DEVELOPMENT OF A VIRTUAL REALITY BASED MICROSURGERY TRAINER

WANG FEI

NATIONAL UNIVERSITY OF SINGAPORE

2004

DEVELOPMENT OF A VIRTUAL REALITY BASED MICROSURGERY TRAINER

WANG FEI

(B.Eng, Shanghai Jiao Tong University)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgements

I wish to express my sincerely gratitude and appreciation to my two supervisors, Dr. TEO Chee Leong and Dr. ETIENNE Burdet for their continuous supervision and personal encouragement along my research. I greatly respect their inspiration, professional dedication and attitude on teaching and research.

I would like to thank Dr. LIM Kian Meng, Dr. Tim Poston, Dr. LIM Beng Hai, Ankur DHANIK, James RAPPEL, Roger GASSERT, TEE Keng Peng and TEO Cheng Yong William for their collaboration and help on the micromanipulation learning project.

My gratitude also goes to Mr. Yee, Mrs. Ooi, Ms. Tshin and all the stuff in Control and Mechantronics Laboratory for the helps on facility support.

I gratefully acknowledge the financial support provided by the National University of Singapore through Research Scholarship and project funding that makes it possible for me to study and progress my research.

Table of contents

Summary

This thesis describes the development of a Virtual Reality (VR) based training system for micromanipulation in collaboration with the National University Hospital in Singapore. Microsurgery is currently trained on mocks, animals or cadavers, and by observing and assisting experienced colleagues. If the training is inefficient it is hard to identify problems; for safety, cost and ethical reasons (less training with animals), one cannot try all possible strategies. It is expected that VR system will enable more effective and systematic training, and to test the future surgeons.

While conventional approaches judge a virtual environment by its resemblance to the real environment, we address only efficiency in learning. We train simple dexterity primitives corresponding to selected features of the complex task, and develop fast algorithms to systematically investigate suitable multisensory learning cues.

For the modelling of soft tissue mechanics, we described a multi-scale FEM algorithm which uses elements from multiple levels in a hierarchy of mesh similar to the progressive mesh. We show that this algorithm computes fast compared to the traditional FEM algorithm with low loss of accuracy. For the needle maneuvering simulation, we introduced a method using stereographic projection to compute the distance between the curved needle and a curved vessel. This analytical algorithm was shown to be faster by orders of magnitude than numerical ones and the computation time barely increases with increasing precision. For the knot-tying simulation, we presented a physics-based thread simulator that enable realistic knot tying intersection in haptic rendering rate. It has considered most of the

properties of the real thread and behaves naturally. Further work should involve the development of other training primitives and evaluation of the performance on real subjects.

List of Figures

Chapter 1

Introduction

1.1 Microsurgery Training

Micro-reconstructions due to accidents are relatively frequent. In Singapore for example there is about one case for 10'000 people every year [1]. When the patient arrives at the hospital, a primary task consists of restoring vessels (i.e., arteries, veins) and nerves that have been severed during trauma. This includes trimming the vessels (with typical diameter of 1mm) and anastomosing them together using a microscope with magnification of 10 to 20. Fine manipulation under an optical microscope requires significant learning. This is done by observing experimented surgeons, participating in case, and by practicing on cadavers and animal. This traditional training is expensive (A 40 hours basic micro vascular surgery training course costs on average $1500 [2]), inflexible (a student experimenting freely increases costs), and the learning process is hard to quantify. In contrast, virtual reality setups can create computer-controlled environments with which the trainee can interact and train a task, with negligible costs for disposables and close tracking of the performance and learning [3, 4].

1.2 Motivation

As autonomous robots can perform only highly stereotyped manipulations, most manipulation tasks must be carried out by humans. Micromanipulation is critical to many growing technologies, such as Microsurgery, Micro-electronics, MEMS, Life Sciences, and Nanotechnology, with corresponding needs for skilled manpower. Humans have excellent inference and sensorimotor capabilities, are very versatile and cost less than highly complex robotic systems. However, a human requires significant learning to manipulate objects skillfully under a microscope, for various reasons:

- The operator neither feels nor hears a tool interact with its target, and must rely entirely on visual input and proprioception to control motor activity. The force involved is generally very small, so that haptic feedback is limited.
- The micro world requires greatly modified visuomotor coordination. The image from the microscope is not only larger than the hands' environment, but also differently oriented, and has less depth of field than normal vision.
- The operator must operate indirectly, not with fingers but via tools such as forceps, scalpel, laser, and micropipette.
- For accurate control, the trainee must develop new motor strategies involving only the fine motion of the fingers. Arm movements induce too much tremor.

Currently trainees typically learn micromanipulation directly on the task, or with real material. For example, microsurgery is learned on latex sheet and then on rats (Fig. 1.1, left). One must simultaneously control tremor, find a new position for the arms, adapt to the microscope image, recognize the rat's anatomy, *etc.* If the learning is inefficient it is hard for the instructor to identify problems; for safety, cost and ethical (less training with animals) reasons, one cannot try all possible strategies.

Figure 1.1: Microsurgery learning of rats (left) may be reduced by virtual training on a workstation (right).

1.3 Methodology

We are developing a Virtual Reality based training system (Fig. 1.2) as a tool for hand microsurgeons at the National University Hospital. In contrast to many Virtual Reality trainers aiming at realistic emulation [5, 6], we are only concerned with efficiency in learning. Our approach [7] is to extract and train simple dexterity primitives corresponding to basic movements complex tasks are composed of. Studies suggest [8] that humans may form internal models of these primitives which they can combine in more complex tasks, so that when this task is decomposed and learned in simple steps than all at once, learning a complex task can be faster and performance better. This confirms practice in (for instance) typing courses, which address subtasks of keyboard use *asdf* before whole words are typed. The virtual environment corresponding to real micromanipulation would be purely visual [9, 10], as the interaction forces are generally negligible with no 'natural' haptic or audio component. In virtual training, however, we can create additional haptic or audio cues to help acquiring a skill. We are using multi-sensory learning

