_3 are the coordinates in the rest configuration, and they are bound to the collision vertices v_1, v_2 and v_3, respectively. Now that a cut is introduced to this collision triangle (v_1, v_2, v_3) (see Figure 5-10(b)), vertices v_1, v_2 and v_3 define a plane. 2D coordinates are assigned to these vertices in such a way that it keeps the angles between the edges (v_1, v_2), (v_2, v_3) and (v_1, v_3). The lengths between the above three vertices are also kept (see Figure 5-11(a)). Then, all the visual vertices are transformed onto this plane and their 2D coordinates are obtained. This is achieved by Formula 5-1 since all the binding parameters of the visual vertices have already been assigned in the pre-processing phase. Two cut points on the collision triangle are also transformed in the same way (see Figure 5-11(b)). The intersection points between the cut path and the visual edges are then calculated by their 2D coordinates on this plane. With reference to Figure 5-11(c), a standard method [167] is applied to compute the intersection points between the cutting lines and the visual edges. All the parameters of the intersection points are then calculated by interpolating from the parameters of the end points of the edge. For examples, in Figure 5-11(c), suppose v_3 and v_a are two of the transformed visual vertices. After the computation of intersections, the parameters (c_1, c_2) for the intersected point v are obtained such that $v = c_1 * v_3 + c_2 * v_a$. Then, an intersected point p is generated on the visual mesh by $p = c_1 * p_3 + c_2 * p_a$ (see Figure 5-10(d) and Figure 5-11(c)). After that, all the new visual points are added to the original visual mesh. The visual

triangles that are intersected by these new visual edges are split and re-meshed (Figure 5-10(e)). Therefore, a new visual mesh after cut is obtained.

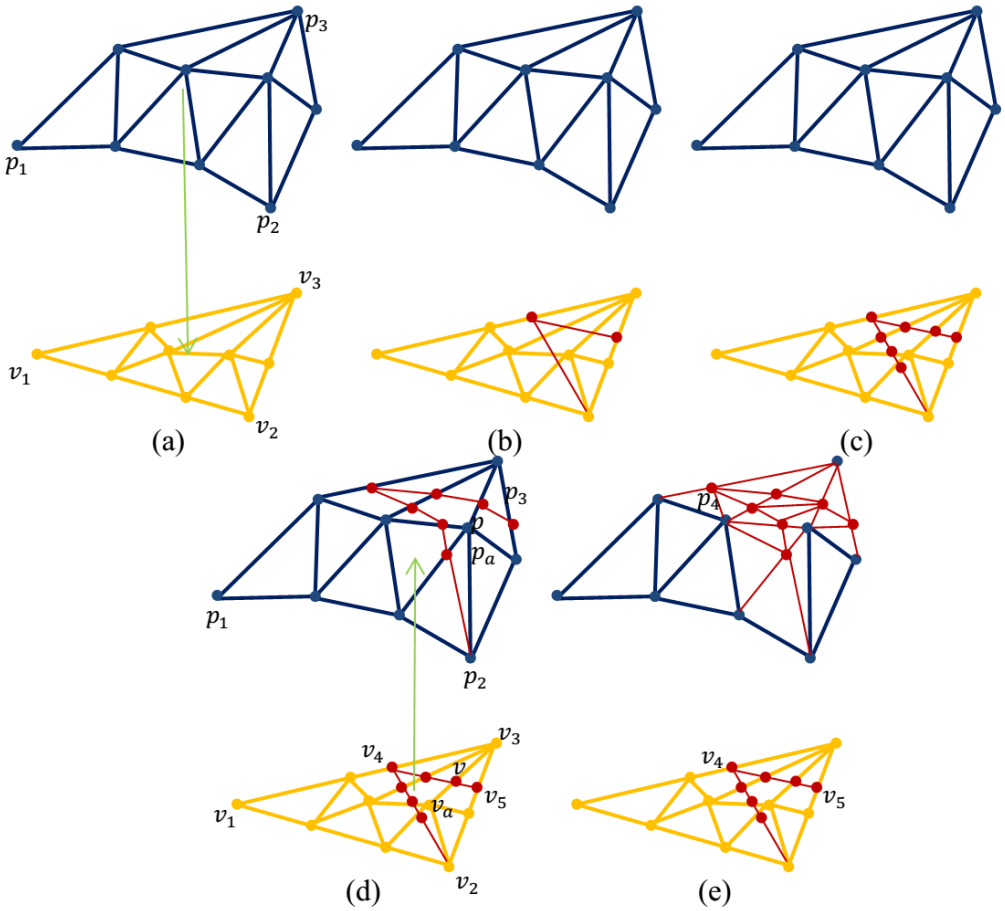


Figure 5-10 The process of cutting the visual mesh. The dark blue points are the visual points; while the orange points are their parameters on the collision triangle in rest configuration. (a) The visual vertices have already been bound to the collision triangle in the pre-processing phase. The green arrow shows the binding direction. (b) The collision triangle is cut by the blade surface. (c) The intersection points are calculated. (d) New visual vertices are added according to the intersection points on the collision triangle. The green arrow shows the mapping direction. (e) The visual triangles are re-meshed to adapt to the new visual vertices.

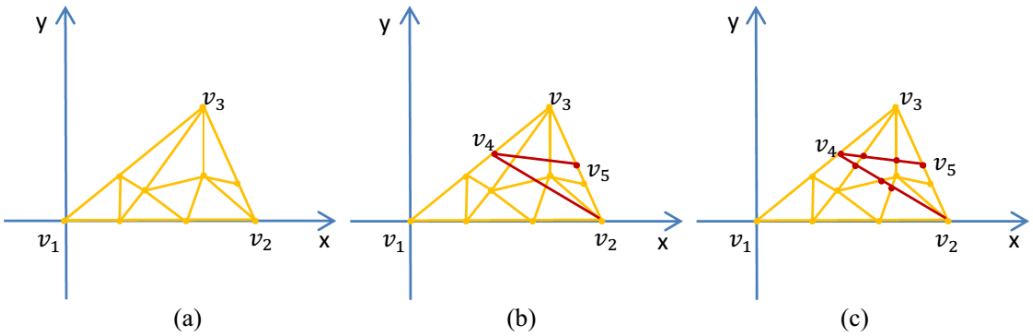


Figure 5-11 Calculation of the intersection points on the plane defined by the collision triangle in the rest configuration. The orange points are the parameters of the visual vertices, while the red points are the intersection points. (a) The 2D coordinates are assigned for the visual vertices bound to the collision triangle that is cut. (b) The cut points are transformed to the rest configuration and assigned 2D coordinates. (c) The intersection points are calculated.

5.3.3 Updating the visual-collision binding

After the cutting, the visual-collision binding needs to be updated for all the new collision triangles. With reference to Figure 5-11(b), suppose the original collision triangle (v_1, v_2, v_3) is split into three new collision triangles (v_1, v_2, v_4), (v_2, v_5, v_4) and (v_3, v_4, v_5). For the binding of these collision vertices, v_4 is taken for example (see Figure 5-10(e)). The new added visual vertex p_4 is bound to it. The updating of its binding parameter is not necessary, since the parameter is only defined by the specific collision triangle. For the binding of the intersection points, v is taken for example (Figure 5-10(d)). Its relevant new added visual vertex p is bound to the collision edge (v_4, v_5), the binding parameter of which is calculated by the accumulated length as in the pre-processing phase. For the binding of the original visual vertices that are bound to the collision triangle (Figure 5-10(d) and Figure 5-11(c)), v_a is taken for example. Here, the binding parameter of v_a with respect to the collision triangle (v_1, v_2, v_3) is already known. Let this parameter be $(\mathrm{a}, \mathrm{b}, \mathrm{c})$. The

purpose is to calculate the binding parameters of v_a with respect to the collision triangle (v_2, v_5, v_4). Let this parameter be (a', b', c'). The binding parameters of the visual vertices v_2, v_4 and v_5 with respect to the collision triangle (v_1, v_2, v_3) are also known. Thus, the 3×3 matrix M is known, such as:

$$\begin{pmatrix} v_2 \\ v_4 \\ v_5 \end{pmatrix} = \mathrm{M} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Therefore the parameter are computed as

$$(a', b', c') = (a, b, c) \times \mathrm{M}^{-1} \quad (5\text{-}2)$$

However, it is not known that which new collision triangle v_a is bound to. Thus, for v_a, (a', b', c') is calculated for all the three new collision triangles by Formula 5-2. Then, v_a is bound to the collision triangle such as $0 \le a' \le 1$, $0 \le b' \le 1$ and $0 \le c' \le 1$. Finally, all the visual vertices are bound to the new collision triangles, respectively.

5.3.4 Handling the cut surface mesh

After cutting of the visual mesh, the cut surface mesh is generated to keep the visual look of the objects as a water-tight surface. Since the visual mesh in this framework will not be used for collision handling and simulation, the quality of the cut surface mesh is not very important and even ill-shaped triangles can be accepted. Therefore, there are many ways to generate the cut surface mesh for our simulation. For each cut collision edge, the cut surface is generated by connecting the positive and negative visual vertices bound to this

edge (Figure 5-12). In [100], the cut surface is generated by down-sampling the related visual vertices in order to lower the data size of the new mesh. Here the related visual vertices are up-sampled since the visual mesh in the simulation will not be involved in collision handling. This can be achieved by merging the parameters of positive and negative visual vertices that are bound to the cut collision edge. With reference to Figure 5-12(b), suppose the parameters for v_1, v_2 and v_3 are (0.7, 0.3), (0.2, 0.8) and (0.4, 0.6), respectively. After the up-sampling, the parameters for v_1, v_2 and v_3 remain unchanged, while the parameters for v_4, v_5 and v_6 are (0.4, 0.6), (0.7, 0.3) and (0.2, 0.8), respectively. After the generation, the cut surface vertices are then bound to the cut collision edge, and their binding parameters are assigned according to the relevant positive and negative visual vertices. With reference to Figure 5-12(c), for example, cut surface vertices p_1, p_2, p_3 and p_4 are assigned the same parameter (0.7, 0.3) according to the parameter of v_1.

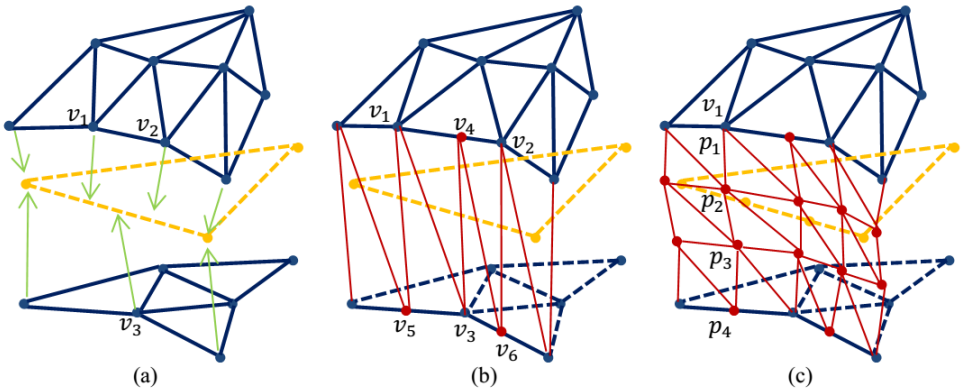


Figure 5-12 Generating the cut surface mesh. The dark blue points are the visual vertices; the orange triangle is the collision triangle; the green arrows show the binding directions; the red points are the cut surface vertices. (a) Before the generation, the positive and negative visual vertices are bound to the cut collision edge. (b) Up-sampling the positive and negative visual vertices and connecting these vertices to form a cut surface. (c) Different resolutions of cut surface can be generated by adding different resolutions of in-between visual vertices.

If the collision edges are cut again (see Figure 5-13), the cut surface is cut simply by comparing the parameters between the cut surface vertices and the cut points on the collision edge. The cut surface mesh is cut and separated by introducing new vertices according to the parameter of the cut points. All these new vertices are then bound to the new collision edges, respectively. Since the calculation of the interactions between the cut path and the cut surface mesh is not required, high resolution cut surface mesh is supported by this cutting framework.

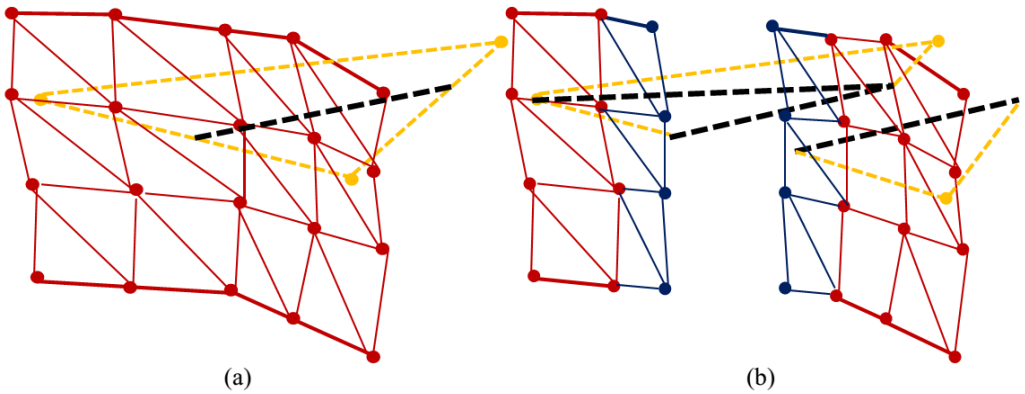


Figure 5-13 Cutting of the surface mesh. (a) The orange triangle is the collision triangle, while the black dash line is the cut path on it. The red mesh is the cut surface mesh. (b) The blue points are the new added vertices on the cut surface.

5.3.5 Updating the visual-simulation embedding

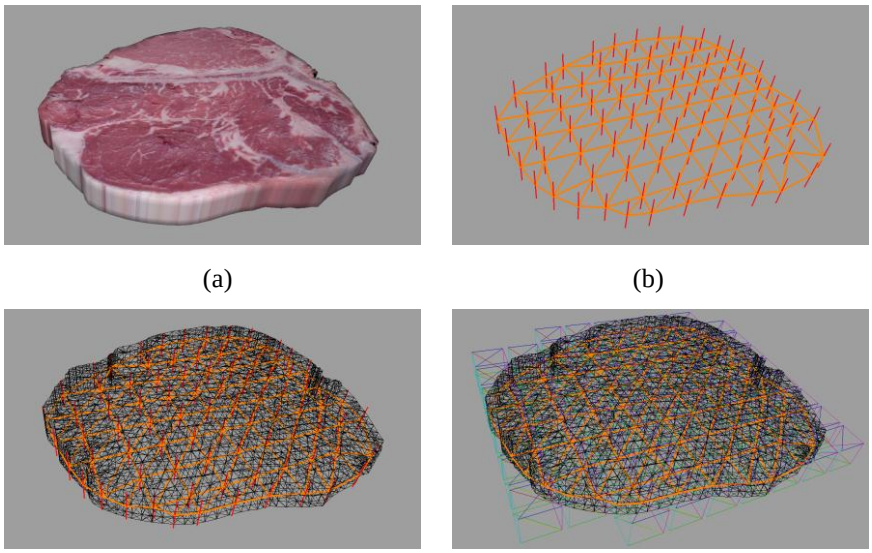
Once the collision-simulation embedding and visual-collision binding have been built, the visual-simulation embedding can be built accordingly. For each new triangle generated by cutting, the hexahedra containing their three vertices are found. Then, for all the visual and cut surface vertices bound to this triangle, the barycentric coordinates are found and calculated with these containing hexahedra. With reference to Figure 5-9, the dark blue points represent visual vertices bound to the collision triangles. After the cutting, all these visual vertices are to be re-bound to new collision triangles and then re-embedded to the hexahedra according to the collision-simulation embedding.

5.4 Experimental results and discussion

This section presents two examples, and discusses the experimental results.

5.4.1 Cutting of a steak

For the first example, the proposed cutting framework is tested by the simulation of cutting a steak. The visual mesh of the steak is used as an actual standard test provided by SOFA for testing elastic object deformations and used by authors of several papers [5, 100, 153, 168]. With reference to Figure 5-14, the visual mesh of the steak has 2,641 vertices and 5,278 triangles. The simulation is run on the hexahedral mesh with 190 DOFs and 76 hexahedra. The above visual and simulation mesh are coupled with two collision meshes of different resolutions.



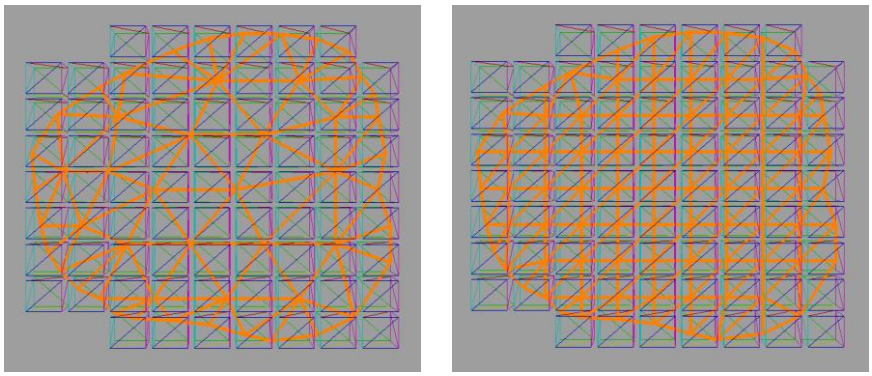
(a) (b)

(c) (d)

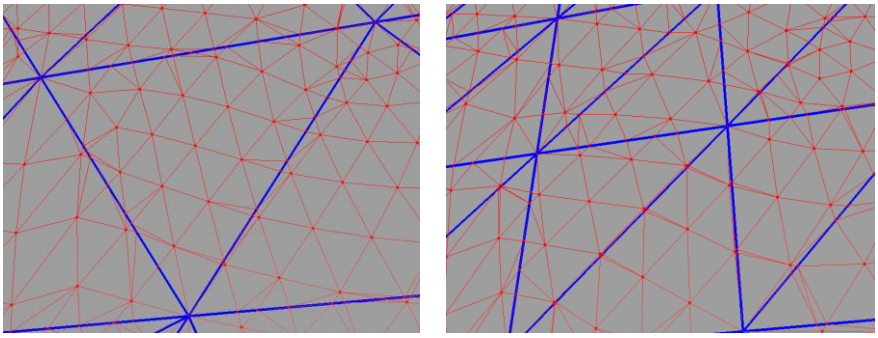
Figure 5-14 The steak model. (a) The visual model with texture. (b) The collision model. The orange mesh is the collision mesh, while the red lines are the radii for the collision vertices. (c) The visual mesh is overlapped with the collision mesh. (d) The visual mesh, collision mesh and simulation mesh are overlapped.