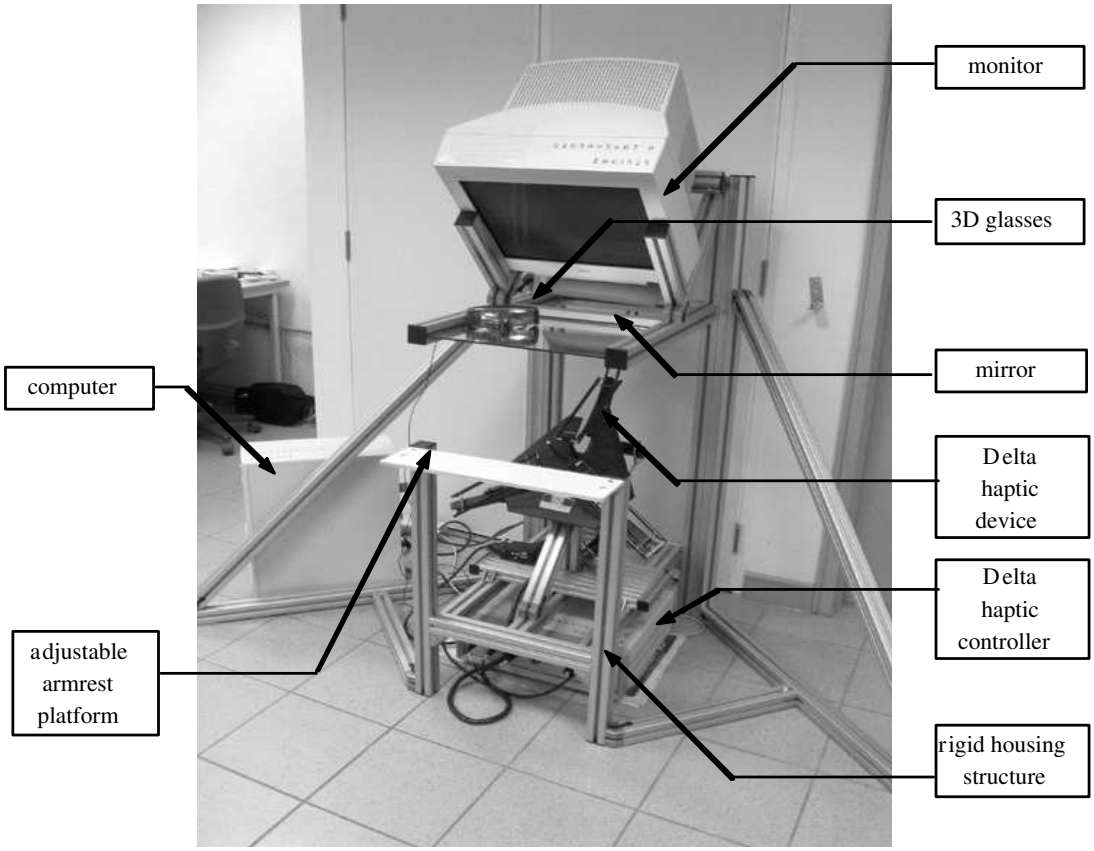

Figure 1.2: A simulation workstation with 3D view and haptic response provides intuitive manipulation and real-time interaction with virtual objects.

cues stemming from neuroscience results. In particular, recent study has shown that motion and path guidance can facilitate learning hand motion for Chinese handwriting [11] and for object handling [12].

1.4 Organization

In this thesis, we focus on the development of the tools for our VR based micro-surgery trainer. The thesis is organized as follows: in Chapter 2, we investigate the methodology of how to decompose the typical microsurgery training task to some primitive ones and investigate the principles behind; in Chapter 3, We present a multi-scale FEM algorithms to simulate the soft-tissue deformation; in Chapter 4, we introduce an analytical collision detection algorithm to measure the distance and detect collisions between a needle and a vessel; in Chapter 5, we describe

a physics based model to simulate the real-time knot-tying and we conclude the thesis in Chapter 6.

Chapter 2

Decomposition of complex tasks

Conventional VR judges a virtual environment by its resemblance to the real environment. In contrast, we address only efficiency in learning. We train simple dexterity primitives corresponding to selected features of the complex task, and systematically investigate suitable multisensory learning cues.

2.1 Multisensory learning cues

Many Virtual Reality trainers aim at realistic emulation, *e.g.,* laparoscopic surgery trainers that display the current position of the tool and shape of the organ so that the apprentice surgeon can operate virtually and feel somewhat realistic interaction forces. The virtual environment corresponding to real micromanipulation would be purely visual [9, 10], as the interaction forces are generally negligible with no 'natural' haptic or audio component. In virtual training, however, we can create additional haptic or audio cues to help acquiring a skill. These cues may be pseudo-realistic (the sound or the impact force when the tool hits a narrow tube around the intended path), or iconic (highlighting, the click of a Geiger counter, etc.) [13, 14].

As the success of a manipulation task depends on neuromechanical control of the arm during the whole motion, we propose using haptic cues in the form

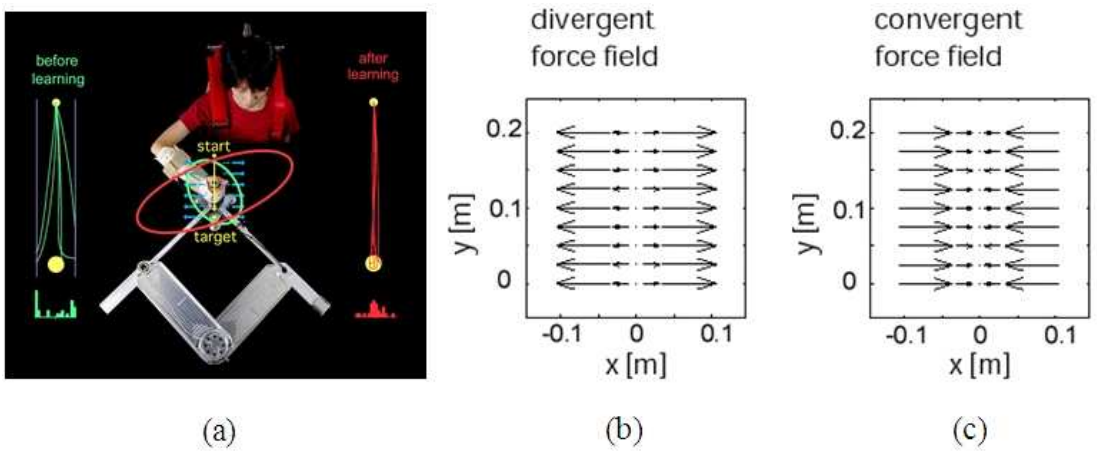

Figure 2.1: (a) By training in unstable dynamics, Burdet et. al. have found that humans adapt the feedforward motor command to decrease motion deviation [15]. Similarly, we will train micromanipulation dexterity using a divergent force field along the movement (b). Another haptic cue will be a convergent field guiding motion (c).

of movement conditions. In particular, recent study has shown that motion and path guidance can facilitate learning hand motion for Chinese handwriting [11] and for object handling [12, 16]. Burdet et. al. have also discovered [15] that humans respond to unstable dynamics with neural strategies for improved accuracy (Fig. 2.1a). We will test how these haptic cues, as well as amplified tremor, can help micromanipulation learning. Audio signals related to trajectory error relative to a desired movement will be used as audio cues.

The trainee learning a task acquires a memory or internal model [17] of this task and the conditions under which it must be performed. This internal model persists [18] even in the absence of learning cues, enabling skill transfer to a real task. This results in feedforward control, free from the long delay of visual control [3, 19], similar to the mechanisms in [15]. Neuroscience studies suggest that learning this internal model requires compensating for the delay of the motor-sensory loop [20]. As the visual loop has a particularly long delay of $200ms$ or more [21], the predictive visual cues may compensate for this delay. In one such case, we will extrapolate the error at the end of a movement from the past and present states and signal it to the trainee, allowing early correction. Could prediction made