Figure 5-15 shows the collision and hexahedral meshes, as well as the visual collision binding, respectively. Figure 5-16 shows the cutting of a steak with different collision meshes. For the real time record of the interactive cutting of the steaks with different collision meshes, the accompanying videos can be seen at https://youtu.be/sVwq8CRdNfY. Table 5-1 collects the performance data of the implementation of the pre-processing phase. The average preprocessing time of these two models are 5.1 sec for model No. 1 and 10.2 sec for model No. 2. This is because the finer collision mesh requires more computation time to obtain the geodesic paths. Since the preprocessing is run

before the simulation loop, time is not important. In the implementation, the preprocessing is only done once and then the data is stored. After that, the data will be loaded before every simulation. Table 5-2 collects the performance data of the implementation of the simulation phase. In the simulation, the time for cutting of model No. 1 is generally longer than the time for model No. 2. This is because model No. 1 has a coarser collision mesh, and thus has more visual points mapped to each collision triangle. More visual points to handle for each cut require more time. The time for intersection detections and for visual and collision re-meshing increases slightly, while the time for updating the binding increases significantly.

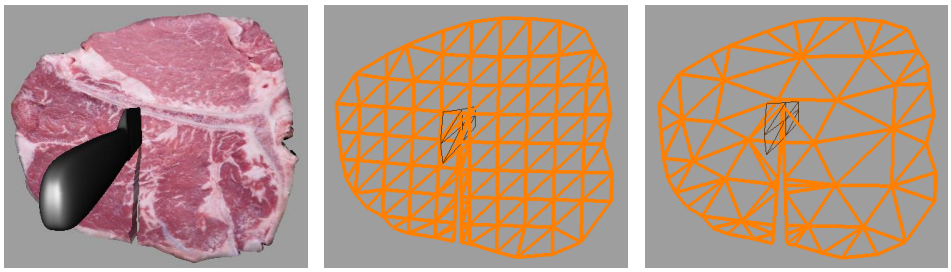


(a)



(b)

Figure 5-15 Collision meshes with different resolutions and the visual collision binding after pre-processing. (a) The orange meshes are the collision meshes. From left to right, the resolution of collision is increased while the resolutions of visual and simulation meshes are fixed. (b) The close up view of visual collision binding for different collision meshes, respectively, after the pre-processing.



(a) (b) (c)

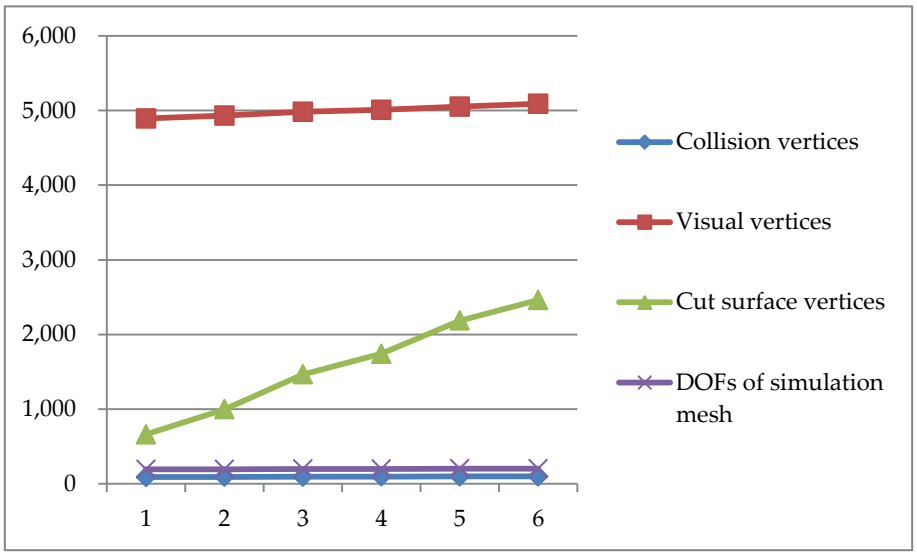
Figure 5-16 Cutting of a steak with different collision meshes. (a) The visual mesh after cut. (b) The collision mesh after cut. There are 85 vertices and 137 triangles for the collision mesh of this model. (b) There are 50 vertices and 67 triangles for the collision mesh of this model.

Table 5-1 Performance of the pre-process phase of cutting a steak.

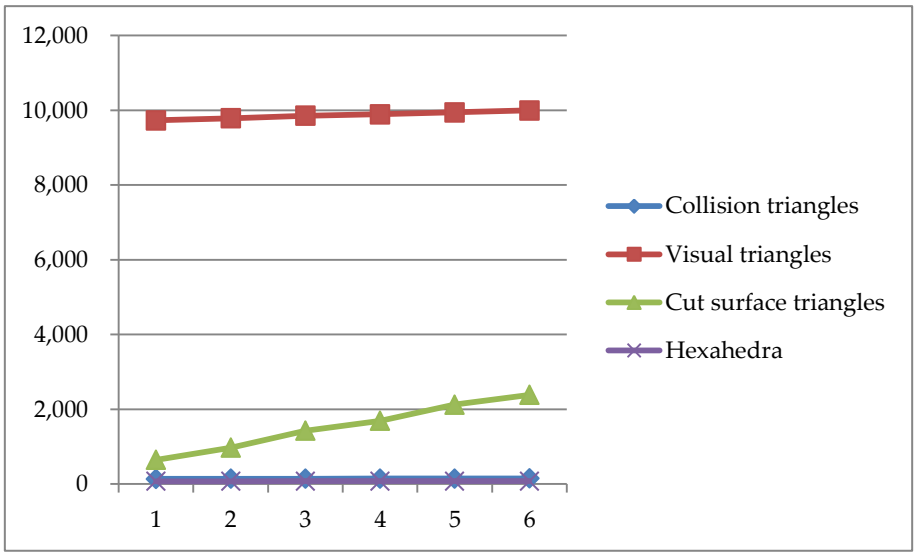
#	Collision vertices	Collision triangles	Preprocessing Time [s]	Per collision triangle after preprocessing	
				visual vertices	visual triangles
1	50	67	5.1	105	137
2	85	137	10.2	50	70

Table 5-2 Performance of the simulation phase of cutting a steak.

#	Per cut time [ms]				
	Cut and update Collision-visual binding	Update collision-simulation and visual-simulation embedding	Add surface mesh	Total cut time	Simulation time
1	2.5	15.3	3.7	17.6	2.5 - 3.5
2	2.4	5.8	3.2	9.4	2.6 - 3.8



(a)



(b)

Figure 5-17 The numbers of vertices and elements in 6 consecutive cuts. (a) The number of vertices. (b) The number of elements.

Since the cutting introduces new vertices for both visual, collision and simulation meshes, it will generally increase the simulation time. To better analyze the performance of our cutting method, the changes in 6 consecutive cuts of the No. 2 model are recorded (see Figure 5-17). Figure 5-17(a) shows the numbers of vertices of visual, collision and simulation meshes, while Figure

5-17(b) shows the numbers of elements. There is no clear difference for the simulation time between the first and the last cuts (however the time is definitely less than 1 ms). Although the number of vertices and triangles of the visual mesh increases significantly, the number of vertices and elements of collision and simulation meshes grows very slowly.

It is clear that the number of collision vertices and triangles affect the performance of the most of the collision algorithms. The very same steak model was also used in [100, 168], where the same triangle mesh was used for both visualization and collision in their cutting simulation, while in this simulation the resolutions of visual and collision meshes are of different levels of magnitude. Since the authors of [100, 168] did not provide the data of their collision meshes during cutting, the performance of this method is compared with the best scenario they could get, i.e. when the collision mesh remains unchanged. With reference to Figure 5-18, it is clear that the resolutions of collision meshes used in the previous cited works are of a different level of magnitude.

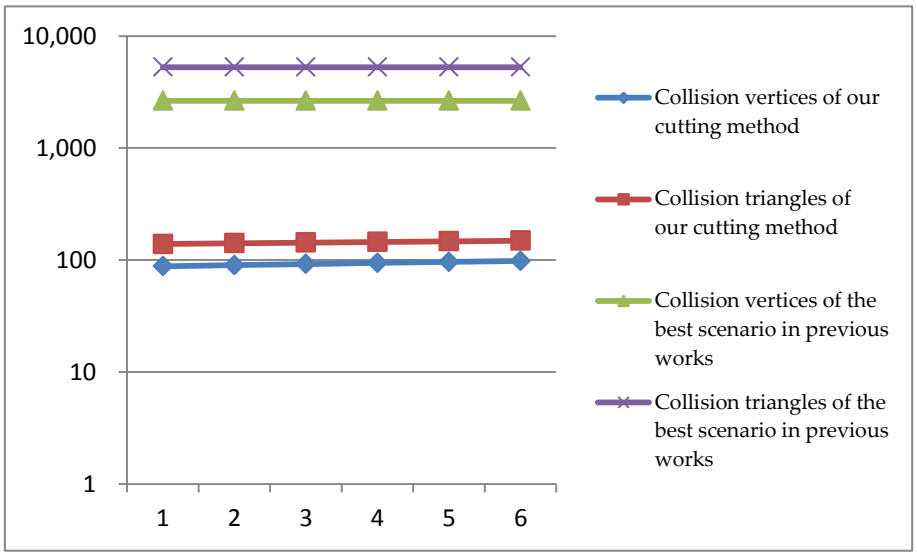


Figure 5-18 Comparison of the number of collision vertices and triangles in our method with the best scenario which could be found in [100, 168]. Data is displayed in logarithmic scale of 10.

It is difficult to compare precisely with the cutting method of Wu et al. [139], since different models were used. The collision detection of their method is relied on octree structure, while the collision handling of this thesis is independent from the simulation mesh. Their collision vertices are generated by down-sampling every 2^3th points of the visual mesh. In this way, only collision points can be utilized, while in the simulation of this thesis all the collision points, edges and triangles are available. Furthermore, there was no discussion on how well the collision vertices approximate the visual ones by this down-sampling method. Jia et al [109] have pointed out the difference between the down-sampling and the full simulation at the thin regions of the models. While in the method of this thesis, once a collision mesh with good approximation property is built before the simulation, the approximation will be

maintained after each cut. Since the new collision points generated by the cut are the interpolations from the collision points before cut. This can also be seen in the next example. Their method supports general objects, while the proposed method only supports thin volumetric object. This is a clear disadvantage of the method in this thesis, and the generalization to support all objects will be the future work.

5.4.2 Cutting of a thin object

Thin objects, such as a thin paper sheet, are also supported by the proposed cutting method. In the second example, the simulation of cutting a thin sheet is presented. In the first experiment, the resolutions of the collision and simulation meshes are fixed, and then the cutting method is tested with the visual meshes of different resolutions. There are 65 vertices and 104 triangles in the collision mesh, while there are 200 vertices and 81 hexahedra in the simulation mesh. Figure 5-19 shows the cutting process of the thin sheet. For all the models, Figure 5-20 lists the visual collision binding after the preprocessing, and the visual mesh overlapped with the collision mesh before and after cutting. The videos for the interactive cutting of different models of the thin sheet can be seen at https://youtu.be/ekX-lYfNmxA. Table 5-3 collects the performance data of the pre-processing phase of these models. These data are similar to the data in the previous example. The preprocessing time increases largely with the increase of the resolutions of the visual mesh. Table 5-4 collects the performance data of the simulation phase. The time to test intersections and re-meshing the collision and visual meshes increases slightly, while the time for

updating the visual-simulation and collision-simulation embedding increases significantly. While the resolution of its visual triangles of model No. 4 is over the magnitude of 100 K, it still can be cut interactively. This indicates that the proposed method is very flexible with respect to different computation powers. For example, just like in many computer games, different configurations of mesh resolutions could be provided for the cutting simulation on different computers. Although the simulations with all the models can be run in real time, updating of the collision-simulation and visual-simulation embedding is the bottleneck of the current implementation.

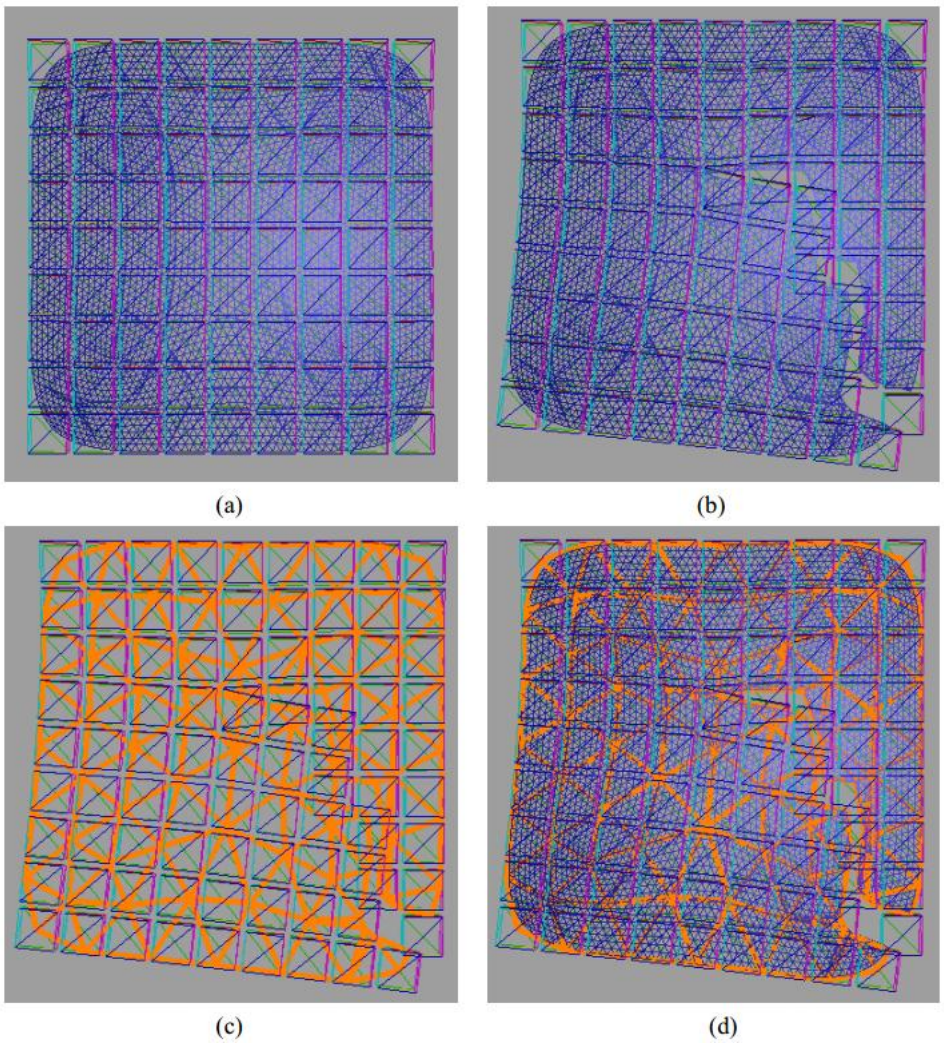


Figure 5-19 Simulation of cutting a thin sheet. (a) The visual and simulation meshes before cutting. (b) The visual and simulation meshes after cutting. (c) The collision and simulation meshes after cutting. (d) The visual, collision and simulation meshes after cutting.

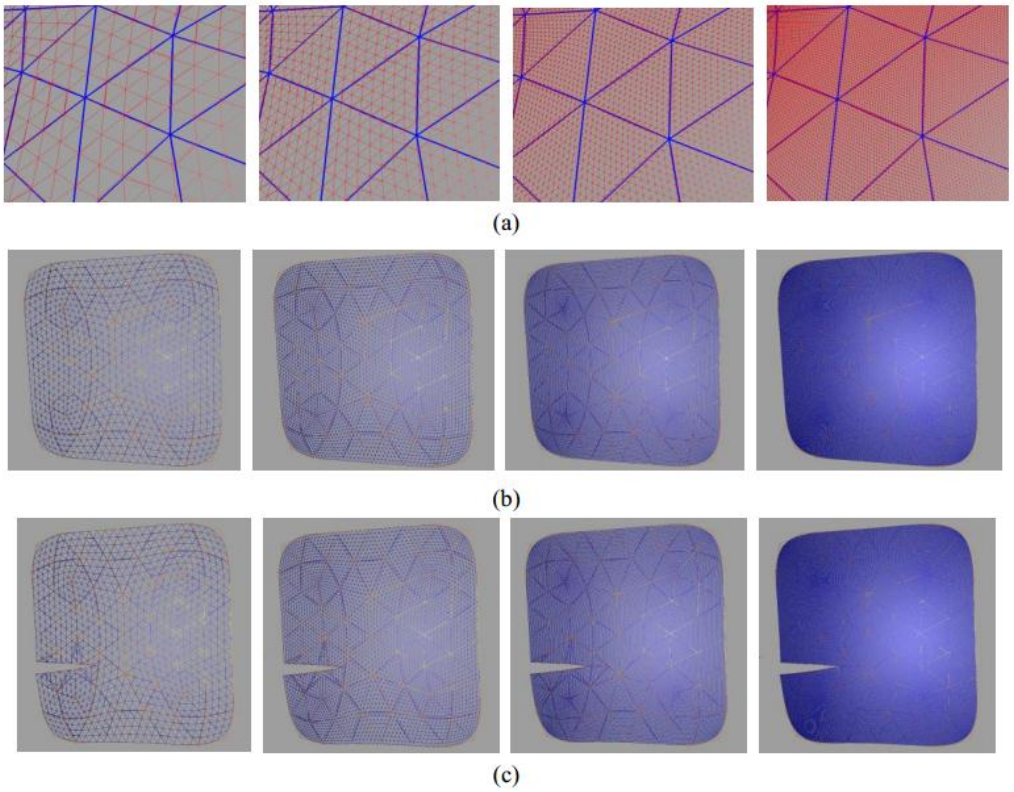


Figure 5-20 The visual mesh overlapped with the collision mesh before and after cutting. From left to right, the resolution of the visual mesh is increased while the resolution of the collision mesh remains unchanged. (a) The close up view of the visual and collision bindings after the pre-processing. (b) The collision mesh is overlapped with the visual mesh before cutting. (c) The collision mesh is overlapped with the visual mesh after cutting.

Table 5-3 Performance of the pre-processing phase in the first experiment of cutting a thin sheet.

#	Visual vertices	Visual triangles	Preprocessing time [s]	Per collision triangle after preprocessing	
				visual vertices	visual triangles
1	881	1664	0.8	26	31
2	3425	6656	7.1	73	102
3	13505	26624	57.5	214	337
4	53633	106496	422.2	688	1193

Table 5-4 Performance of the simulation phase in the first experiment of cutting a thin sheet.

#	Per cut time [ms]			
	Cut and update collision-visual binding	Update collision-simulation and visual-simulation embedding	Total cut time	Simulation time
1	0.9	9.1	10.5	2.5-3.5.
2	1.9	12.3	14.6	2.7-3.6
3	3.7	17.9	21.9	3.9-5.2
4	10.9	38.6	49.9	7.2-8.5

In the second experiment of this example, the resolution of visual mesh is fixed and then the cutting method is tested with the collision meshes of different

resolutions. There are 3425 vertices and 6656 triangles in the visual mesh, while there are 200 vertices and 81 hexahedra in the simulation mesh. Four different collision meshes are used. More information can be seen on the videos at https://youtu.be/hCG_1LuFlJU (cutting and collision handling) and https://youtu.be/3_mGMxvrPbg (cutting to separate the object). Figure 5-21 lists the visual and collision meshes after cutting. Table 5-5 collects the performance data of the pre-processing phase of these models. The resolutions of the collision mesh increase from model No. 1 to model No. 3. No pre-processing phase is required for the fourth model, since it uses the same mesh for both the visualization and collision handling. Table 5-6 collects the performance data of the simulation phase. The average time for each cut of these models decrease slightly from model No. 1 to model No. 3. All the models can run in real time when there are no collisions with the environments. However, once the collisions are introduced, all the simulation times increase rapidly. From model No. 1 to model No. 4, the simulation times increase, following the increment of the resolutions of the collision mesh. The first two models can still run in real time, model No. 3 can run interactively at 21.7 FPS, and model No. 4 only has 7.5 FPS. Figure 5-22 illustrates the changing FPS from model No. 1 to model No. 2. These data prove the flexibility of the proposed cutting method that, when the resolution of visual mesh is fixed, different resolutions of collision mesh can be chosen to meet the requirements of the simulation.

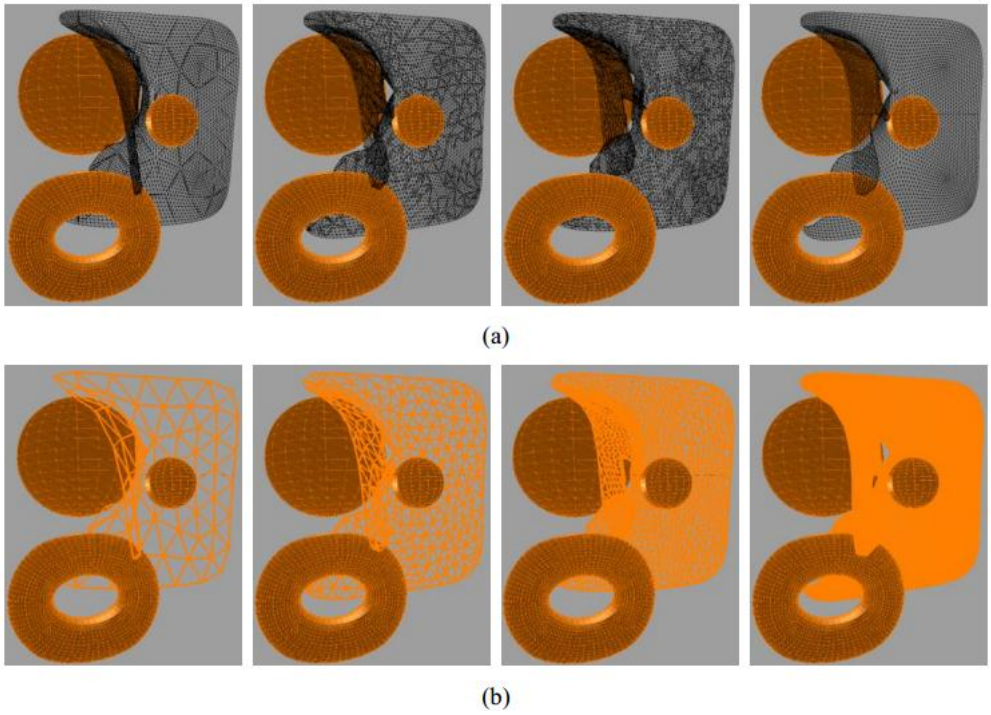


Figure 5-21 All the four models are cut and collide with the same environments. From left to right, the resolution of the collision mesh is increased. (a) Visual meshes. (b) Collision meshes.

Table 5-5 Performance of the pre-processing phase in the second experiment of cutting a thin sheet.

#	Collision vertices	Collision triangles	Preprocessing time [s]	Per collision triangle after preprocessing	
				visual vertices	visual triangles
1	65	104	6.9	73	102
2	280	500	35.1	25	29
3	796	1500	105.9	11	10

Table 5-6 Performance of the simulation phase in the second experiment of cutting a thin sheet.

#	Per cut time [ms]			FPS	
	Total cut time	Simulation without collision time	Simulation with collision time	Without collision	With collision
1	13.6	5.1-7.0	12.0-15.1	141.1	68.1
2	12.8	6.5-7.8	21.2-24.5	134.2	39.7
3	11.9	7.1-8.5	44.1-47.3	112.0	21.7
4	9.8	12.1-14.5	130.6-134.2	86.3	7.5

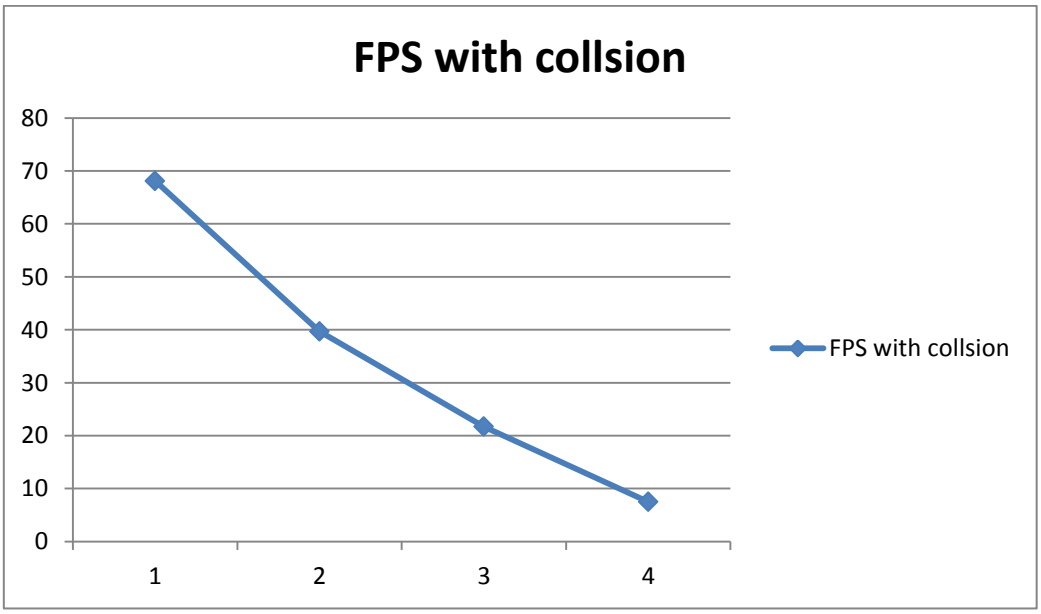


Figure 5-22 Comparison of FPS of the simulation with collision detection for four different collision meshes.

5.4.3 Ligaments

The proposed model was further applied to the ligaments in the knee. A

typical diagnostic procedure for ligaments is to examine the firmness of the ligaments with a hook—a surgical instrument used for retraction of tissues. Figure 5-23 illustrates the models for PCL and ACL. With reference to Figure 5-24, PCL and ACL are probed by the virtual instrument. The simulation includes interactions of tool-tissue and tissue-tissue. Both PCL and ACL are deformed by co-rotational FEM. The video with this example can be seen at https://youtu.be/aiD_8Oucjic.

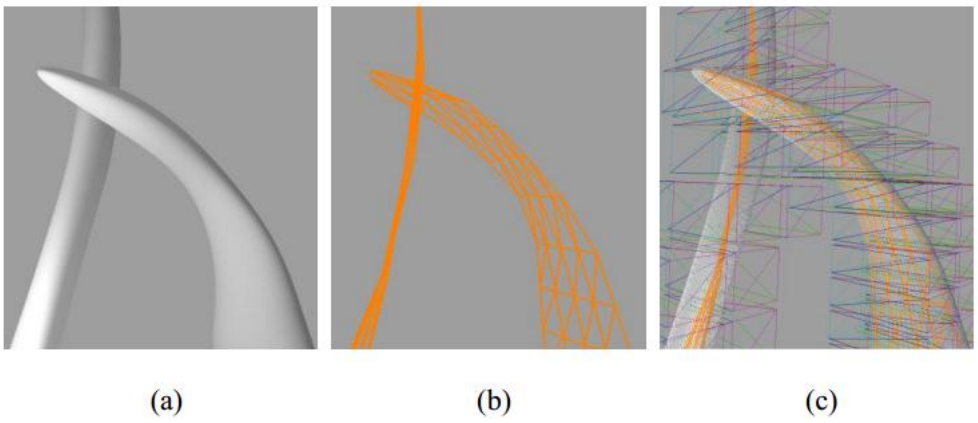


Figure 5-23 ACL and PCL. (a) Visual meshes. (b) Collision meshes. (c) Visual meshes overlap with collision meshes and simulation meshes.

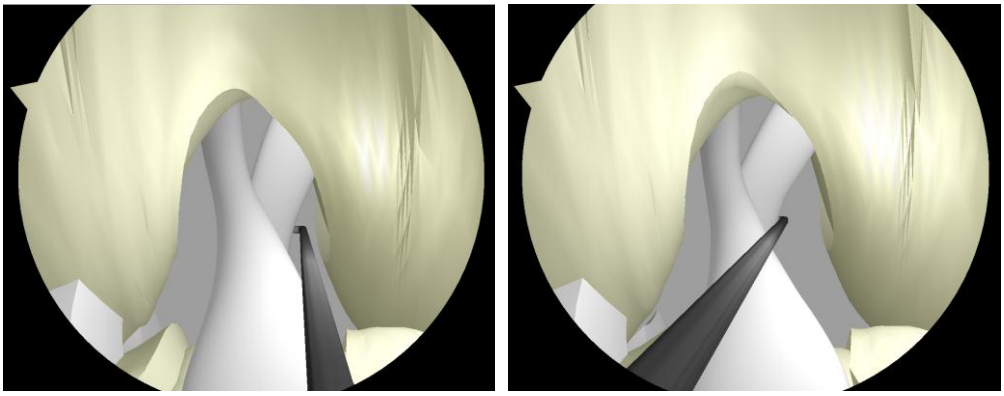


Figure 5-24 Examine the firmness of ligaments. (a) Before deformation. (b) After deformation.

Figure 5-25 illustrates various settings for the collision meshes of both ACL and PCL, all the other models in the scene are the same. The same operations are performed for these settings. The haptic force feedbacks in these settings are calculated by the method in [169], the computational time of which is also affected by the resolution of collision meshes. The simulation data are collected in Table 5-7.

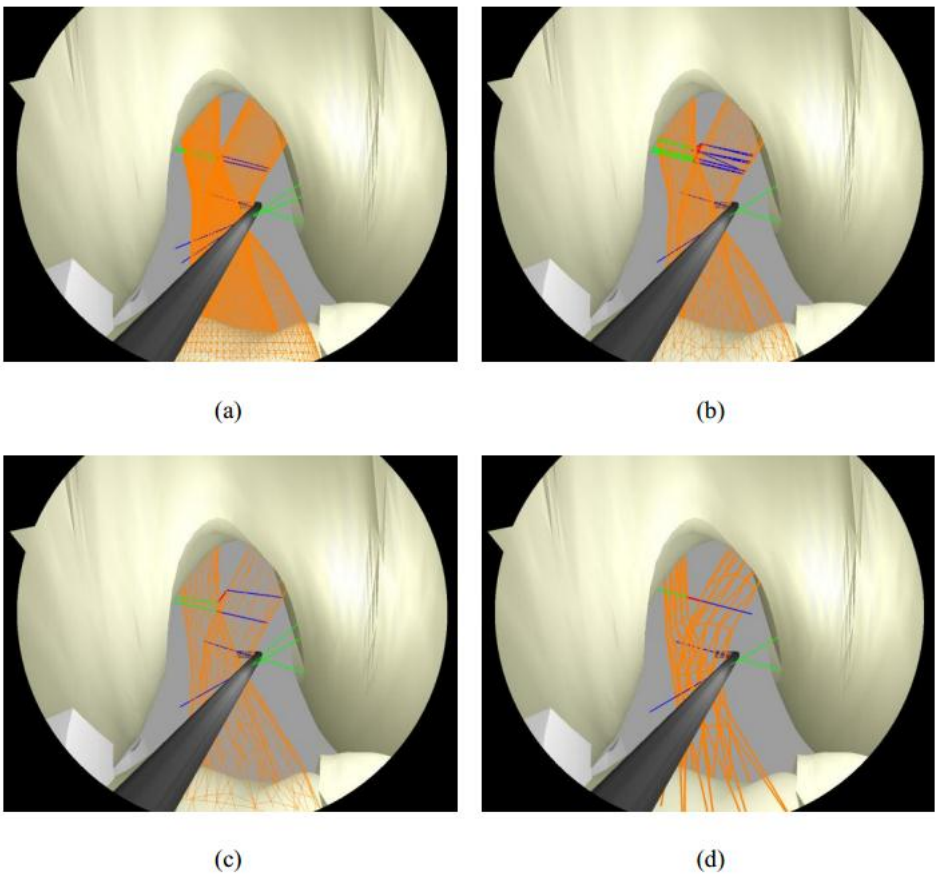


Figure 5-25 Various setting for the collision meshes of ACL and PCL. Interaction forces are illustrated by red lines. Green and blue lines are the opposite forces on the objects, respectfully.(a) The visual meshes are directly used for collision handling. (b) The collision meshes of ACL and PCL are simplified with about 2000 triangles. (c) The collision meshes are simplified by about 500 triangles. (d) The collision meshes are the reduced triangles with radii.

In the first setting (Figure 5-25(a)), the same meshes are used for both visualization and collision handling. The simulation rate is about 4.1 FPS after the ACL and PCL are deformed. In the second setting (Figure 5-25(b)), the collision mesh for ACL has 1012 vertices and 2001 triangles, while the collision mesh for PCL has 998 vertices and 1999 triangles. The simulation rate is about

19.3 FPS after deformation. In the third setting (Figure 5-25(c)), the collision mesh ACL has 252 vertices and 500 triangles, while the collision mesh for PCL has 249 vertices and 499 triangles. The simulation rate is about 44.2 FPS after deformation. In the fourth setting (Figure 5-25(d)), the reduced triangles with radii are used. The collision mesh of ACL has 25 vertices and 32 triangles, while the collision mesh of PCL has 45 vertices and 64 triangles. The simulation rate is about 86.1 FPS after deformation. Figure 5-26 illustrates the improvements of FPS from setting No. 1 to setting No. 4. The reduced triangles with radii (setting No. 4) achieve the best performance.

Table 5-7 The simulation data for various settings of ACL and PCL.