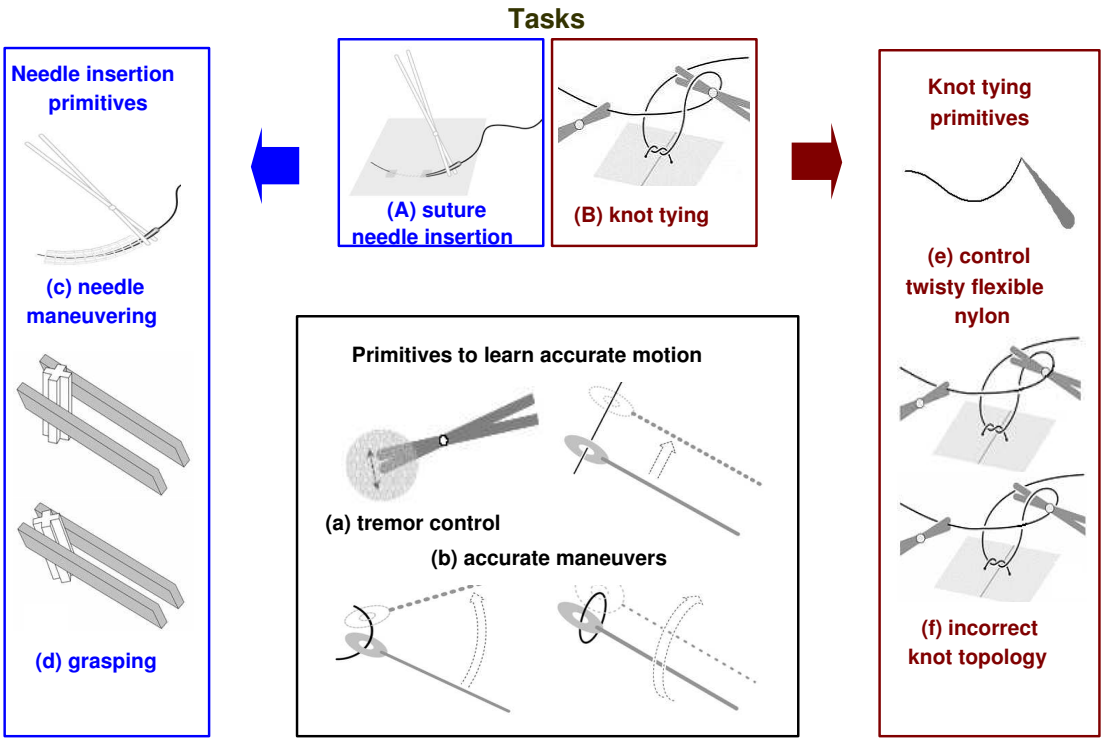

Figure 2.2: Major tasks of microvascular surgery (A,B) and underlying primitive skills (a-f).

apparent by a visual or acoustic signal be as efficient as haptic cues? As haptic feedback is relatively costly to provide, such questions are critical to the design and test of micromanipulation trainers.

2.2 Dexterity primitives

We have studied typical tasks from microassembly [22] and microsurgery and selected representative micromanipulation tasks, which require correct perception of position, orientation and also of topology (inside, outside, loop, *etc.*) in 3D, and have both motor and cognitive aspects. Fig. 2.2 shows two of the major tasks typical of microsurgery. In suture, a curved needle suturing together two membrane edges, must pass through at carefully planned places. Knot tying is a bi-manual task requiring skillful control of the nylon and knowledge of the correct topology.

- **Identify two major tasks of microvascular surgery**

- **Suture needle insertion**

  A curved needle, suturing together two membranes, must pass through at carefully planned places (Fig. 2.2A).

- **knot tying**

  A bi-manual task requiring skillful control of the nylon and knowledge of the correct topology (Fig. 2.2B).

- **Dexterity primitives for training these two tasks**

  - **Tremor control**

    While tremor probably cannot be reduced consciously, feedback lets the trainee learn a steady grip using the fine motorics of the fingers to reduce tremor (Fig. 2.2a).

  - **Accurate maneuvering of a held object**

    The trainee must pass a loop accurately along a wire in straight and turning motions, in several orientations (Fig. 2.2b).

  - **Needle maneuvering along a curved tube**

    During needle maneuvering, position and orientation change simultaneously, in six-degree-of-freedom motion (Fig. 2.2c).

  - **Grasping**

    If the jaws of a pair of forceps align poorly with the object they close on, it rotates during the grasping (Fig. 2.2d).

  - **Control of a flexible nylon**

    The trainee must grasp the nylon and create a loop matching a visual target (Fig. 2.2e).

  - **Memorize correct knot topology**

    Differentiating a topologically correct surgical knot from incorrect ones can be learned separately (Fig. 2.2f).

  - **Knot placement**

    Microsurgeons place knots evenly around the whole section, at a con-

stant distance from the end. Misplaced knots risk poor or no blood circulation, or blood leak.

From these complex tasks we have extracted simple dexterity primitives corresponding to basic movements they are composed of. Studies suggest [8] that humans may form internal models of these primitives which they can combine in more complex tasks, so that when this task is decomposed and learned in simple steps than all at once, learning a complex task can be faster and performance better. This confirms practice in (for instance) typing courses, which address subtasks of keyboard use before whole words are typed. We in particular want to train motorics separately from cognition, and simple motions one at a time. In following chapters we will present some fast algorithms, which are the tools to train the subtasks separately.

Chapter 3

Soft-tissue deformation simulation

There are several key problems in the development of a VR based surgical trainer. First, there is a need for some visual-realism. The geometric and physical models of anatomical structures must be well defined. Moreover, since all the organs in the human body are not rigid, their shape may change during an operation. The realism can be enhanced by the introduction of devices, which allows for a better immersion in the virtual world. Secondly, real-time interactions are essential. To support the user with visual feedback, a deformable model need to be updated in $30Hz$. To support haptic feedback, the interaction dynamics with tissues and organs must be computed at a frequency of at least $1kHz$. The different update rates are achieved in multi-threads programming environment, where visual thread run in $30Hz$ and haptic thread run in $1kHz$. This chapter investigates the use of multi-scale FEM to compute the dynamics of soft tissues fast and accurately.

3.1 Review of related works

The modelling of soft tissue mechanics has become an important research topic in robot-assisted surgery and surgical simulation. Much effort has aimed at the devel-

opment of real-time simulation of the physical behaviour of deformable tissue and the integration of these methods into simulators [23–27]. However, computation speed has always been a major barrier to accurate real-time tissue models for haptic feedback, whether with quasi-static [28, 29] or dynamic [30–32] models. Early developments often simulated deformable bodies using mass-spring systems, for simplicity of implementation and the low computation compared to more realistic stress-strain relations. The body is represented by a set point masses connected by elastic links and the equation of motion is written for each point mass. However, the mass and spring constants to be used in the model cannot be derived from measured properties like the shear modulus, and thus the models do not capture underlying physics properly. Moreover, since the constants are assigned heuristically, varying the level of detail (essential to adequate speed) is hard to systematize.

Realistic elasticity calculations often use the finite element method (FEM). However, it is seldom used in real-time simulation [33–35], due to its complex implementation and expensive computation. It provides more accurate description of deformation mechanics than mass-spring models, but for real-time work it needs modification for adequate accuracy while reducing the computation time.

Several fast algorithms [36–40] have recently been the focus of research in real-time simulation, especially with the need for haptic feedback whose sampling rate is much higher than that needed for visual display. Bro-Nielsen and Cotin [37] used a condensation method to reduce the number of unknowns to only the surface degrees of freedom. Others use an adaptive mesh to reduce the number of unknowns. DiMaio and Salcudean [39] used a real-time mesh refinement technique to generate a multi-grid, with distinct fine and coarse grids, for their needle insertion simulation. Wu et. al. [40] proposed a dynamic progressive mesh which is an extension of the progressive mesh concept. The idea is to generate a hierarchy of meshes off-line by collapsing mesh edges recursively, starting from of a fine mesh, and then using elements from the different meshes in the final computation. They

presented simulation results for a two levels mesh, but the technique to combine elements from multiple levels in the hierarchy need to be developed.

In this chapter, we describe a multi-scale FEM algorithm [41] which uses elements from multiple levels in a hierarchy of mesh similar to the progressive mesh. We show that this algorithm computes fast compared to the traditional FEM algorithm with low loss of accuracy. This algorithm has been integrated with a visual/haptic feedback workstation as shown in Fig. 2.

3.2 Multi-scale FEM for soft tissue simulation

The main idea of our multi-scale FEM is to dynamically use different levels of mesh, at different places (not uniformly progressing from 'coarse all over' to 'fine all over', as in multi-grid detailed solutions). We create a fine mesh in the domain and then generate coarser elements at various levels by recursively combining adjacent elements. The final mesh used to the computation is formed by choosing elements from different levels, with small elements near an area of interest (for example, where a force is applied or bending is high) and progressively coarser elements as we move away from the region of interest. Fig. 3.1 shows levels of the mesh for a rectangular domain, starting from fine (top left) to coarse (bottom middle). The properties of these 5 levels of mesh are precomputed in the beginning of the simulation and saved in memory. The multi-scale mesh for an applied line force (for example, bottom right) combines elements from the different levels.