Setting #	Collision mesh				FPS after deformation	Simulation Time(ms)
	ACL		PCL			
	vertices	triangles	vertices	triangles		
1	5702	12064	5634	11264	4.1	239-243
2	1012	2001	998	1999	19.3	49-51
3	252	500	249	499	44.2	21-23
4	25	32	45	64	86.1	10-12

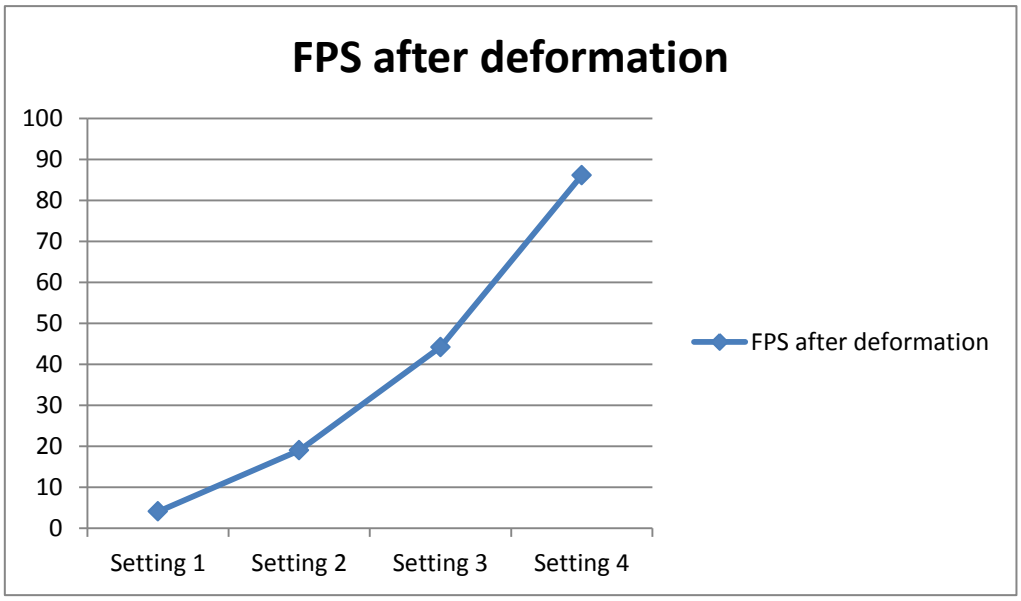
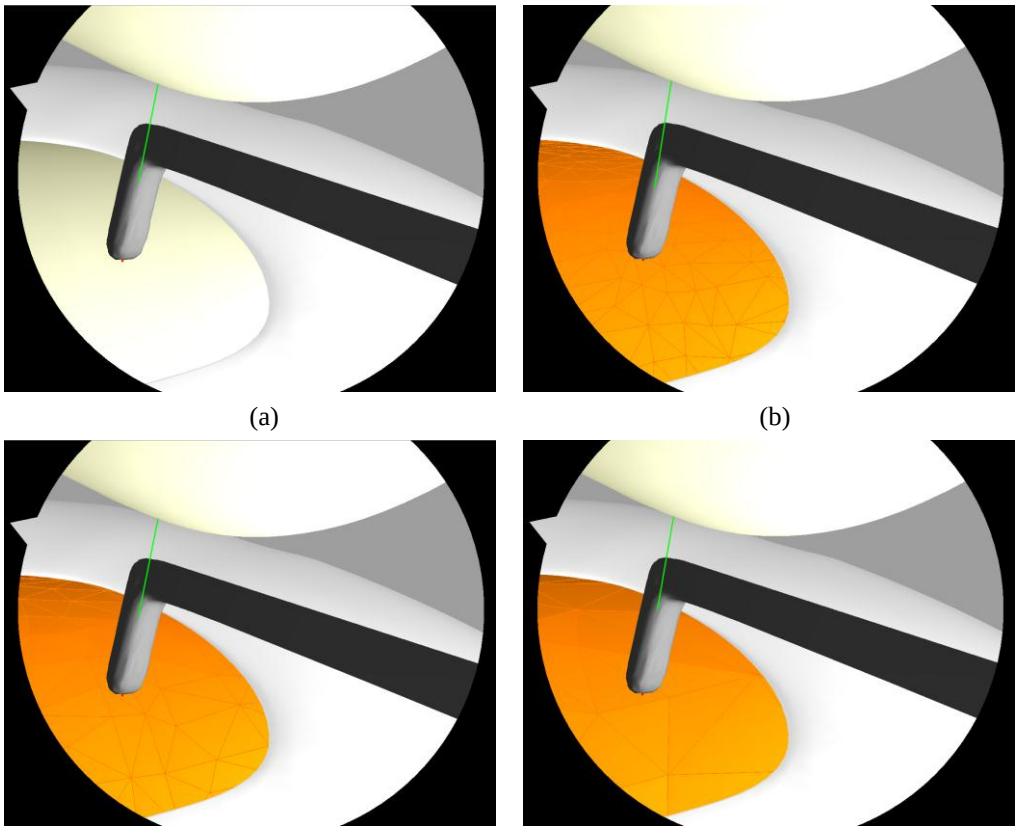


Figure 5-26 Comparison of FPS for four different settings of collision meshes of ACL and PCL.

5.4.4 Cartilages

The proposed model was also applied to simulation of cartilages. In most of the procedures on knee arthroscopy, the cartilages do not show large deformation. Surgical tools and other tissues may collide with cartilages. Thus, to save computational time, the simulation of cartilages is simplified as none-deformable tissues in most cases. For example, in the medial cartilage of tibia a high resolution mesh is used for the visualization, while various resolutions of the collision meshes could be used (Figure 5-27). In setting No. 1, the visual mesh and the collision mesh are the same mesh (5062 triangles). In setting No. 2, the collision mesh is reduced to 1000 triangles. In setting No. 3,

the collision mesh is reduced to 500 triangles. In setting No. 4 the collision mesh is reduced to 100 triangles. With reference to Figure 5-28, some part of the cartilage could be worn out, and the bone under the cartilage could be seen. Different material properties could be set, such that the haptic force feedback double be felt for the bone (hard) and the cartilage (soft).



(a) (b)

(c) (d)

Figure 5-27 Various resolutions for the collision mesh of cartilage. The orange mesh is a collision mesh. Similar to the previous figures, the forces are visualized as lines. (a) The visual meshes. (b) The collision mesh has 1000 triangles. (c) The collision mesh has 500 triangles. (d) The collision mesh has 100 triangles.

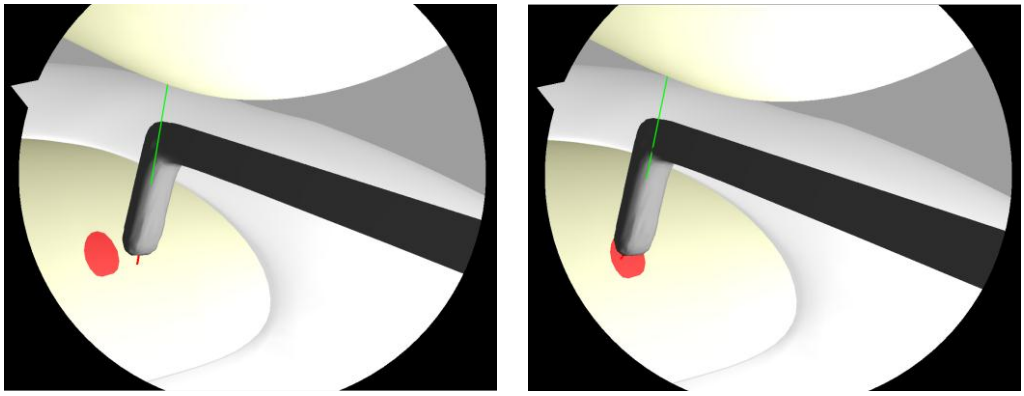


Figure 5-28 Haptic force feedback for different materials. (a) Force generated by touching the cartilage. (b) Force generated by touching the bone.

5.5 Summary

In this chapter, the visual-collision binding has been proposed, by which the framework described in [140] was extended. Several new techniques have been developed to support the interactive cutting of deformable thin objects in our framework. A high resolution visual mesh is bound to a low resolution mesh by the calculation of geodesic paths. With the visual-collision binding built in the pre-processing phase, high resolution visual mesh is cut efficiently with local 2D coordinates in the simulation phase. After the cutting, the visual-collision binding, visual-simulation and collision-simulation embedding are updated locally. Experimental results show that thin deformable objects can be cut interactively with different resolution of visual, collision and simulation meshes. Compared to the works that use the same mesh for visualization and collision, the resolution of the collision mesh in the proposed method can be tuned to different magnitudes. Compared to the methods that generate collision points by down-sampling the visual mesh, the proposed method can provide the collision

points, edges, and triangles with the same approximation property as the original collision mesh. The hypothesis for the cutting simulation has been verified by the above results. The research goal of developing a cutting model has also been achieved. In addition, the proposed model can be applied to simulation of ligaments and cartilages in the knee. In the next chapter, the implementation issue of the cutting simulation will be discussed in more details using the example of meniscus simulation.

Chapter 6 Implementation of Virtual Reality Knee Arthroscopic Simulator

This chapter discusses the technical implementation issues for the simulation which was needed for validation of the proposed and developed models. Relevant libraries and toolkits are briefly reviewed in Section 6.1. The data structure for the simulation is presented in Section 6.2 and a multi-resolution implementation method is discussed in Section 6.3. A knee arthroscopy surgery-training simulator is presented in Section 6.4, followed by a summary given in Section 6.5. The chapter was partially published in [163] and [55].

6.1 Related libraries and toolkits

In this section, related libraries are briefly reviewed in terms of VR-based medical training simulation. As described in Chapter 2, the key elements in such application are deformable body simulation, collision handling, cutting operation and haptic force feedback.

Vega [170] is an open source middleware library that supports three-dimensional deformable object simulation. It includes co-rotational linear FEM [145], invertible isotropic nonlinear FEM [171], Saint-Venant Kirchhoff FEM deformable model [172], mass-spring system, etc. It also supports model reduction [173], cloth simulation [161], and rigid body dynamics, etc. The library is written in C/C++, and its components are dependent minimally on each other. It supports Windows, Linux and Mac OS X. Thus, this library could

be easily integrated into many other existing codes. However, its support for the cutting operation and collision handling is limited.

FEBio [174, 175] is an open source software suite that supports finite element analysis (FEA) in biomechanics and biophysics. The main focus of this library is to solve nonlinear large deformation problems. It contains three packages: PreView that provides the mesh generation; FEBio that provides boundary conditions and constitutive models; PostView that could visualize the results from FEA. However, it does not support the haptic force feedback, topological operation and collision handling.

NiftySim [176] is a light weight FEM toolkit written in C++ and CUDA. It supports several nonlinear models which include Total Lagrangian Explicit Dynamics (TLED). Although this toolkit handles collision, it does not support topological operation and haptic force feedback.

chai3d [177] is an open source C++ library that supports many haptic devices, force algorithms, audios, contact detection, rigid and soft bodies. It could be run on Windows, Linux and Mac OS X. However, it does not support FEM, and topological operation.

PhysX [178] is multi-platform library provided by the company NVIDIA. Its features include raycasting and shape sweeping, collision detection, rigid body dynamics simulation, cloth simulation, and smoke simulation. It supports not only the PC platform (Windows, Linux and Mac OS X), but also the smart phone platform (Android and iOS). However, it does not support FEM, topological operation, and haptic force feedback.

Simulation Open Framework Architecture (SOFA) [5] is an open source C++ library that supports multi-model framework. FEM simulation, collision detection and response and haptic force calculation are provided as independent components. It supports Windows, Linux and Mac OS X. One major feature which makes it especially suitable for this project is that, the library allows for the deformation of an object with different resolutions in the visualization, collision and simulation. Table 6-1 summarizes the related libraries in terms of surgical simulation.

Table 6-1 Summarization of related libraries

	FEM	Collision	Haptic	Topological operation	Multi-model
Vega	√	×	×	×	×
FEBio	√	×	×	×	×
NitfySim	√	√	×	×	×
chai3d	×	√	√	×	×
PhysX	×	√	×	×	×
SOFA	√	√	√	√	√

These toolkits (or libraries) have their own advantages. For the surgical training simulation, SOFA is the best choice for this project. The flexible architecture and extensive components provided by SOFA make it very suitable

for testing new algorithms. In addition, many medical simulators have been successfully developed based on SOFA [179-181]. Therefore, all the proposed methods are implemented on SOFA. At the same time, although basic topological operations have been proved by SOFA, the cutting of multi-resolution model is not directly provided currently. Therefore, the implementation of this thesis enriches its cutting components. The contributions mainly include:

(1) Codes for the function enhanced deformation method described in Chapter 4,;

(2) Components for the visual-collision binding discussed in Chapter 5;

(3) New way of organizing the scene for multi-resolution cutting simulation, because of the introduction of the above new components and codes.

Contribution (1) is implemented and integrated into the existing component called “BarycentricMapping” in SOFA. Contribution (2) and (3) are described in Section 6.2, and demonstrated through experiments in Section 6.3 and Section 6.4.

6.2 The organization of the simulation scene

Following the scene graph data structure provided in SOFA, here the simulation of knee arthroscopy is elaborated as an example. The whole simulation scene is organized as in Figure 6-1. The root node controls the whole simulation,

including the node of tool, meniscus, femur, etc. There are child nodes of collision and visual for the tool node. Meniscus and Femur nodes are similar.

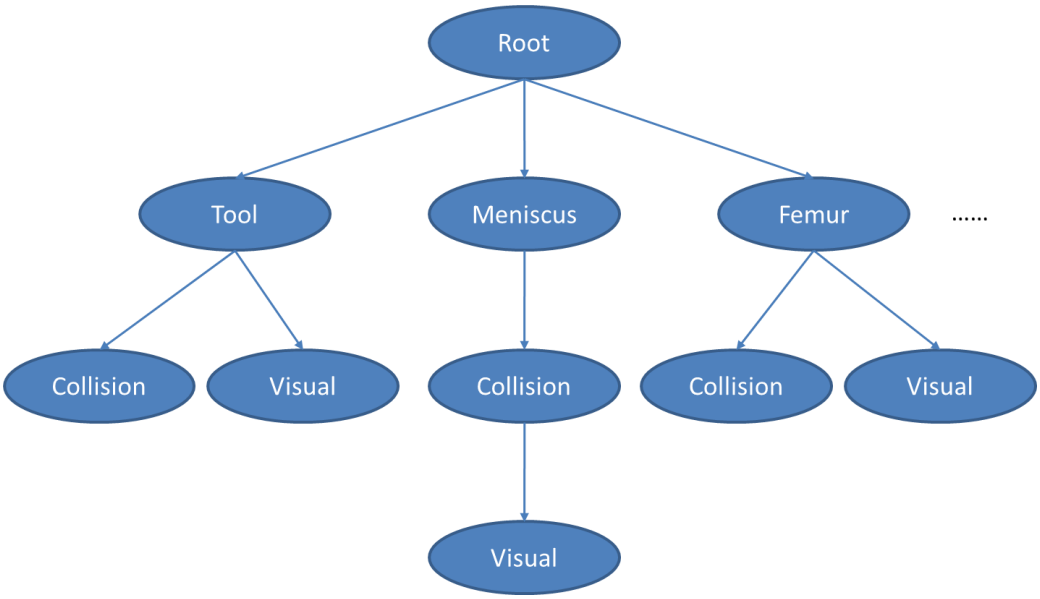


Figure 6-1 Overall data structure of the simulation scene of VR-based knee arthroscopy. The blue arrow links the parent node to their child nodes.

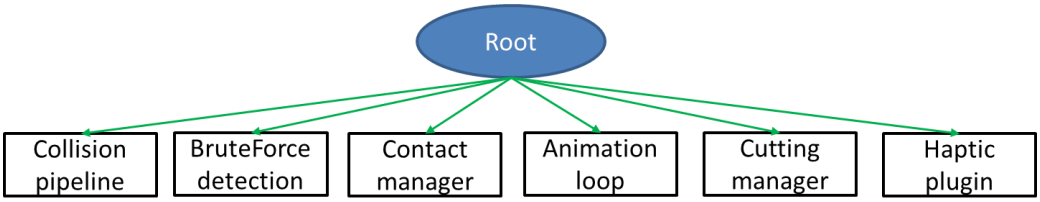


Figure 6-2 Major components in the root node. The green arrows connect the node and the components belonged to it.

With reference to Figure 6-2, for the “Root” node, the components mainly include:

1. Collision pipeline, which controls the collision handling sequence;
2. Brute force detection, which controls the broad phase and the narrow phase of collision detection;
3. Contact manager, which controls the contact response;
4. Animation loop, which controls the whole animation steps including the updating of collision, visual, and mechanical state;
5. Cutting manager, which controls the cutting operations;
6. Haptic plugin, which links the simulation to the haptic device and calculates the force feedback.

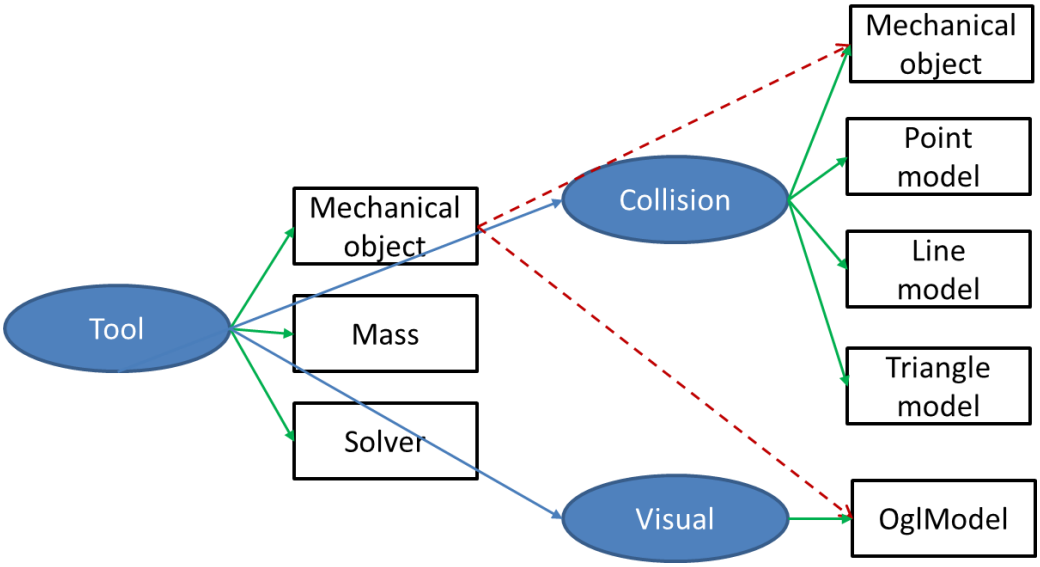


Figure 6-3 The structure of the "Tool" node. The red arrows show the rigid mapping between components.

With reference to Figure 6-3, for the "Tool" node, it has at least three components:

1. *Mechanical object*. It controls all the mechanical states, which include positions, velocities and forces for the degree of freedoms;
2. *Mass*, which stores the mass of the tool;
3. *Solver*, which controls the movement of the mechanical object.

For the "Collision" node, it has mechanical object and collision detection components (including point, line and triangle). For the "Visual" node, it has a component for the visualization (called "OglModel"). The red dash arrows show the rigid mapping that map the position and velocity from "Tool" node to "Collision" node and "Visual" node. With this mapping, the collision model and the visual model could be moved consistently during the simulation.

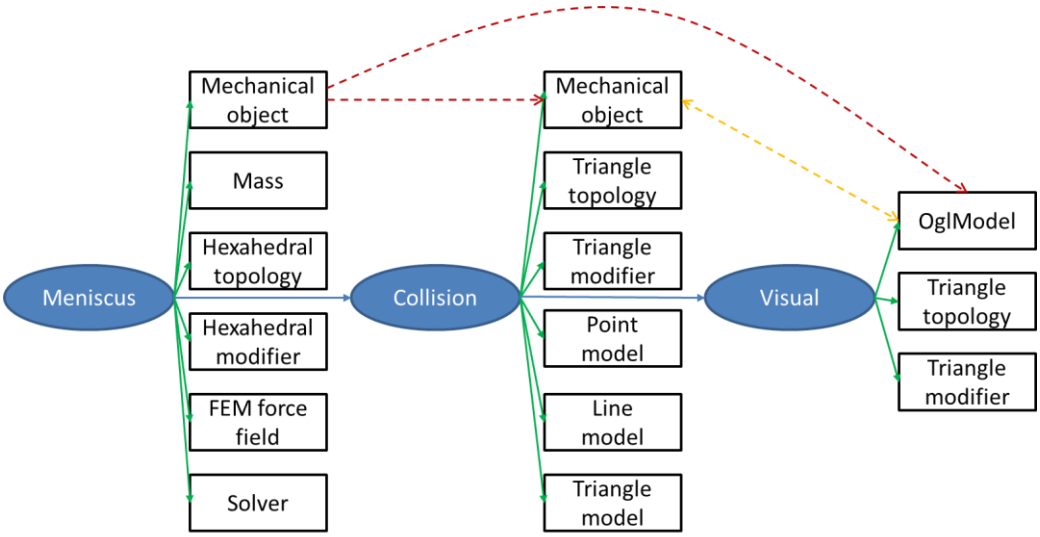


Figure 6-4 The structure of the "Meniscus" node. The red dash arrows are the barycentric mapping between components, while the orange dash double arrow is the visual-collision binding.

With reference to Figure 6-4, for the “Meniscus” node, it has at least six components. The mechanical object, mass and solver are similar to the components in the “Tool” node. The hexahedral topology stores the information of the simulation mesh, including hexahedra, faces and edges. The hexahedral modifier provides the topological operations for the hexahedral mesh, such as the removal of hexahedra. The FEM force field defines the co-rotational linear FEM force field. The “collision” child node consists of six components. The triangle topology stores the information of the collision mesh, i.e. triangles and edges. The triangle modifier provides the topological operations for the triangle mesh, such as the removal and the adding of triangles. It also has the components for the collision detection. For the “Visual” child node, it has three components: OglModel, triangle topology, and triangle modifier. The red dash arrows show the barycentric mapping from the “Meniscus” node to the “collision” node and the “Visual” node. The forces are generated in the “collision” node as the result of the collision response and then mapped to the “Meniscus” node. The simulation is run on the “Meniscus” node, and then the positions and velocities are mapped to both the “Collision” node and the “Visual” node. In this way, the collision mesh and the visual mesh could be moved consistently.

For the implementation of the wrinkles generation method proposed in Chapter 4, the monitor and region points are embedded into the simulation-visual barycentric mapping. Both the visual and collision mesh are then updated during the simulation, to maintain their consistency.

For the implementation of the cutting method proposed in Chapter 5, the visual-collision binding is stored in the “Visual” node in the pre-processing phase (shown as the orange dash double arrow in Figure 6-4). In the simulation phase, the cutting operation is relied on the “Visitor” design pattern, which has already been provided in SOFA. In each animation loop, the cutting manager (as shown in Figure 6-2) monitors if the cutting blade intersects with the collision mesh. If intersection happens, the cutting of collision triangles is implemented by adding new triangles and removing old triangles. The cutting manager controls the whole cutting process with the following steps:

1. Send “TrianglesAdded” events. The “Collision” node catches this event to add triangles to its topology component. The “Visual” node catches this event to cut the visual mesh and update the visual-collision binding.

2. Send “TrianglesRemoved” events. The “Collision” node catches this event to remove triangles from its topology component. The “Visual” node catches this event to remove the binding information of these triangles.

3. Send “UpdateMapping” events. The “Collision” node catches this event to perform branch analysis to modify the hexahedral mesh in the “Meniscus” node, and then update the simulation-collision barycentric mapping. The “Visual” node catches this event to update the simulation-visual barycentric mapping. Since the “Visual” node is a child node of the “Collision” node, its event catching operation is performed after the operation of the “Collision” node. The sequence is important, since the hexahedral mesh needs to be modified before updating the simulation-visual mapping.

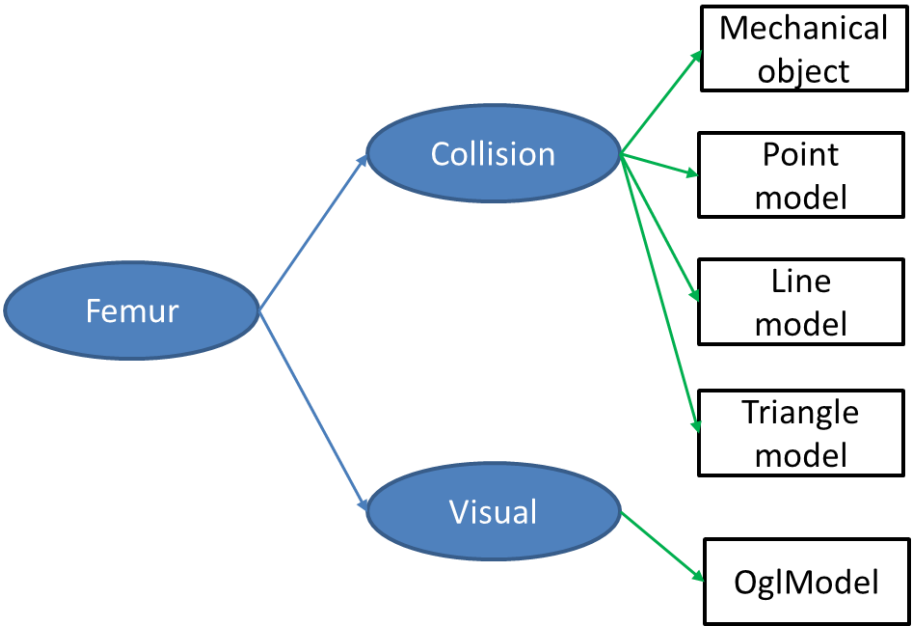


Figure 6-5 The structure of the "Femur" node.

With reference to Figure 6-5, since the femur is simulated as a static bone in the simulation, no component is required for the "Femur" node. The "collision" node and the "Visual" node are similar to the ones in the "Tool" node. The difference is that, no mapping component is required.

To summarize, there are three different types of objects in the simulation: (1) surgical tools, the implementation of which is similar to the "Tool" node; (2) soft tissues, the implementation of which is similar to the "Meniscus" node; (3) static background scene, the implementation of which is similar to the "Femur" node.

6.3 Multi-resolution implementation

The simulation must be run in real time. The collision detection and response

can easily become the bottleneck of the whole simulation. This section discusses the performances of multi-resolution implementation through the simulations of meniscus, ligaments and cartilages. All the implementations are run on a desktop PC equipped with Intel i7-4770 3.40 GHz, 16 GB of RAM, and NVIDIA GeForce GTX 750Ti.

Take the simulation of meniscus palpation for example. With reference to Figure 6-6, the upper-side surface of the meniscus is pressed down by the palpation hook, and then the lower-side surface of the meniscus touches the tibia. The video of this example can be seen at https://youtu.be/v4ABNZRtoTg. The same visual meshes are used for the simulation (Figure 6-6 (a)), while three different setting of the collision meshes are used (Figure 6-6(b)(c)(d)). The same operations are performed for these settings. The co-rotational linear FEM [75] on hexahedral mesh is used for the simulation of meniscus. The simulation data are collected in Table 6-2. In the first setting (Figure 6-6(a) and Figure 6-6(b)), the same meshes are used for both visualization and collision handling. The simulation rate is about 33 FPS before the meniscus is pressed down, while the simulation rate after the meniscus is pressed down is about 14 FPS. Interactive simulation rate cannot be achieved in setting No. 1. In the second setting (Figure 6-6(c)), the collision meshes of the lower bone and the upper bone are simplified, while the collision mesh of meniscus remains unchanged. The collision mesh of the tool is replaced by its center line. A proper collision distance of this center line needs to be tuned to avoid the penetration between

the visual model of the tool and the visual model of the meniscus. The simulation rate after the meniscus is pressed down is about 108 FPS. In the third setting (Figure 6-6(d)), the collision mesh of meniscus is further replaced by the reduced middle triangle mesh with radii as described in Chapter 6, while all the rest collision meshes are the same in setting No. 2. The simulation rate after the meniscus is pressed down is about 149 FPS. Compared to the setting No. 2, there is a clear improvement of the performance for the introduction of the reduced triangle mesh for the meniscus. Improvements of the simulation rates from settings No. 1 to No. 3 could be seen in Figure 6-7. This result is similar to the experimental results in in Section 5.4.2.

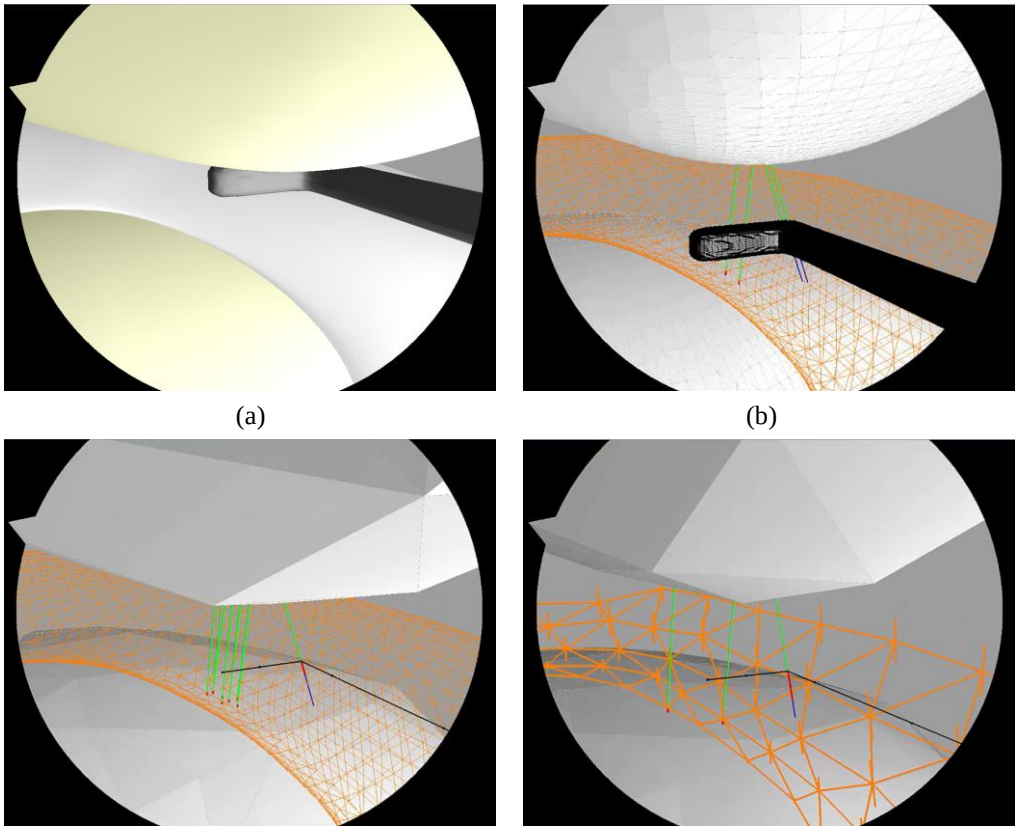


Figure 6-6 Simulation scene with different settings of collision meshes. Interaction forces are illustrated red lines. Green and blue lines are the opposite forces on objects respectfully. (a) The visual meshes. The upper and lower bones are visualized by smooth yellow meshes, the meniscus is visualized by smooth white meshes. (b) Setting No.1, the collision meshes are the same as visual meshes. (c) Setting No.2, the collision meshes are simplified except for the meniscus. (d) Setting No.3, the collision mesh of meniscus is replaced with a reduced triangle mesh.

Table 6-2 The simulation data for three different settings.

Setting #	DOFs of collision mesh				FPS after pressed	Simulation Time(ms)
	Meniscus	Lower bone	Upper bone	Tool		
1	2,882	2206	5850	2562	14	28-31
2	2,882	63	59	7	108	8.5-9.4
3	105	63	59	7	149	6.2-6.8

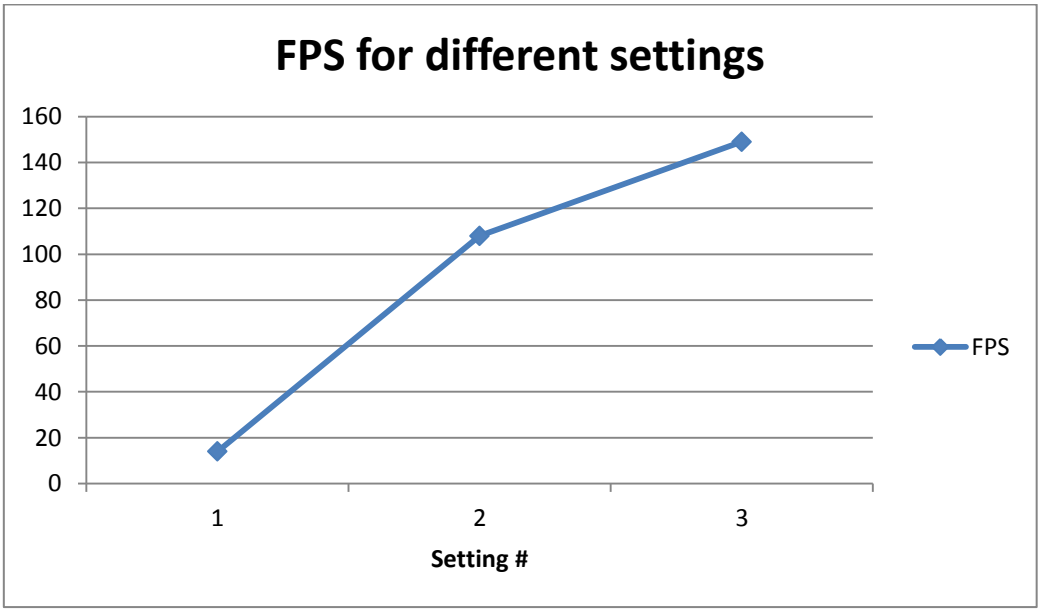


Figure 6-7 Comparison of FPS for three different settings of collision meshes.

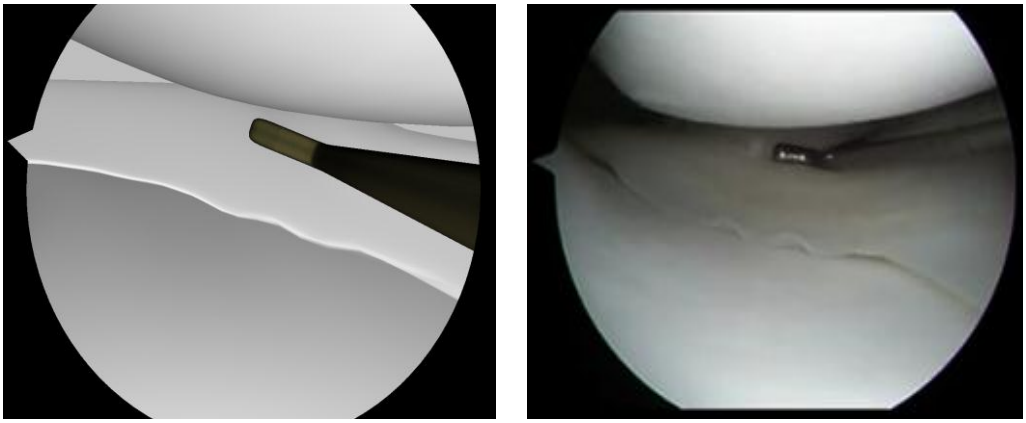
6.4 Knee arthroscopy simulation

The computational advantages of the proposed multi-resolution cutting framework have already been discussed in Chapter 5 (See Section 5.4). In this

section, the proposed methods are integrated into a VR-based prototype knee arthroscopy simulator. This simulator includes the simulation of the procedures of the palpation of the meniscus with wrinkles and the meniscus lesion handling.