Such a multi-scale mesh especially suits the application of surgical simulation, as the surgical tool usually applies a force on the tissues/organs only in a localized region around it. Hence small elements are placed around the contact region, as haptic feedback needs accurate deformation and force here accurately, and they can capture the large displacement gradient in that region. Progressively larger elements suffice as we move away from the tool, for the accuracy needed for the results for visual display. This multi-scale mesh is formed in real-time as the

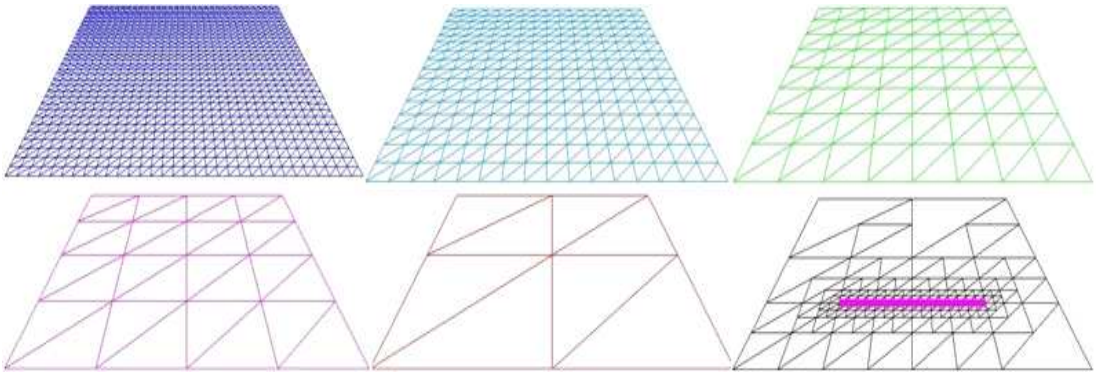

Figure 3.1: Multi-scale mesh generation shows the mesh preparation from fine (top left) to coarse (bottom middle) and the last one with assumption of suffering a line pressure.

surgical tool moves over the domain by choosing the correct elements from the various mesh levels. The stiffness matrices of the elements of each level are pre-calculated and stored at the beginning of the simulation. They are assembled in real-time to form the global stiffness matrix for the multi-scale mesh during the simulation.

Such a mesh greatly decreases the model's DOF compared to the fine mesh, and so reduces the computational time. In the example of Fig. 3.2 the fine mesh model has 961DOF which is reduced to 138DOF in the multi-mesh.

We implemented the method for the simple case of 2D out-of-plane or anti-plane strain problem. Three-dimensional motion with bending and stretching will be more complex, but can be handled by the same mesh logic.

The governing equation, with applied pressure P (in this case, is the known value of the force in right hand column of 3.7), is

$$\nabla(G\nabla w) + P = 0 \tag{3.1}$$

where G is shear modulus, ∇ Nabla-Operator and w out-of-plane displacement.

The force applied by the haptic device is modeled by a line load on the 2D domain. The case of a concentrated point force has been studied in [42], and a singularity in the deformation was observed. Such singularity does not arise with

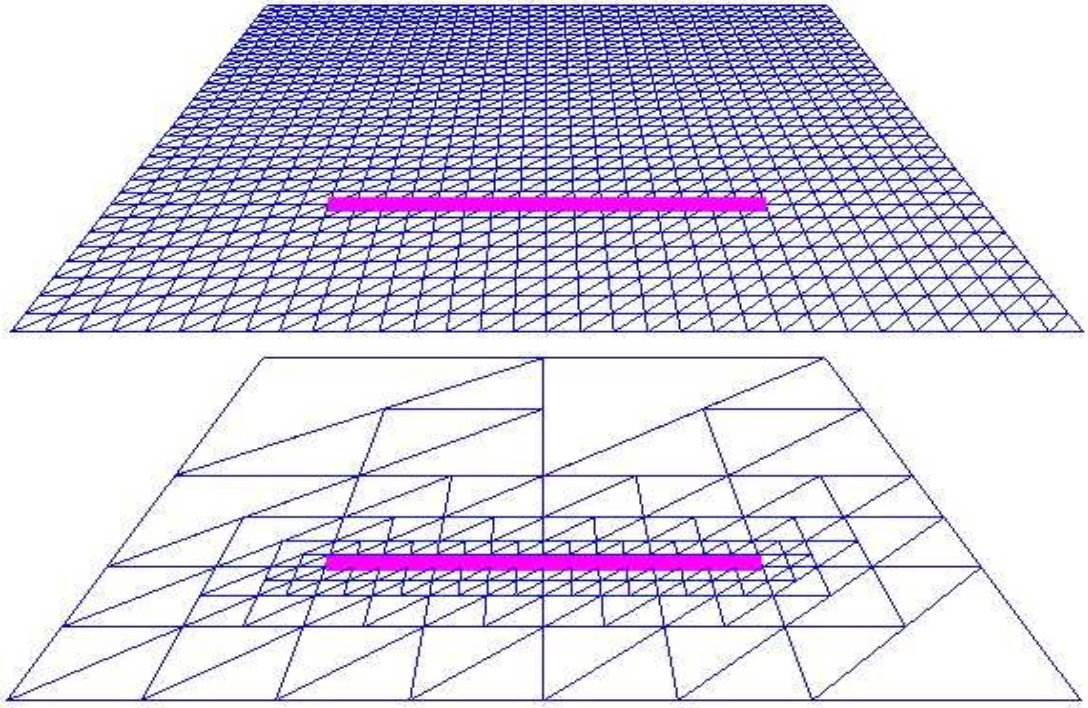

Figure 3.2: Fine mesh FEM and multi-scale mesh FEM

a line force. A line force also allows us to derive torque even for the scalar out-of-plane deformation case, around horizontal though not vertical axes, and display it using our 6DOF haptic device. The algorithm is developed to accept any input of shapes in real-time but not only restricted to a point or a line. Here we use the line force so as to compare the multi-scale method's accuracy to that of the Navier's serious solution (See Appendix for details).

With a line load applied at y=y1 and between x=x1 to x=x2, the pressure term takes the form

$$P(x,y) = F_{\partial} \prod (x, x_1, x_2) \delta(y - y_1) \tag{3.2}$$

where

$$\prod = \begin{cases} 1 & x_1 < x < x_2 \\ 0 & otherwise \end{cases} \tag{3.3}$$

δ = Dirac delta function

In the FEM formulation, the displacement in an element is approximated by

$$w(x,y) = \sum_{i=1}^{3} N^{(i)} w^{(i)} \tag{3.4}$$

Where $w^{(i)}$is the nodal displacement and $N^{(i)}$ is the shape function associated with node i. The weak form of 3.1 is

$$\iint_{\Omega} N^{(i)} \nabla (G \nabla w) d\Omega + \iint_{\Omega} N^{(i)} P d\Omega = 0 \tag{3.5}$$

with the shape functions used as weights in the domain integral. This gives a system of algebraic equations of the form

$$[K]\{U\} = \{F\} \tag{3.6}$$

where $\{U\}$ is the vector of nodal displacements, $\{F\}$ represents the applied force, and $[K]$ is the stiffness matrix of the system.

Our current implementation treats a square domain fixed so that its edges have zero displacement. We need to determine (1) the reaction force at the haptic device and (2) the displacements elsewhere given the displacement of the haptic device.