6.4.1 Meniscus examination

The method proposed in Chapter 4 is tested in the simulation of meniscus examination. The video of the real time recording of the simulation could be seen at https://youtu.be/NsZvuuEFbPc. With reference to Figure 6-8, a triangle mesh with 6096 faces is used for the surface of the meniscus, 42 hexahedra are used for the finite element simulation, 296 monitor points are inserted into the thin border of the meniscus and 70 monitor points are selected for the wrinkle points according to the position of instrument-tissue contact points. Figure 6-8(a) illustrates the simulated deformation process and compares it with the actual meniscus deformation (Figure 6-8(b)). Compared with the data-driven method in [119], the time for the wrinkle generation is about 0.4 ms, which is similar to their work. They require pre-storage of 3.33 MB data, while this method requires 12.9 KB of monitor points and region points. Furthermore, this method is more flexible since the wrinkles deformation shape could be modified by adjusting a few coefficients. Figure 6-9 illustrates the simulated wrinkle in the examination of a torn meniscus.



(a) (b)

Figure 6-8 The simulation of the meniscus examination. (a) The wrinkles generated by our method. (b) Snapshot from the real surgical video.

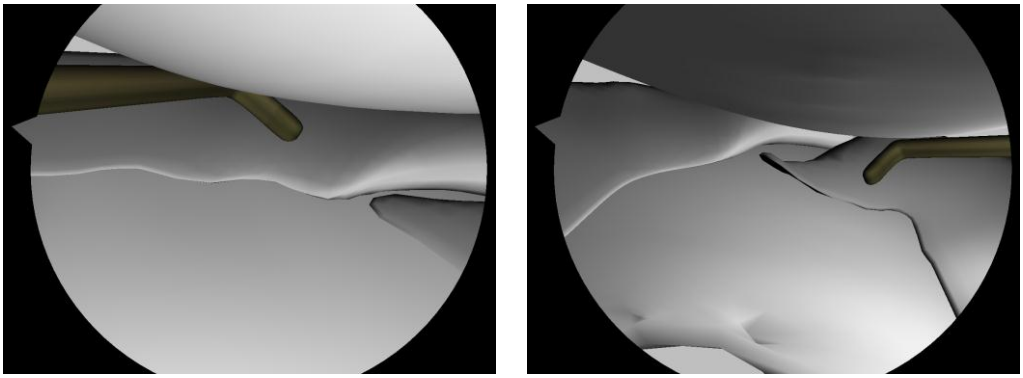
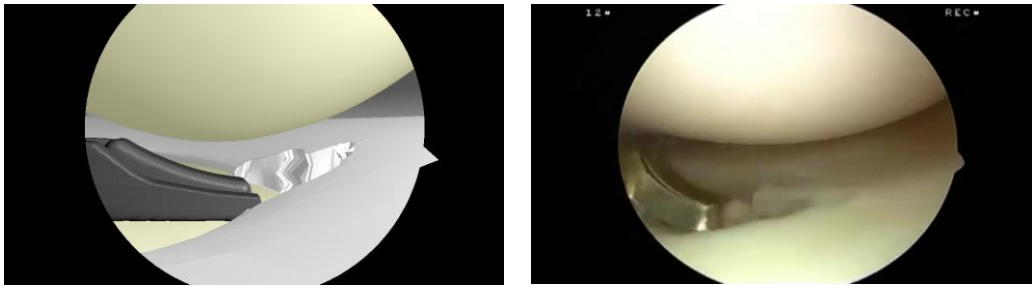


Figure 6-9 The virtual examination of the meniscus with tears. Different part of the torn meniscus is examined by a rigid hook.

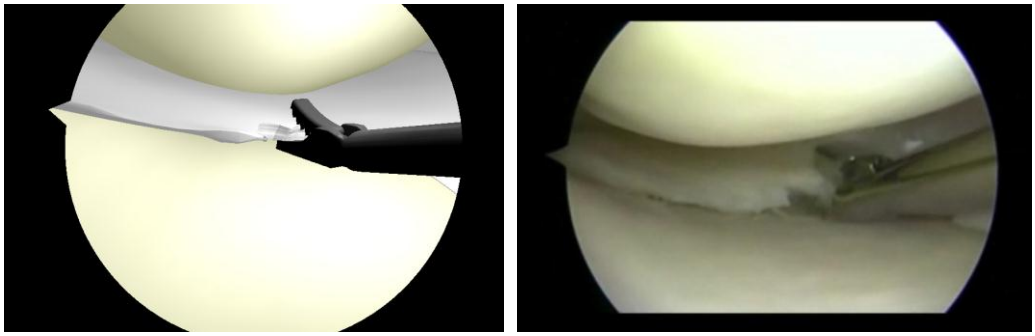
6.4.2 Punching out the torn tissues of meniscus

The proposed cutting method in Chapter 5 has been applied in the simulation of handling meniscus lesion. In this procedure, the surgeons punch out the torn tissue and then contour the edge of the meniscus with a small shaver. Figure 6-10 shows the cutting of the meniscus in our simulator compared to the

snapshots from real surgical videos with the similar angle of views. More information can be seen on the videos at https://youtu.be/dk2yTVX3d7A. With reference to Figure 6-11, the meniscus model is composed of the visual mesh with 2,882 vertices and 5,760 triangles, the collision mesh with 105 vertices and 160 triangles, as well as the simulation mesh with 120 vertices and 42 hexahedra. The pre-processing time is about 12.7 sec. There are about 64 visual vertices and 76 visual triangles for each collision triangle after the pre-processing phase. Figure 6-12 shows the cutting process. Each punch of the meniscus is composed of the cuts of a few collision triangles (Figure 6-12(b)). For each punch, the average total cutting time is 48.2 ms. The time for cutting collision and visual mesh and updating collision-visual binding is about 6.2 ms. The time for updating visual-simulation and collision-simulation embedding is 32.8 ms, while the time for adding the cut surface is about 8.1 ms. The time for each simulation step is about 6.5 ms, while the average number of frames per second is 162.

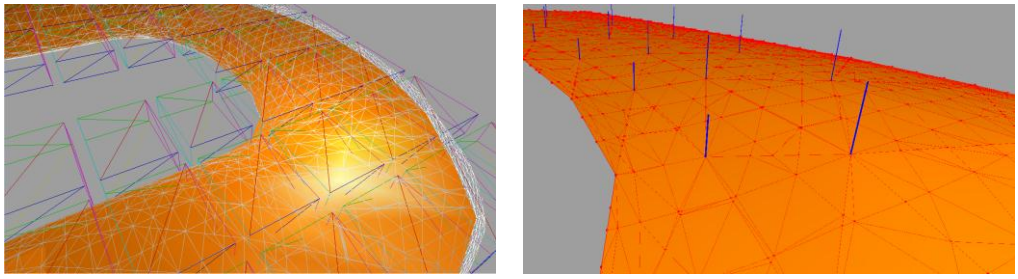


(a)



(b)

Figure 6-10 The simulation of punching out the torn parts of a meniscus. The left images are our simulations while the right ones are the snapshots from real surgical videos. (a) The meniscus after several punches. (b) The meniscus after several punches on some other area.



(a) (b)

Figure 6-11 The meniscus model. (a) The visual, collision and simulation meshes before cutting. The white wireframe mesh is the visual mesh, the orange mesh is the collision mesh, and the hexahedral mesh is the simulation mesh. (b) The close up view of the visual and collision bindings after the pre-processing. The red points and lines are the visual vertices and edges bound to the collision mesh respectfully. The blue lines show the links between the collision vertices and their paired visual vertices.

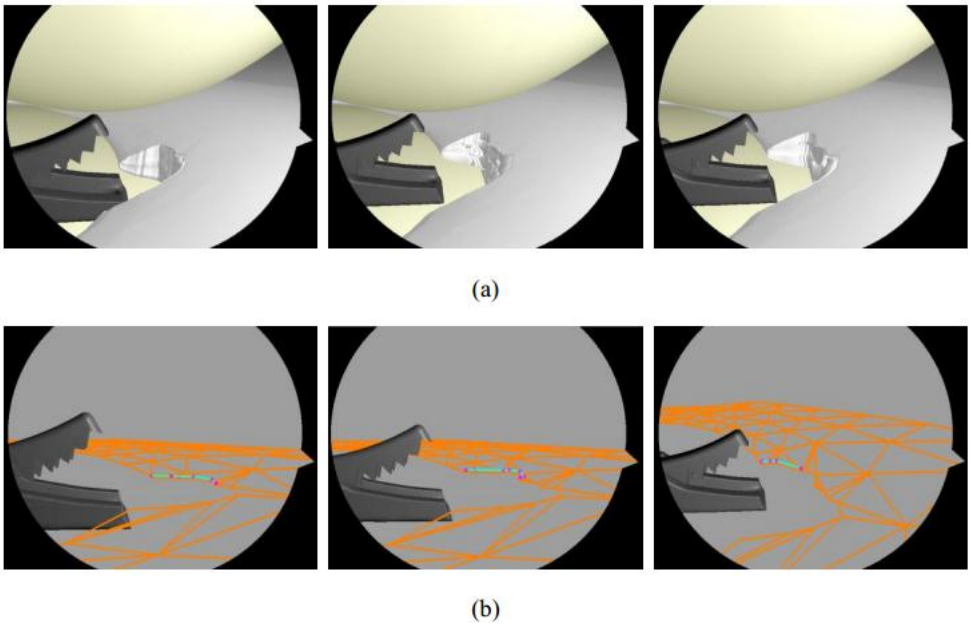


Figure 6-12 The cutting process of the meniscus. (a) The visual mesh of meniscus after each punch. (b) The relevant collision mesh of the meniscus. The blue lines are the cutting paths, while the red points are the intersection points between the cutting triangles of the scissor and the collision edges of the meniscus.

In this example, the most related work which can be used as a benchmark is [158]. However, its authors did not provide the data of their models, thus it is difficult to make a precise comparison. The authors simulated punching of the meniscus by element removal, while in the thesis it is simulated by cutting the collision triangles. In the authors' simulation, the resolution of tetrahedra has to be fine enough to represent the shape of the cutting blade. The resolution of triangles in the method proposed in the thesis does not have this constraint. The authors reported 86 ms for a total cut time of a punch, while the total cut time in our method is 48.2 ms. However, it is an unfair comparison since the paper was published in 2011, and the meniscus model used is different from that used in the thesis. Anyway, a clear advantage of the proposed method is the updating of

the collision mesh, just as the example of cutting a steak. In [158] there is no mentioning of the collision, while the proposed method separates the resolution of the visual and collision meshes.

6.4.3 Removing torn tissues pieces by pieces

In a real knee arthroscopy, torn tissues are usually removed pieces by pieces using a large shaver. The method proposed in Chapter 5 also supports the simulation of this operation. Based on the visual-collision binding, when the shaver's blade intersects with collision triangles of the tissues, both the collision triangles and the visual triangles bound to them are removed. Figure 6-13 shows the process of removing torn tissues. The videos can be seen at https://youtu.be/sTNMJeMbkdY. Figure 6-14 illustrates the visual and collision models for the torn tissues.

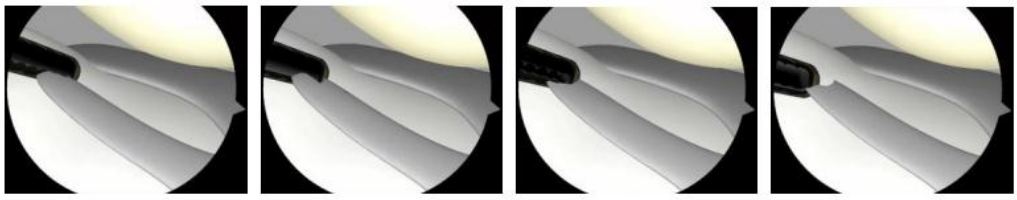


Figure 6-13 The removal of the torn tissues pieces by pieces.

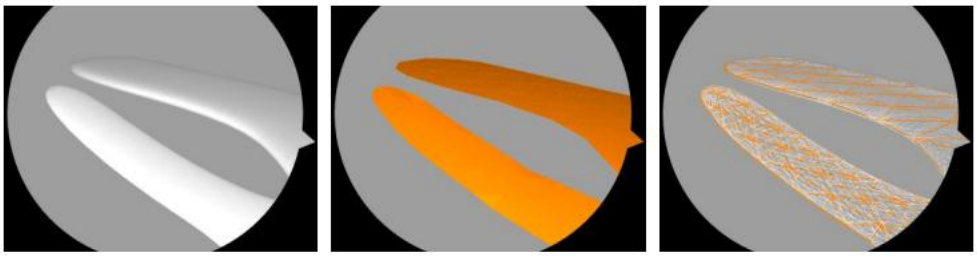


Figure 6-14 The visual and collision model for the torn tissue. (a) The visual mesh. (b) The collision mesh. (c) The visual mesh is overlapped with the collision mesh. Meshes are shown in wire frame mode.

6.4.4 Contouring the edge of meniscus

Figure 6-15 shows the contouring process in our simulation. To simulate the contouring operation, jagged surface needs to be generated on the edge of the meniscus in the punching simulation, see Figure 6-12(a). This is achieved by introducing random numbers at the cut surface mesh generating section. Then, the model of a small shaver is activated and the shaving mode is turned on. In the shaving mode, collision triangles will not be cut when the small shaver collides with them. Instead, the intersected parameters on the collision edges are monitored. Since the cut surface mesh was bound to these collision edges in the punching step, the simulation of contouring is achieved by smoothing the cut surface around the intersected parameters. During the contouring process, the meniscus is getting deformed when the small shaver pushes on it, just as the behavior of the meniscus in the real surgery. The simulation of contouring the edge of meniscus has not been found in the existing simulators. In the examples of [158] and the released demos of the commercial simulator ArthroS in [63],

both smooth cut surface are generated, which leaves no room for the contouring operation.

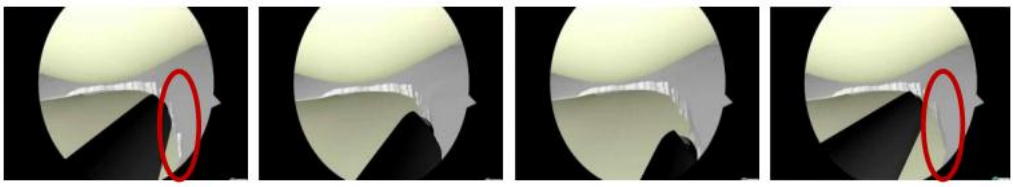


Figure 6-15 Contouring the edge of the meniscus. The red circle highlights the edge of the meniscus before and after contouring.

6.5 Summary

In this chapter, the implementation of the simulation on SOFA has been presented. The efficiency of the multi-resolution simulation has been proved by the comparison results in different collision settings. The proposed methods in the previous chapters have been further tested by the virtual knee arthroscopy simulator. Compared to the previous works, the proposed method in the meniscus cutting simulation has achieved higher FPS, and the resolution of collision mesh is not limited by the geometry of the cutting blade. Besides, the essential procedure of contouring the edge of a meniscus, which was not simulated in the existing simulators, can be supported by the proposed methods.

Chapter 7 Conclusion and Future Work

This chapter summarizes the conclusions and future recommended work based on the information presented in this thesis.

7.1 Conclusion

A survey of virtual reality arthroscopic simulation was presented and the literatures related to the simulation of deformable objects were reviewed. The research room for the development of the model for the simulation of thin deformable objects was identified. For the deformation simulations, a hypothesis was proposed that by embedding points into coarse simulation mesh, local detailed deformation could be efficiently generated. For the cutting simulations, a hypothesis was proposed that by introducing visual-collision binding, multi-resolution thin deformable objects could be cut consistently and interactively. To defend the above hypotheses, methods were proposed for the deformation and cutting simulations that resulted in the following new research and development protocols:

1. A novel method for local detailed deformation has been proposed. Monitor points and region points are embedded into the coarse co-rotational linear FEM simulation to identify the factors for the detailed deformation. The

coarse simulation state is monitored using the embedded points. The reference configuration for wrinkle generation is dynamically defined. A function-based method is used to generate tunable wrinkles. This method has been applied in the simulation of meniscus examinations and verified using a side-by-side comparison with actual surgical videos. Experimental results show that efficient local detailed deformations could be achieved using this method with only a small amount of computer memory.

2. Visual-collision binding used in the simulation of cutting thin deformable objects has been presented. A method has been proposed to bind a high-resolution visual mesh to a coarse-resolution collision mesh before simulation. A new technique that combines triangular splitting, 2D line intersections, and branch analysis has been presented to update the binding during the cutting simulation. The proposed visual-collision binding has been applied in the interactive cutting simulations of a steak, a thin sheet, and a meniscus. A comparison with previous works verified the research novelties of this method.

3. A virtual reality knee arthroscopy prototype simulator has been developed. A scene graph structure is utilized for the multi-model implementation of the simulator. The performances using various collision model settings in

the simulation of knee arthroscopic surgery have been presented and compared. Simulations of the examination and handling of the meniscus and its lesions have verified the usefulness of this model. The framework of the multi-model cutting simulation has also been verified using simulations of cutting a steak and a thin sheet.

The above results of this research and development have been published in one journal paper and three conference papers, which are listed in Appendix I "List of Publications". The results contributed to the completed externally funded project "Collaborative Haptic Modeling for Orthopaedic Surgery Training in Cyberspace". The knee arthroscopic surgery–training simulator was developed in consultation with the surgeon from the Alexandra Hospital, Singapore. In addition, a plugin for simulating the cutting of thin deformable objects will be published on SOFA in the near future.

7.2 Future Work

Based on the achieved results, this project has identified several directions for future work.

1. Automatic generation of the reduced collision mesh for thin objects

In this thesis, the multi-model framework based on a reduced collision mesh has proved to be efficient for cutting simulations; however, the collision mesh is

constructed manually using 3ds Max. This could be improved using a method to automatically generate the reduced collision meshes from the visual meshes of thin objects. The input of the simulation could be only the high resolution visual meshes. Then, the coarse-resolution (within the specified number by users) collision mesh with radii will be automatically generated.

2. Developments of the other types of virtual reality surgical training simulations

Only the knee arthroscopy simulator has been presented in this thesis. The proposed multi-model cutting framework could be applied to the simulation of many other surgical procedures. Shoulder and hip arthroscopic surgery could also be similarly simulated. In addition, the simulation could be applied to endoscopic and laparoscopic surgeries.

3. Multi-resolution cutting simulations for general deformable objects

The cutting method proposed in this thesis can be applied only to thin deformable objects. Visual-collision binding could be generalized to general deformable objects. A coarse tetrahedral mesh could be used, and the high-resolution visual mesh could be bound to the boundary of the tetrahedral mesh.

4. CPU-GPU mixed implementation of the simulation with different force fields

The proposed visual-collision binding method is independent of the simulation mesh. This thesis tested it only with the co-rotational FEM on the hexahedral meshes, which have boundary-alignment drawbacks. In the future, the proposed method will be tested with the tetrahedral meshes and different simulation methods. In addition, the performance could be further improved by implementing the collision detection on GPU.

Reference

1. Richmond J.C., Bono J.V., and McKeon B.P., *Knee Arthroscopy*, Springer-Verlag New York, New York, USA, **2009**.
2. Gonzalez, O. and A.M. Stuart, *A first course in continuum mechanics / Oscar Gonzalez and Andrew M. Stuart*. Cambridge University Press, Cambridge, UK, **2008**.
3. Scuderi G.R. and Tria A.J., *The knee: a comprehensive review*, World Scientific Publishing Co. Pte. Ltd., Singapore, **2010**.
4. Cueto, E. and Chinesta F., *Real time simulation for computational surgery: a review*, Advanced Modeling and Simulation in Engineering Sciences, **2014**, 1, 1-18.
5. Faure, F., et al., *SOFA: A Multi-Model Framework for Interactive Physical Simulation*, in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, Payan Y., Editor, Springer Berlin Heidelberg, **2012**, 283-321.
6. Erdemir A., *Open Knee: Open Source Modeling and Simulation in Knee Biomechanics*, The Journal of Knee Surgery, **2016**, 29, 107-116.
7. Seyfi B., Fatouraee N., and Imeni M., *Mechanical modeling and characterization of meniscus tissue using flat punch indentation and inverse finite element method*, Journal of the Mechanical Behavior of

Biomedical Materials, **2018**, 77, 337-346.

8. Guess T.M. and Razu S., *Loading of the medial meniscus in the ACL deficient knee: A multibody computational study*, Medical Engineering & Physics, **2017**, 41, 26-34.
9. Kazemi M., Dabiri Y., and Li L.P., *Recent Advances in Computational Mechanics of the Human Knee Joint*, Computational & Mathematical Methods in Medicine, **2013**, 1-27.
10. Kempson G.E., Freeman M.A.R., and Swanson S.A.V., *The determination of a creep modulus for articular cartilage from indentation tests on the human femoral head*, Journal of Biomechanics, **1971**, 4, 239-250.
11. Mow V.C., Kuei S.C., Lai W.M., and Armstrong C.G., *Biphasic creep and stress relaxation of articular cartilage in compression? Theory and experiments*, Journal Biomechanical Engineering, **1980**, 102, 73-84.
12. Thambyah A., Nather A., and Goh J., *Mechanical properties of articular cartilage covered by the meniscus*, Osteoarthritis Cartilage, **2006**, 14, 580-588.
13. Lai W.M., Hou J.S., and Mow V.C., *A triphasic theory for the swelling and deformation behaviors of articular cartilage*, Journal Biomechanical Engineering, **1991**, 113, 245-258.

14. Touraine S., Bouhadoun H., Engelke K., and Laredo J.D., *Influence of meniscus on cartilage and subchondral bone features of knees from older individuals: A cadaver study*, PLOS ONE, **2017**, 12, 1-15.

15. Sim S., Hadjab I., Garon M., and Quenneville E., *Development of an Electromechanical Grade to Assess Human Knee Articular Cartilage Quality*, Annals of Biomedical Engineering, **2017**, 45, 2410-2421.

16. Hadjab I., Sim S., Karhula S.S., and Kauppinen S., *Electromechanical properties of human osteoarthritic and asymptomatic articular cartilage are sensitive and early detectors of degeneration*, Osteoarthritis and Cartilage, **2017**, in press.

17. Marchi B.C. and Arruda E.M., *A study on the role of articular cartilage soft tissue constitutive form in models of whole knee biomechanics*, Biomechanics and Modeling in Mechanobiology, **2017**, 16, 117-138.

18. Liu B., Nimit K.L., Amber T.C., and Pramodh K.G., *In Vivo Tibial Cartilage Strains in Regions of Cartilage-to-Cartilage Contact and Cartilage-to-Meniscus Contact in Response to Walking*, The American Journal of Sports Medicine, **2017**, 45, 2817-2823.

19. Quiroga J.M.P., et al., *Relative contribution of articular cartilage's constitutive components to load support depending on strain rate*,

Biomechanics and Modeling in Mechanobiology, **2017**, 16, 151-158.

20. Edd S.N., et al., *Altered gait mechanics and elevated serum pro-inflammatory cytokines in asymptomatic patients with MRI evidence of knee cartilage loss*, Osteoarthritis and Cartilage, **2017**, 25, 899-906.
21. Soong T.T. and Huang W.N., *A stochastic model for biological tissue elasticity in simple elongation*, Journal of Biomechanics, **1973**, 6, 451-458.
22. Decraemer W.F., Maes M.A., and Vanhuyse V.J., *An elastic stress-strain relation for soft biological tissues based on a structural model*, Journal of Biomechanics, **1980**, 13, 463-468.
23. Lanir Y., *Constitutive equations for fibrous connective tissues*, Journal of Biomechanics, **1983**, 16, 1-12.
24. Fung Y.C., *Elasticity of soft tissues in simple elongation*, The American Journal of Physiology, **1967**, 213, 1532-1544.
25. Oberhofer K., et al., *The influence of muscle-tendon forces on ACL loading during jump landing: a systematic review*, Muscles, Ligaments and Tendons Journal, **2017**, **7**, 125-135.
26. Pappas E., et al., *Lessons learned from the last 20 years of ACL-related in vivo-biomechanics research of the knee joint*, Knee Surgery, Sports Traumatology, Arthroscopy, **2013**, 21, 755-766.

27. Bates N.A., et al., *Anterior Cruciate Ligament Biomechanics During Robotic and Mechanical Simulations of Physiologic and Clinical Motion Tasks: A Systematic Review and Meta-Analysis*, Clinical biomechanics, **2015**, 30, 1-13.
28. Aspden R.M., *A model for the function and failure of the meniscus*, Engineering in Medicine, **1985**, 14, 119-122.
29. Meakin J.R., et al., *Finite element analysis of the meniscus: the influence of geometry and material properties on its behaviour*, Knee, **2003**, 10, 33-41.
30. Tissakht M. and Ahmed A.M., *Parametric study using different elastic and poroelastic axisymmetric models of the femurmeniscus-tibia unit*, in *Winter Annual Meeting of the American Society of Mechanical Engineers*, **1992**, 241-243
31. Spilker R.L., Donzelli P.S., and Mow V.C., *A transversely isotropic biphasic finite element model of the meniscus*, Journal of Biomechanics, **1992**, 25, 1027-1045.
32. Renani M.S., et al., *Ulna-humerus contact mechanics: Finite element analysis and experimental measurements using a tactile pressure sensor*, Medical Engineering & Physics, **2017**, 50, 22-28.
33. Li Q., et al., *Micromechanical anisotropy and heterogeneity of the meniscus extracellular matri*, Acta Biomaterialia, **2017**, 54, 356-366.

34. Andrews S.H.J., et al., *Current concepts on structure–function relationships in the menisci*, Connective tissue research, **2017**, 58, 271-281.
35. Greybe D., et al., *A finite element model to investigate the effect of ulnar variance on distal radioulnar joint mechanics*, International Journal for Numerical Methods in Biomedical Engineering, **2017**, 33, 1-11.
36. Naghibi Beidokhti H., et al., *The influence of ligament modelling strategies on the predictive capability of finite element models of the human knee joint*, Journal of Biomechanics, **2017**, 65, 1-11.
37. Klets O., et al., *Comparison of different material models of articular cartilage in 3D computational modeling of the knee: Data from the Osteoarthritis Initiative (OAI)*, Journal of Biomechanics, **2016**, 49, 3891-3900.
38. Freutel M., et al., *Finite element modeling of soft tissues: Material models, tissue interaction and challenges*, Clinical Biomechanics, **2014**, 29, 363-372.
39. Beveridge J.E., et al., *Relationship between increased in vivo meniscal loads and abnormal tibiofemoral surface alignment in ACL deficient sheep is varied*, Journal of Biomechanics, **2016**, 49, 3824-3832.
40. Plantefève R., et al., *Patient-Specific Biomechanical Modeling for Guidance During Minimally-Invasive Hepatic Surgery*, Annals of Biomedical Engineering, **2016**, 44, 139-153.
41. Riener R., and Harders M., *Virtual Reality in Medicine*, Springer London, London, England, **2012**.

42. Akhtar K., et al., *Virtual Reality Simulators*, in *Effective Training of Arthroscopic Skills*, M. Karahan, et al., Editors, Springer Berlin Heidelberg, Berlin, Heidelberg, **2015**, 71-80.
43. Ziegler R., et al., *Virtual reality arthroscopy training simulator*, Computers in Biology and Medicine, **1995**, 25, 193-203.
44. Gibson S., et al., *Simulating arthroscopic knee surgery using volumetric object representations, real-time volume rendering and haptic feedback*, in *Proceedings of the First Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medial Robotics and Computer-Assisted Surgery*, Springer Berlin Heidelberg, Berlin, Heidelberg, **1997**, 369-378.
45. Gibson S., *3D chainmail: a fast algorithm for deforming volumetric objects*, in *Proceedings of the 1997 symposium on Interactive 3D graphics*, Rhode Island, United States, **1997**, 149-156.
46. Ward J.W., et al., *The development of an arthroscopic surgical simulator with haptic feedback*, Future Generation Computer Systems, **1998**, 14, 243-251.
47. Megali G., et al., *A New Tool for Surgical Training in Knee Arthroscopy*, in *Medical Image Computing and Computer-Assisted Intervention — MICCAI 2002*, T. Dohi and R. Kikinis, Editors, Springer Berlin Heidelberg, **2002**, 170-177.

48. Heng P.A., et al., *A virtual-reality training system for knee arthroscopic surgery*, IEEE Transactions on Information Technology in Biomedicine, **2004**, 8, 217-227.
49. Ganapathy S. and Dennehy T.G., *A new general triangulation method for planar contours*, SIGGRAPH Computer Graphics, **1982**, 16, 69-75.
50. Yang X., Heng P.A., and Tang Z., *Constrained Tetrahedral Mesh Generation of Human Organs on Segmented Volume*, in *International Conference on Diagnostic Imaging and Analysis*, Shanghai, China, **2002**.
51. Kühnapfel U., Çakmak H.K., and Maaß H., *Endoscopic surgery training using virtual reality and deformable tissue simulation*, Computers & Graphics, **2000**, 24, 671-682.
52. Cakmak H., Maass H., and Kuhnapfel U., *VSOne, a virtual reality simulator for laparoscopic surgery*, Minim Invasive Ther Allied Technol, **2005**, 14, 134-144.
53. Lu J., et al. *A knee arthroscopy simulator for partial meniscectomy training*, in 7th *Asian Control Conference*, **2009**, 763-767.
54. Lyu S.R., et al., *Experience-based virtual training system for knee arthroscopic inspection*, Biomed Eng Online, **2013**, 12, 63.
55. Rasool S., et al., *Towards hand-eye coordination training in virtual knee arthroscopy*, in *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology*, Singapore, **2013**, 17-26.

56. Wang Y., et al., *vKASS: a surgical procedure simulation system for arthroscopic anterior cruciate ligament reconstruction*, Computer Animation and Virtual Worlds, **2013**, 24, 25-41.
57. Bayonat S., et al., *Shoulder Arthroscopy Training System with Force Feedback*, in *International Conference on Medical Information Visualisation - BioMedical Visualisation*, **2006**, 11-16.
58. Yaacoub F., Hamam Y., and Abche A., *Computer-Based Training System for Simulating Wrist Arthroscopy*, in *21st IEEE International Symposium on Computer-Based Medical Systems*, **2008**, 421-423.
59. http://www.simendo.eu/index.php/simendo-arthroscopy/.
60. Tuijthof G., et al., *Does Perception of Usefulness of Arthroscopic Simulators Differ with Levels of Experience?* Clinical Orthopaedics and Related Research, **2011**, 469, 1701-1708.
61. https://www.toltech.net/medical-simulators/products/arthrosim-arthroscopy-simulator.
62. http://simbionix.com/simulators/arthro-mentor/.
63. http://www.virtamed.com/products/arthros.
64. Garfjeld Roberts P., et al., *Validation of the updated ArthroS simulator: face and construct validity of a passive haptic virtual reality simulator with novel performance metrics*, Knee Surgery, Sports Traumatology, Arthroscopy, **2017**, 25, 616-625.

65. Tofte J., et al., *Knee, Shoulder, and Fundamentals of Arthroscopic Surgery Training: Validation of a Virtual Arthroscopy Simulator*, Arthroscopy, **2017**, 33, 641-646.
66. Rahm S., et al., *Validation of a virtual reality-based simulator for shoulder arthroscopy*, Knee Surgery, Sports Traumatology, Arthroscopy, **2016**, 24, 1730-1737.
67. Martin K.D., et al., *Comparison of Three Virtual Reality Arthroscopic Simulators as Part of an Orthopedic Residency Educational Curriculum*, The Iowa Orthopaedic Journal, **2016**, 36, 20-25.
68. Vaughan N., et al., *A review of virtual reality based training simulators for orthopaedic surgery*, Medical Engineering & Physics, **2016**, 38, 59-71.
69. Li L., et al., *Application of virtual reality technology in clinical medicine*, American Journal of Translational Research, **2017**, 9, 3867-3880.
70. Nealen A., et al., *Physically Based Deformable Models in Computer Graphics*, Computer Graphics Forum, **2006**, 25, 809-836.
71. Misra S., Ramesh K., and Okamura A., *Modeling of tool-tissue interactions for computer-based surgical simulation: a literature review*, Presence: Teleoperators and Virtual Environments, **2008**, 17, 463-491.
72. Meier U., et al., *Real-time deformable models for surgery simulation: a survey*, Computer Methods and Programs in Biomedicine, **2005**, 77, 183-197.

73. Famaey N. and Sloten J.V., *Soft tissue modelling for applications in virtual surgery and surgical robotics*, Computer Methods in Biomechanics and Biomedical Engineering, **2008**, 11, 351-366.
74. Zhang J., et al., *Deformable Models for Surgical Simulation: A Survey*, IEEE Reviews in Biomedical Engineering, **2017**, 99, 1-22.
75. Sifakis E. and Barbic J., *FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction*, in *ACM SIGGRAPH 2012 Courses*, Los Angeles, California, United States, **2012**, 1-50.
76. Muller M, et al., *Real time physics: class notes*, in *ACM SIGGRAPH 2008 classes*, ACM, Los Angeles, California, United States, **2008**, 1-90.
77. Piegl L. and Tiller W., *The NURBS Book*, Monographs in Visual Communication, Springer Berlin Heidelberg, Berlin, Heidelberg, **1997**, Second Edition.
78. Hu S.M., et al., *Modifying the shape of NURBS surfaces with geometric constraints*, Computer-Aided Design, **2001**, **33**, 903-912.
79. Wang Y., and Zheng J., *Control point removal algorithm for t-spline surfaces*, in *Proceedings of the 4th international conference on Geometric Modeling and Processing*, Springer-Verlag, Pittsburgh, **2006**, 385-396.
80. Sederberg, T.W., et al., *T-spline Simplification and Local Refinement*, ACM Transactions on Graphics, **2004**, 23, 276-283.
81. Sederberg, T.W., et al., *T-splines and T-NURCCs*, ACM Transactions on Graphics, **2003**, 22, 477-484.