Here we describe the case with two control points, the cases of multi control points are extended in the same way. Given a linear system $[K]\{u\} = \{F\}$, the

matrix form can be written as:

$$\begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{i1} & K_{i2} & \cdots & K_{im} \\ \vdots & \vdots & \ddots & \vdots \\ K_{j1} & K_{j2} & \cdots & K_{jm} \\ \vdots & \vdots & \ddots & \vdots \\ K_{m1} & K_{m2} & \cdots & K_{mm} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{i-1} \\ U_i \\ u_{i+1} \\ \vdots \\ u_{j-1} \\ U_j \\ u_{j+1} \\ \vdots \\ u_m \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ f_i \\ 0 \\ \vdots \\ 0 \\ f_j \\ 0 \\ \vdots \\ 0 \end{Bmatrix} \tag{3.7}$$

Where U_i and U_j are the prescribed displacement at node i and j. The other displacements $(u_1, u_2, \cdots, u_{i-1}, u_{i+1}, \cdots, u_{j-1}, u_{j+1}, \cdots, u_m)$ and the force f_i, f_j applied at node i and j are unknown. Neglecting the i^{th} and j^{th} row in equation 3.7 yields:

$$\begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{(i-1)1} & K_{(i-1)2} & \cdots & K_{(i-1)m} \\ K_{(i+1)1} & K_{(i+1)2} & \cdots & K_{(i+1)m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{(j-1)1} & K_{(j-1)2} & \cdots & K_{(j-1)m} \\ K_{(j+1)1} & K_{(j+1)2} & \cdots & K_{(j+1)m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{m1} & K_{m2} & \cdots & K_{mm} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{i-1} \\ U_i \\ u_{i+1} \\ \vdots \\ u_{j-1} \\ U_j \\ u_{j+1} \\ \vdots \\ u_m \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{Bmatrix} \tag{3.8}$$

Equation 3.8 can be written as:

$$\begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{(i-1)1} & K_{(i-1)2} & \cdots & K_{(i-1)m} \\ K_{(i+1)1} & K_{(i+1)2} & \cdots & K_{(i+1)m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{(j-1)1} & K_{(j-1)2} & \cdots & K_{(j-1)m} \\ K_{(j+1)1} & K_{(j+1)2} & \cdots & K_{(j+1)m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{m1} & K_{m2} & \cdots & K_{mm} \end{bmatrix} \left\{ \left\{ \begin{matrix} u_1 \\ u_2 \\ \vdots \\ u_{i-1} \\ 0 \\ u_{i+1} \\ \vdots \\ u_{j-1} \\ 0 \\ u_{j+1} \\ \vdots \\ u_m \end{matrix} \right\} + \left\{ \begin{matrix} 0 \\ 0 \\ \vdots \\ 0 \\ U_i \\ 0 \\ \vdots \\ 0 \\ U_j \\ 0 \\ \vdots \\ 0 \end{matrix} \right\} \right\} = \left\{ \begin{matrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{matrix} \right\} \tag{3.9}$$

$$\begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{(i-1)1} & K_{(i-1)2} & \cdots & K_{(i-1)m} \\ K_{(i+1)1} & K_{(i+1)2} & \cdots & K_{(i+1)m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{(j-1)1} & K_{(j-1)2} & \cdots & K_{(j-1)m} \\ K_{(j+1)1} & K_{(j+1)2} & \cdots & K_{(j+1)m} \\ \vdots & \vdots & \ddots & \vdots \\ K_{m1} & K_{m2} & \cdots & K_{mm} \end{bmatrix} \left\{ \begin{matrix} u_1 \\ u_2 \\ \vdots \\ u_{i-1} \\ 0 \\ u_{i+1} \\ \vdots \\ u_{j-1} \\ 0 \\ u_{j+1} \\ \vdots \\ u_m \end{matrix} \right\} = - \left(U_i \left\{ \begin{matrix} K_{1i} \\ K_{2i} \\ \vdots \\ K_{(i-1)i} \\ K_{(i+1)i} \\ \vdots \\ K_{(j-1)i} \\ K_{(j+1)i} \\ \vdots \\ K_{mi} \end{matrix} \right\} + U_j \left\{ \begin{matrix} K_{1j} \\ K_{2j} \\ \vdots \\ K_{(i-1)j} \\ K_{(i+1)j} \\ \vdots \\ K_{(j-1)j} \\ K_{(j+1)j} \\ \vdots \\ K_{mj} \end{matrix} \right\} \right) \tag{3.10}$$

Neglecting the i^{th} and j^{th} column of [**K**] in equation 3.10 yields:

$$\begin{bmatrix} K_{11} & \cdots & K_{1(i-1)} & K_{1(i+1)} & \cdots & K_{1(j-1)} & K_{1(j+1)} & \cdots & K_{1m} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ K_{(i-1)1} & \cdots & K_{(i-1)(i-1)} & K_{(i-1)(i+1)} & \cdots & K_{(i-1)(j-1)} & K_{(i-1)(j+1)} & \cdots & K_{(i-1)m} \\ K_{(i+1)1} & \cdots & K_{(i+1)(i-1)} & K_{(i+1)(i+1)} & \cdots & K_{(i+1)(j-1)} & K_{(i+1)(j+1)} & \cdots & K_{(i+1)m} \\ \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ K_{(j-1)1} & \cdots & K_{(j-1)(i-1)} & K_{(j-1)(i+1)} & \cdots & K_{(j-1)(j-1)} & K_{(j-1)(j+1)} & \cdots & K_{(j-1)m} \\ K_{(j+1)1} & \cdots & K_{(j+1)(i-1)} & K_{(j+1)(i+1)} & \cdots & K_{(j+1)(j-1)} & K_{(j+1)(j+1)} & \cdots & K_{(j+1)m} \\ \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ K_{m1} & \cdots & K_{m(i-1)} & K_{m(i+1)} & \cdots & K_{m(j-1)} & K_{m(j+1)} & \cdots & K_{mm} \end{bmatrix}$$

$$\begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{i-1} \\ u_{i+1} \\ \vdots \\ u_{j-1} \\ u_{j+1} \\ \vdots \\ u_m \end{Bmatrix} = - \left(U_i \begin{Bmatrix} K_{1i} \\ K_{2i} \\ \vdots \\ K_{(i-1)i} \\ K_{(i+1)i} \\ \vdots \\ K_{(j-1)i} \\ K_{(j+1)i} \\ \vdots \\ K_{mi} \end{Bmatrix} + U_j \begin{Bmatrix} K_{1j}K_{2j} \\ \vdots \\ K_{(i-1)j} \\ K_{(i+1)j} \\ \vdots \\ K_{(j-1)j} \\ K_{(j+1)j} \\ \vdots \\ K_{mj} \end{Bmatrix} \right) \tag{3.11}$$

Then the displacements vector $(u_1, u_2, \cdots, u_{i-1}, u_{i+1}, \cdots, u_{j-1}, u_{j+1}, \cdots, u_m)$ can be solved. Let us consider the i^{th} row and j^{th} of equation 3.7, we can solve the

force f_i and f_j applied at node i and j separately:

$$[K_{i1} \quad K_{i2} \quad \cdots \quad K_{im}] \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{i-1} \\ U_i \\ u_{i+1} \\ \vdots \\ u_{j-1} \\ U_j \\ u_{j+1} \\ \vdots \\ u_m \end{Bmatrix} = f_i, \quad [K_{j1} \quad K_{j2} \quad \cdots \quad K_{jm}] \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{i-1} \\ U_i \\ u_{i+1} \\ \vdots \\ u_{j-1} \\ U_j \\ u_{j+1} \\ \vdots \\ u_m \end{Bmatrix} = f_j \tag{3.12}$$