82. Sederberg, T.W. and Parry S.R., *Free-form deformation of solid geometric models.* SIGGRAPH Computer Graphics, **1986**, 20, 151-160.
83. Gain, J. and Bechmann D., *A survey of spatial deformation from a user-centered perspective.* ACM Transactions on Graphics, **2008**, 27, 1-21.
84. Sorkine, O., et al., *Laplacian surface editing*, in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, ACM, Nice, France, **2004**, 175-184.
85. Yu, Y., et al., *Mesh editing with poisson-based gradient field manipulation.* ACM Transactions on Graphics, **2004**, 23, 644-651.
86. Kot, M. and Nagahashi H., *Mass spring models with adjustable Poisson's ratio*, The Visual Computer, **2017**, 33, 283-291.
87. Duan, Y., et al., *Volume Preserved Mass-Spring Model with Novel Constraints for Soft Tissue Deformation*, IEEE Journal of Biomedical and Health Informatics, **2016**, 20, 268-280.
88. San-Vicente, G., Aguinaga I., and Celigueta J.T., *Cubical Mass-Spring Model Design Based on a Tensile Deformation Test and Nonlinear Material Model*, IEEE Transactions on Visualization and Computer Graphics, **2012**, 18, 228-241.
89. Basafa, E. and Farahmand F., *Real-time simulation of the nonlinear visco-elastic deformations of soft tissues*, International Journal of Computer Assisted Radiology and Surgery, **2011**, 6, 297-307.

90. Qin, J., et al., *A Novel Modeling Framework for Multilayered Soft Tissue Deformation in Virtual Orthopedic Surgery*, Journal of Medical Systems, **2010**, 34, 261-271.
91. Omar, N., et al., *Local deformation for soft tissue simulation*, Bioengineered, **2016**, **7**, 291-297.
92. Muller, M., et al., *Position based dynamics*, Journal of Visual Comunication Image Representation, **2007**, 18, 109-118.
93. Bender, J., et al., *A Survey on Position-Based Simulation Methods in Computer Graphics*, Computer Graphics Forum, **2014**, 33, 228-251.
94. Bender, J., M. Muller and Miles M., *A Survey on Position Based Dynamics*, in *Eurographics 2017*, The Eurographics Association, **2017**, 1-31
95. Bender, J., Weber D., and Diziol R., *Fast and stable cloth simulation based on multi-resolution shape matching*, Computers & Graphics, **2013**, 37, 945-954.
96. Muller, M. and Chentanez N., *Wrinkle meshes*, in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Madrid, Spain, **2010**, 85-92.
97. Macklin, M. and Muller M., *Position based fluids*, ACM Transactions on Graphics, **2013**, 32, 1-12.
98. Takahashi, T., et al., *Volume preserving viscoelastic fluids with large*

deformations using position-based velocity corrections, The Visual Computer, **2016**, 32, 57-66.

99. Berndt, I., Torchelsen R., and Maciel A., *Efficient Surgical Cutting with Position-Based Dynamics*, IEEE Computer Graphics And Applications, **2017**, 38, 24-31.
100. Pan, J., et al., *Real-time dissection of organs via hybrid coupling of geometric metaballs and physics-centric mesh-free method*, The Visual Computer, **2016**, 32, 1-12.
101. Sui, Y., et al., *Real-time simulation of soft tissue deformation and electrocautery procedures in laparoscopic rectal cancer radical surgery*, The International Journal of Medical Robotics and Computer Assisted Surgery, **2017**, 13, 1-12.
102. Bro-Nielsen, M., *Finite element modeling in surgery simulation*, Proceedings of the IEEE, **1998**, 86, 490-503.
103. Tang, W. and Wan T.R., *Constraint-Based Soft Tissue Simulation for Virtual Surgical Training*, IEEE Transactions on Biomedical Engineering, **2014**, 61, 2698-2706.
104. Taylor, Z.A., et al., *On modelling of anisotropic viscoelasticity for soft tissue simulation: Numerical solution and GPU execution*, Medical Image

Analysis, **2009**, 13, 234-244.

105. Joldes, G.R., Wittek A., and Miller K., *Real-time nonlinear finite element computations on GPU – Application to neurosurgical simulation*, Computer Methods in Applied Mechanics and Engineering, **2010**, 199, 3305-3314.

106. Mafi, R. and Sirouspour S., *GPU-based acceleration of computations in nonlinear finite element deformation analysis*, International Journal for Numerical Methods in Biomedical Engineering, **2014**, 30, 365-381.

107. Courtecuisse, H., et al., *GPU-based real-time soft tissue deformation with cutting and haptic feedback*, Progress in Biophysics and Molecular Biology, **2010**, 103, 159-168.

108. Muller, M., et al., *Stable real-time deformations*, in *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, San Antonio, Texas, United States, **2002**, 49-54.

109. Jia, S., et al., *CPU–GPU Parallel Framework for Real-Time Interactive Cutting of Adaptive Octree-Based Deformable Objects*, Computer Graphics Forum, **2017**, in press.

110. Phuoc Bui, H., et al., *Real-time Error Control for Surgical Simulation.* IEEE Transactions on Biomedical Engineering, **2017**, 1-12.

111. Nesme, M., et al., *Physically realistic interactive simulation for biological soft tissues.* Recent Research Developments in Biomechanics, **2005**, 1-22.

112. Phuoc, B.H., et al., *Controlling the error on target motion through real-time mesh adaptation: Applications to deep brain stimulation.* International Journal for Numerical Methods in Biomedical Engineering, **2018**, 29-58.

113. Otaduy, M.A., et al., *Data-driven simulation methods in computer graphics: cloth, tissue and faces*, in *ACM SIGGRAPH 2012 Courses*, ACM, Los Angeles, California, **2012**, 1-96.

114. Haouchine, N., et al., *Monocular 3D Reconstruction and Augmentation of Elastic Surfaces with Self-occlusion Handling.* IEEE Transactions on Visualization and Computer Graphics, 2015, 1-14.

115. Pons Moll, G., et al., *ClothCap*, ACM transactions on graphics, **2017**, 36, 1-15.

116. Zou, Y. and Liu P.X., *A high-resolution model for soft tissue deformation based on point primitives*, Computer Methods and Programs in Biomedicine, **2017**, 148, 113-121.

117. Kadle, P., et al., *Reconstructing personalized anatomical models for physics-based body animation*, ACM transactions on graphics, **2016**, 35,

1-13.

118. Elahi, S.A., N. Connesson, and Y. Payan, *Disposable system for in-vivo mechanical characterization of soft tissues based on volume measurement.* Journal of Mechanics in Medicine and Biology, **2018**, 1-17.

119. Seiler, M., Spillmann J., and Harders M., *Data-Driven Simulation of Detailed Surface Deformations for Surgery Training Simulators*, IEEE Transactions on Visualization and Computer Graphics, **2014**, 20, 1379-1391.

120. Yang, Y., et al., *Boundary-Aware Multidomain Subspace Deformation.* IEEE Transactions on Visualization and Computer Graphics, **2013**, 19, 1633-1645.

121. Barbi, J. and Zhao Y., *Real-time large-deformation substructuring*. ACM Transactions on Graphics, **2011**, 30, 1-8.

122. Harmon, D. and Zorin D., *Subspace integration with local deformations*, ACM Transactions on Graphics, **2013**, 32, 1-10.

123. Yu, D. and Kanai T., *Data-driven subspace enrichment for elastic deformations with collisions*, The Visual Computer, **2017**, 33, 779-788.

124. Xu, H. and Barbi J., *Pose-space subspace dynamics.* ACM Transactions on Graphics, **2016**, 35, 1-14.

125. Von Radziewsky, P., et al., *Optimized subspaces for deformation-based modeling and shape interpolation*, Computers & Graphics, **2016**, 58, 128-138.

126. Mukherjee, R., Wu X., and Wang H., *Incremental Deformation Subspace Reconstruction*, Computer Graphics Forum, **2016**, 35, 169-178.

127. Kerfriden, P., et al., *A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics.* Computer Methods in Applied Mechanics and Engineering, **2013**, 169-188.

128. Kerfriden, P., J.C. Passieux, and S.P.A. Bordas, *Local/global model order reduction strategy for the simulation of quasi-brittle fracture.* International Journal for Numerical Methods in Engineering, **2012**, 154-179.

129. Rohmer, D., et al., *Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles*, ACM Transactions on Graphics, **2010**, 29, 1-8.

130. Kang, M.K. and Lee J., *A real-time cloth draping simulation algorithm using conjugate harmonic functions*, Computers & Graphics, **2007**, 31, 271-279.

131. Milliez, A., et al., *Programmable Animation Texturing using Motion Stamps*, Computer graphics forum, **2016**, 35, 67-75.

132. Jing, M., He B., and Lv Y., *Cloth Wrinkle Enhancement Based on the Coarse Mesh Simulation Inclusive Smart Cities and Digital Health*, Lecture Notes in Computer Science, **2016**, 9677, 293-301.

133. Patkar, S., Jin N., and Fedkiw R., *A new sharp-crease bending element for folding and wrinkling surfaces and volumes*, in *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, ACM, Los Angeles, California, **2015**, 7-15.

134. Kim, H.J., et al., *Interactive Generation of Realistic Facial Wrinkles from Sketchy Drawings*, Computer Graphics Forum, **2015**, 34, 179-191.

135. Berkiten, S., et al., *Learning Detail Transfer based on Geometric Features*, Computer Graphics Forum, **2017**, 36, 361-373.

136. Wu, J., Westermann R., and Dick C., *A Survey of Physically Based Simulation of Cuts in Deformable Bodies*, Computer Graphics Forum, 2015, 34, 161-187.

137. Choi, A.R. and Sung M.Y., *Performance improvement of haptic collision detection using subdivision surface and sphere clustering*, PLOS ONE, **2017**, 12, 1-17.

138. Xu, H. and Barbič J., *6-DoF Haptic Rendering Using Continuous Collision Detection between Points and Signed Distance Fields*, IEEE Transactions on Haptics, **2017**, 10, 151-161.

139. Wu, J., Dick C., and Westermann R., *Efficient collision detection for composite finite element simulation of cuts in deformable bodies*, The Visual Computer, **2013**, 29, 739-749.

140. Spillmann, J. and Harders M., *Robust Interactive Collision Handling between Tools and Thin Volumetric Objects*, IEEE Transactions on Visualization and Computer Graphics, **2012**, 18, 1241-1254.

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

142. Courtecuisse, H., et al., *Real-time simulation of contact and cutting of heterogeneous soft-tissues*, Medical Image Analysis, **2014**, 18, 394-410.

143. Bielser, D., Maiwald V.A., and Gross M.H., *Interactive Cuts through 3-Dimensional Soft Tissue*, Computer Graphics Forum, **1999**, 18, 31-38.

144. Nienhuys, H.W. and Frank van der Stappen A., *A Surgery Simulation Supporting Cuts and Finite Element Deformation*, in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001*, W. Niessen and M. Viergever, Editors, Springer Berlin Heidelberg. **2001**, 145-152.

145. Muller, M. and Gross M., *Interactive virtual materials*, in *Proceedings of Graphics Interface 2004*, Canadian Human-Computer Communications Society, London, Ontario, Canada, **2004**, 239-246.

146. Lindblad, A. and Turkiyyah G., *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, Beijing, China, **2007**, 421-429.

147. Lim, Y.J., Jin W., and De S., *On Some Recent Advances in Multimodal Surgery Simulation: A Hybrid Approach to Surgical Cutting and the Use of Video Images for Enhanced Realism*, Presence, **2007**, 16, 563-583.

148. Steinemann, D., et al, *Hybrid Cutting of Deformable Solids*, in *Virtual Reality Conference,* **2006**, 35-42.

149. Paulus, C., et al., *Virtual cutting of deformable objects based on efficient topological operations*, The Visual Computer, **2015**, 31, 831-841.

150. Molino, N., Bao Z., and Fedkiw R., *A virtual node algorithm for changing mesh topology during simulation,* ACM Transactions on Graphics, **2004**, 23, 385-392.

151. Sifakis, E., Der K.G., and Fedkiw R., *Arbitrary cutting of deformable tetrahedralized objects,* in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, San Diego, California, **2007**, 73-80.

152. Jia, S., et al., *CPU–GPU mixed implementation of virtual node method for real-time interactive cutting of deformable objects using OpenCL*, International Journal of Computer Assisted Radiology and Surgery, **2015**, 10, 1-15.

153. Nesme, M., et al., *Preserving topology and elasticity for embedded deformable models*, ACM Transactions on Graphics, **2009**, 28, 1-9.

154. Jeřábková, L., et al., *Volumetric modeling and interactive cutting of deformable bodies*, Progress in Biophysics and Molecular Biology, **2010**, 103, 217-224.

155. Dick, C., Georgii J., and Westermann R., *A Hexahedral Multigrid Approach for Simulating Cuts in Deformable Objects*, IEEE Transactions on Visualization and Computer Graphics, **2011**, 17, 1663-1675.

156. Wu, J., Dick C., and Westermann R., *Interactive High-Resolution Boundary Surfaces for Deformable Bodies with Changing Topology*. in *Workshop on Virtual Reality Interaction and Physical Simulation*, The Eurographics Association, **2011**, 29-38.

157. Seiler, M., Spillmann J., and Harders M., *A Threefold Representation for the Adaptive Simulation of Embedded Deformable Objects in Contact*, Journal of WSCG, **2010**, 18, 89-96.

158. Seiler, M., et al., *Robust interactive cutting based on an adaptive octree simulation mesh*, The Visual Computer, **2011**, 27, 519-529.

159. Manteaux, P.L., et al., *Interactive detailed cutting of thin sheets*, in *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, ACM, Paris, France, **2015**, 125-132.

160. Weng, B. and Sourin A., *Virtual Meniscus Examination in Knee Arthroscopy Training*, in *28th Annual Conference on Computer Animation and Social Agents*, **2015**, Singapore, 1-4.

161. Baraff, D. and A. Witkin, *Large steps in cloth simulation*, in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, **1998**, 43-54.

162. McDermott, I.D., S.D. Masouros, and A.A. Amis, *Biomechanics of the menisci of the knee*. Current Orthopaedics, **2008**, 193-201.

163. Weng, B. and Sourin A., *Interactive Cutting of Thin Deformable Objects*,

Symmetry, **2018**, 10, 1-22.

164. https://www.autodesk.com/products/3ds-max/overview.

165. Xin, S.Q. and Wang G.J., *Improving Chen and Han's algorithm on the discrete geodesic problem*, ACM Transactions on Graphics, **2009**, 28, 1-8.

166. Floater, M.S., *Mean value coordinates*, Computer Aided Geometric Design, **2003**, 20, 19-27.

167. Schneider, P.J. and Eberly D.H., *CHAPTER 7 - INTERSECTION IN 2D*, in *Geometric Tools for Computer Graphics*, Morgan Kaufmann, San Francisco, **2003**, 241-284.

168. Yang, C., et al., *Real-time physical deformation and cutting of heterogeneous objects via hybrid coupling of meshless approach and finite element method*, Computer Animation and Virtual Worlds, **2014**, 25, 423-435.

169. Saupin, G., Duriez C., and Cotin S., *Contact Model for Haptic Medical Simulations*, in *Proceedings of the 4th international symposium on Biomedical Simulation*, Springer-Verlag, London, UK, **2008**, 157-165.

170. Sin, F.S., D. Schroeder, and Barbic J., *Vega: Non-Linear FEM Deformable Object Simulator*, Computer Graphics Forum, **2013**, 32, 36-48.

171. Teran, J., et al., *Robust quasistatic finite elements and flesh simulation*, in

Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, ACM, Los Angeles, California, **2005**, 181-190.

172. Barbic, J., *Real-time reduced large-deformation models and distributed contact for computer graphics and haptics*, Carnegie Mellon University, **2007**, 1-184.

173. Baraff, D. and Witkin A., *Large steps in cloth simulation*, in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, **1998**, 43-54.

174. Maas, S.A., et al., *FEBio: Finite Elements for Biomechanics.* Journal of Biomechanical Engineering, **2012**, 134, 1-10.

175. Maas, S.A., G.A. Ateshian, and J.A. Weiss, *FEBio: History and Advances.* Annual Review of Biomedical Engineering, **2017**, 19, 279-299.

176. Johnsen, S.F., et al., *NiftySim: A GPU-based nonlinear finite element package for simulation of soft tissue biomechanics*, International Journal of Computer Assisted Radiology and Surgery, **2015**, 10, 1077-1095.

177. http://chai3d.org/.

178. https://www.geforce.com/hardware/technology/physx.

179. Talbot, H., et al., *Surgery Training, Planning and Guidance Using the SOFA Framework*, in *Eurographics 2015*, Zurich, Switzerland, **2015**, 1-5.

180. Marchesseau, S., Chatelin S., and Delingette H., *Non linear Biomechanical Model of the Liver*, in *Biomechanics of Living Organs*, Yohan P. and Jacques O., Editors, Elsevier, **2017**, 243-265.

181. Kim, Y., et al., *Deformable mesh simulation for virtual laparoscopic cholecystectomy training*, The Visual Computer, **2014**, 1-11.

Appendix I: List of Publications

Journal Papers

1. **Bin Weng**, Alexei Sourin, Interactive Cutting of Thin Deformable Objects. Symmetry, MDPI, 2018. 10(1). Impact factor 1.457

Conference Papers

2. **Bin Weng**, Alexei Sourin, Towards Meniscus Elasticity Simulation in Virtual Knee Arthroscopy. 2015 7th International Conference on Multimedia, Computer Graphics and Broadcasting (MulGraB), 2015: p. 22-27.
3. **Bin Weng**, Alexei Sourin, Virtual Meniscus Examination in Knee Arthroscopy Training. in 28th Annual Conference on Computer Animation and Social Agents (CASA2015). 2015. Singapore.
4. Shahzad Rasool, Alexei Sourin, Pingjun Xia, **Bin Weng**, Fareed Kagda, Towards hand-eye coordination training in virtual knee arthroscopy, in Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology. 2013, ACM: Singapore. p. 17-26.