3.3 Simulation results

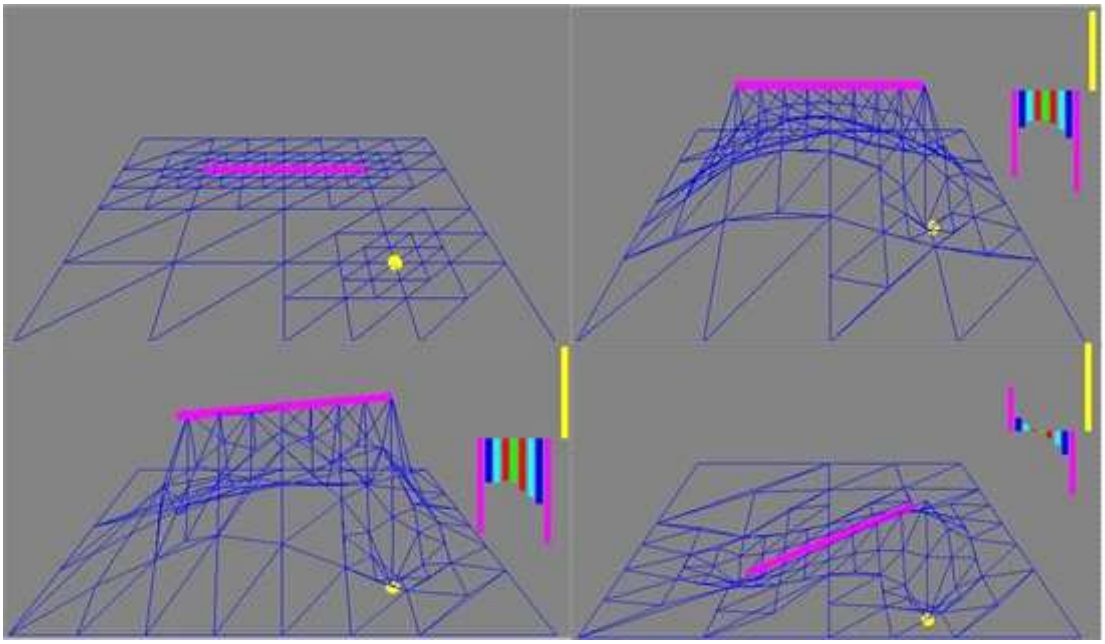

Figure 3.3: Simulation of a line force and point force applied on a rectangular soft domain

We implemented the multi-scale FEM algorithm to study its accuracy and speed. Fig. 3.3 shows four deformation states of a rectangular soft domain (blue mesh) deformed out of plane. The horizontal line and point are analogous to

microsurgery tools in contact with soft tissue, which effectively apply a line load and a concentrated force on the soft tissue. The vertical bars at right show the values and directions of the forces along the contacting area.

3.3.1 Analysis of accuracy

We compare our FEM results with Navier's series solution (See Appendix for details) to test their accuracy. The series solution is known to converge rapidly to the exact solution for a rectangular domain. We considered a 20x20cm domain pressed down 1cm by a 10cm bar, as shown in Fig. 3.4. The shear module G is 1N/cm2.

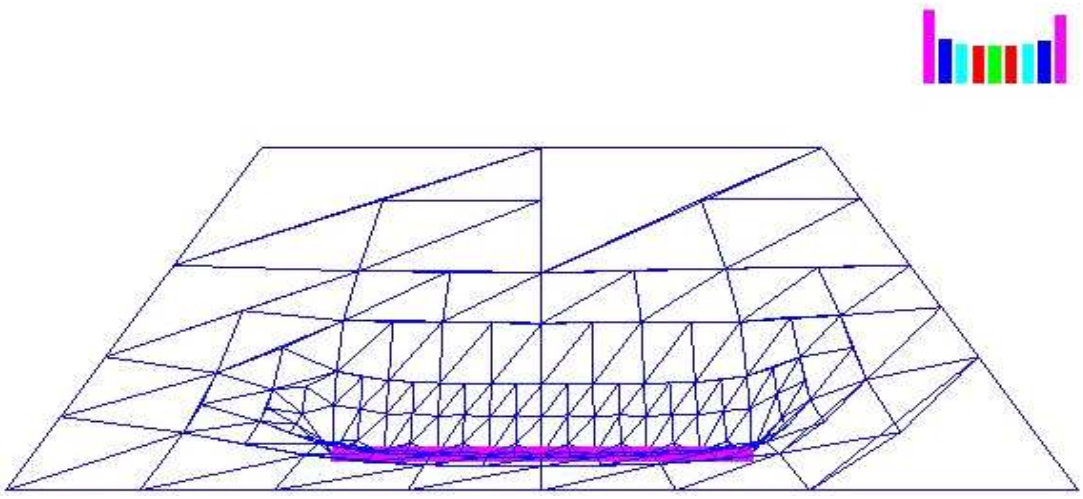

Figure 3.4: Deformation and forces acting of the 2D domain

Comparison of the calculated line force

Fig. 3.5 shows the distributed force acting along the bar obtained from the fine mesh FEM and multi-scale FEM and Navier's series solution. Forty terms are used in Navier's method to give a converged solution, the FEM results also converge to the series solution when enough levels are used. The results for the fine-mesh and multi-scale FEM are indistinguishable. The current model problem needs at least four levels for a good result. With four levels the resultant force found with

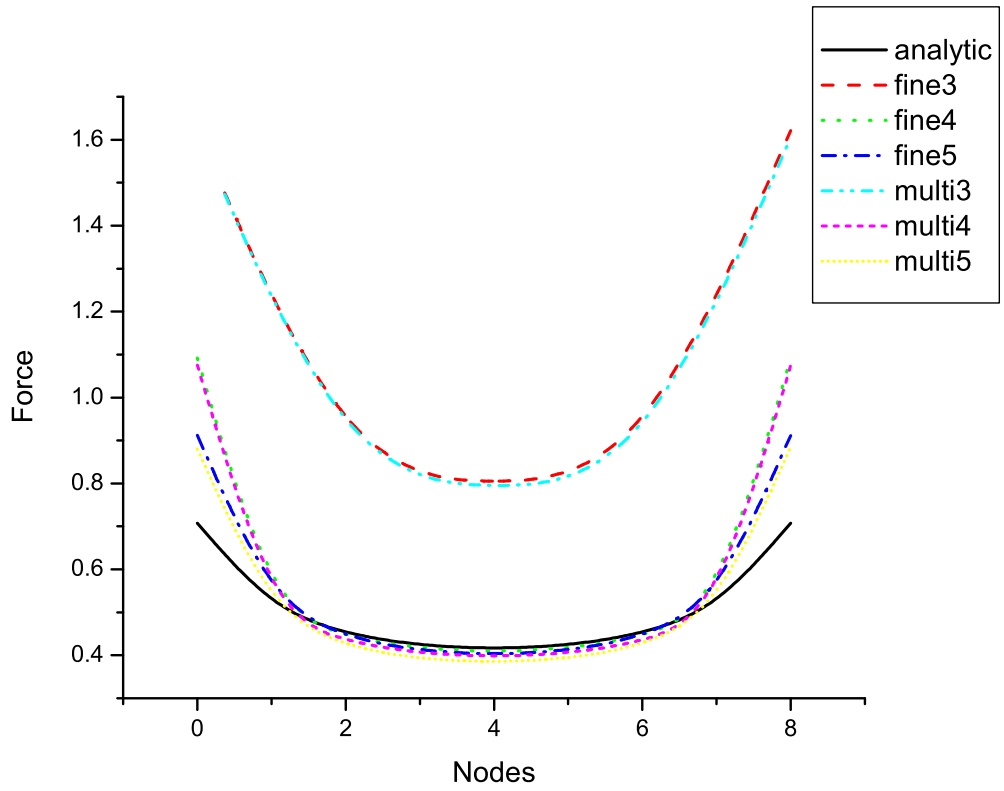

Figure 3.5: Comparison of calculated line force among analytic method, fine mesh FEM and multi-scale FEM at different levels.

the fine versus multi-scale meshes differ by only 4%.

Comparison of RMS error

Fig. 3.6 shows the Root Mean Square (RMS) error of the fine and multiscale FEMs compared to the Navier solution for two loading condition: (G1) with the bar at the center of the mesh, and (G2) with the bar near the border. The RMS errors decrease dramatically with more levels and degrees of freedom. For (G1), the multi-scale result is as good as the fine scale for up to 5 levels. For (G2), with the bar near the edge, the multi-scale mesh fails to match the accuracy of the fine mesh beyond level 4, due to the larger elements used further away from the bar. Nevertheless, the accuracy of the multi-scale mesh at these levels is about 1% to 5% of the Navier solution, normally acceptable for virtual-reality simulation where

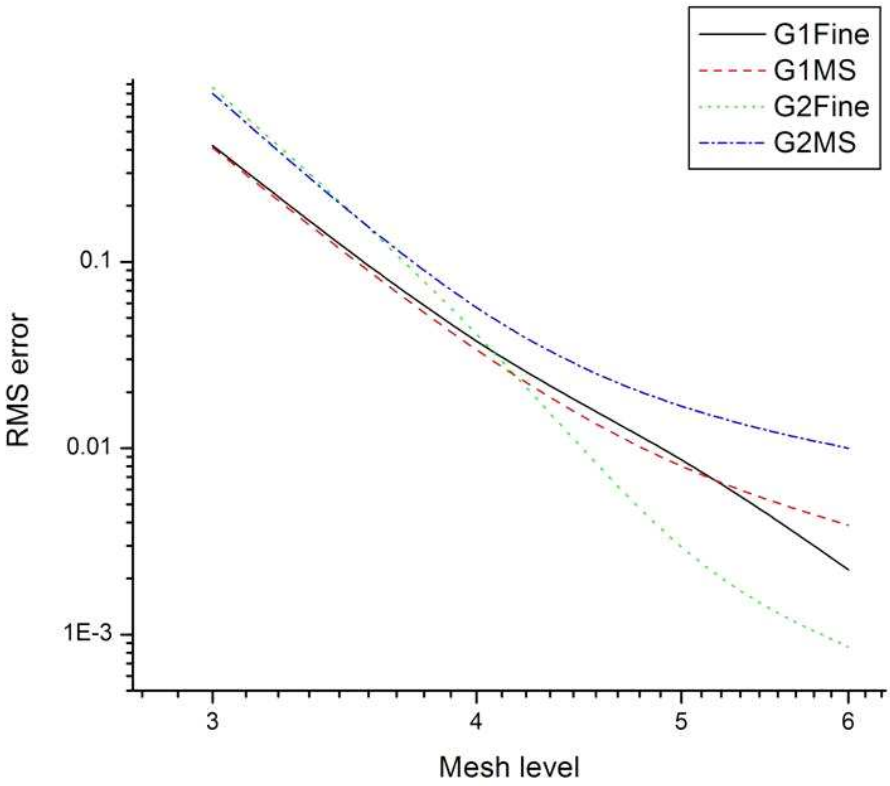

Figure 3.6: RMS(Root Mean Square) Error, by mesh level count

stability and speed outweigh high accuracy.

3.3.2 Analysis of speed

Fig. 3.7 compares CPU time using the multi-scale and fine meshes, with the count of mesh levels used. Three different cases of load position and orientation were used for multi-scale FEM method. For each case, the multi-scale FEM uses a different set of elements with a different number of degrees of freedom. This gives slightly different timing in each case. The computation time for the fine mesh FEM (included for comparison) does not change with loading case, since the number of degrees of freedom in the model is fixed.

The results show that CPU time scales roughly as $O(N^2)$ for both the fine and multi-scale mesh, where N is the number of nodes in each of the mesh. The complexity of the solution process is the same in both cases as the same iterative

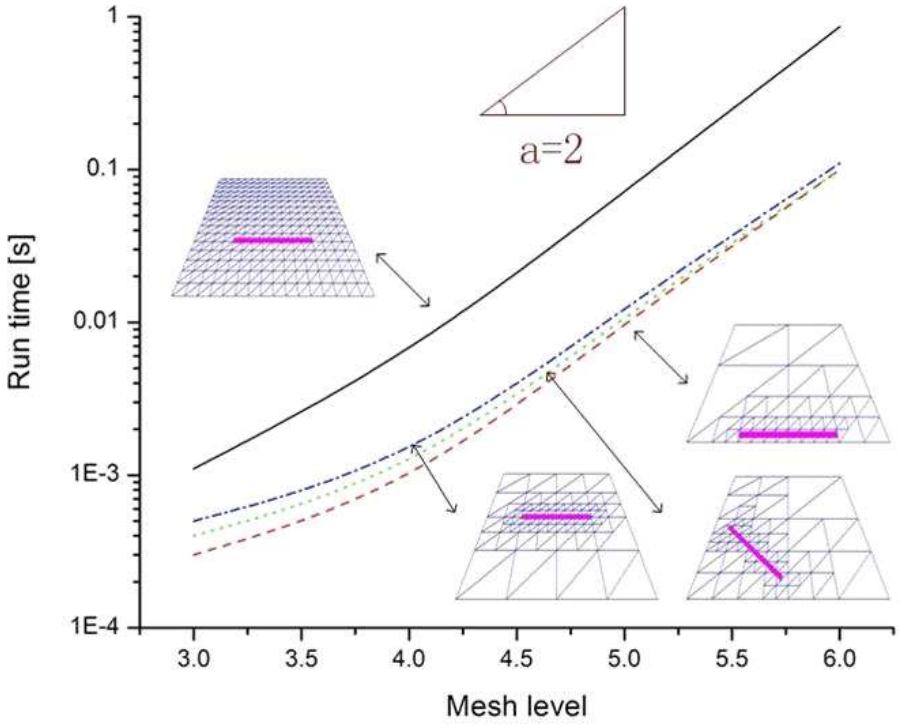

Figure 3.7: CPU time for multi-scale and fine mesh, by mesh level count.

conjugate gradient solver is used. Here, the major gain in efficiency of the multi-scale mesh is the great reduction in the number of nodal unknowns to be solved while maintaining good accuracy. The number of unknowns N in the multi-scale mesh is about 15% of that in the fine mesh. The actual number of unknowns in the multi-scale mesh compared to the fine scale mesh is difficult to predict as it depends on the location and orientation of the haptic device. Based on our study on one-dimensional rod [42], the number of unknowns in the multi-scale mesh is expected to be of the order of the logarithm of the number of unknowns, O(log N), in the fine mesh. Even with the extra time taken to assemble the global stiffness matrix in real-time, the multi-scale FEM is still about an order faster than fine-scale FEM.

For better comparison of the three cases, Fig. 3.8 shows the speed up factor of the multi-scale mesh over fine mesh FEM, defined as the ratio of their computation times. The multi-scale FEM improves the speed by 6 to 7 times at higher mesh

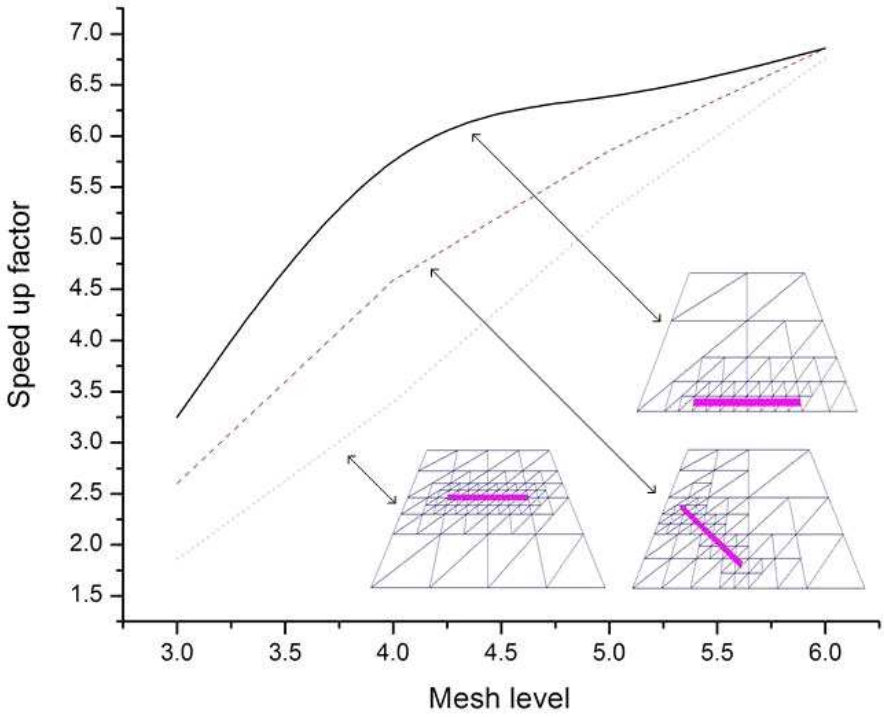

Figure 3.8: Speed up factor(Fine / MS FEM time/ in three cases, by mesh level count.)

levels.

3.4 Discussion

These results show that a major speed up in computation can be achieved using multi-scale FEM, especially at higher mesh levels with more degrees of freedom. These are necessary for more accurate solutions, within say 5% of the 'exact' one. Even with the slight accuracy loss in the multi-scale mesh compared to fine (from 0.1% to 1.5% at level 6), the 'absolute' accuracy of the multi-scale mesh is sufficient for haptic simulation. The 6-to-7-fold speed up is critical for simulation in real-time.

In our implementation (Fig. 3.9) of multi-scale FEM on a Xeon 2.66 GHz CPU with 1 GB RAM, a level 4 simulation (with about 8% accuracy) takes 1ms to

complete [41]. This was just enough to provide the refresh rate of 1 kHz required for haptic feedback. The corresponding fine scale FEM (with about 5% accuracy) cannot meet this criterion. The multi-scale FEM provides a good trade-off between a major gain in speed against a slight loss in accuracy, essential for real-time simulation.

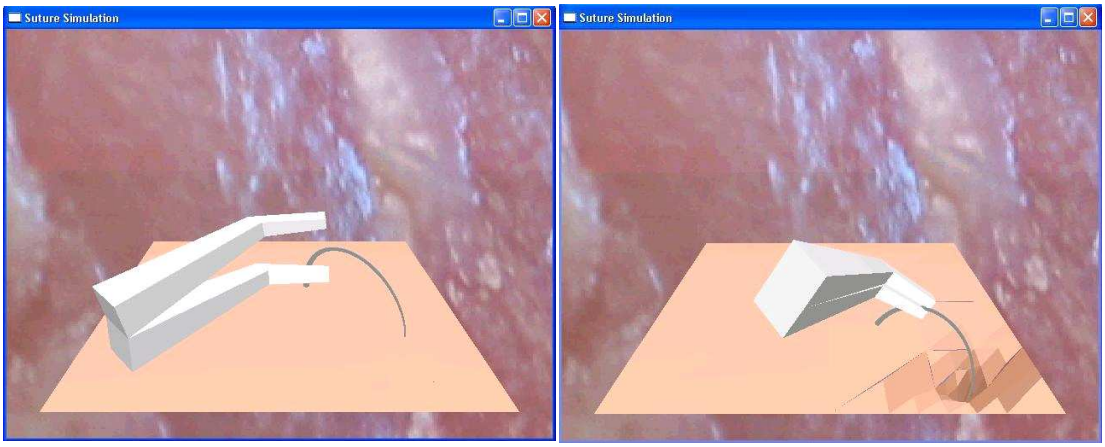

Figure 3.9: A suture simulation: intial state (left) and penetrating state (right)

In conclusion, spatially varying mesh detail in surface modeling can support large global deformation at the speeds necessary to provide visual and haptic feedback in surgical training, by a high time saving vis-á-vis traditional FEM. It can be extended for non-linear deformation of 3D bodies, and other systems of mesh mechanics. Our ultimate goal in applying it is to model human tissue. It shows great potential to realize a real-time haptic simulation system.

Chapter 4

Needle maneuvering simulation

In this chapter we focus on the modeling of suturing such as blood vessel anastomosis, performed under 5 to 20 times magnification. Although haptic feedback is negligible in real surgery, we are investigating its potential as a hyper-real cue in learning. The use of haptics implies a need for haptic speeds, so every part of the application must be tuned to take the least time possible.

To microsurgeon, needle maneuvering is a task requiring very fine control. Both position and orientation must change simultaneously, in six-degree-of-freedom motion. One dexterity skill which will be used to train this task consists of moving a curved needle precisely along its own form, so that as it passes through tissue at two points (Fig. 4.1), it does not waver at these points and tear it. This subskill is well represented by the non-surgical task of moving a curved needle along a curved tube of the same radius, without touching the wall (Fig. 4.2). This presents a more complete test of the required motor skill than does a two-point penetration. By testing the motion all along the length of the needle, it adds situational hyper-reality to the learning of the subtask.

This will be trained with multisensory cues. For example, if the curved needle is going to touch the tube, an audio warning cue may help the trainee correct the motion trajectory, with volume or pitch rising as distance to the tube decreases. (Time discrimination in hearing is acute, so that like haptics this requires fast

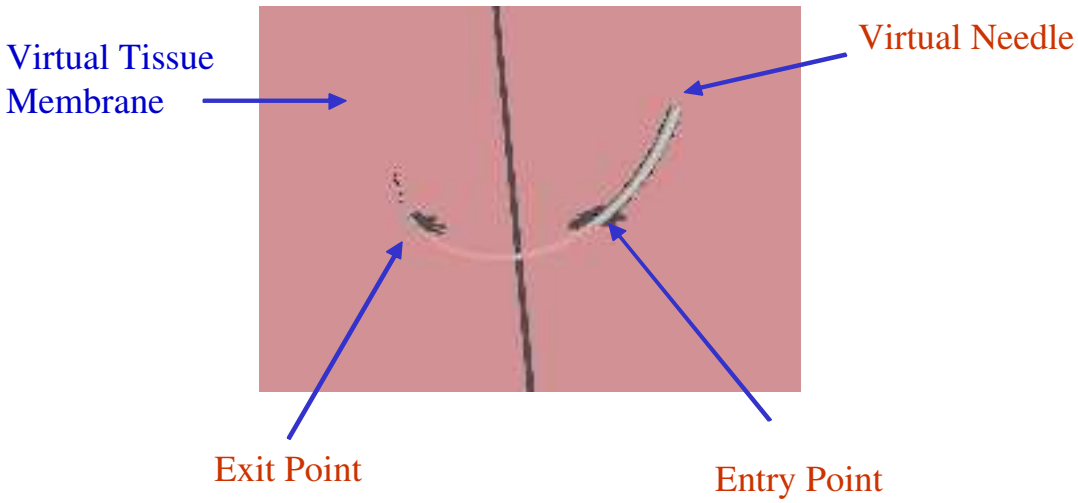

Figure 4.1: Manipulating a geometric 'suture needle' with small movements of the hand, visually displayed as large, as in a microscope.

Figure 4.2: Sliding an arc along a circular path is one of the dexterity primitives we use to learn maneuvering a needle in microsurgery

computation.) A needle that touches the virtual tube causes an error signal. Other cues that will be investigated include:

- visual cues: display of the path, or limits to motion, display of deviation;
- audio cues: contact with virtual objects; error;
- haptic cues: motion guidance, path guidance, destabilization from the intended path.

The implementation of all these cues requires computation of the distance between the needle, modelled as an arc circle, and the vessel, modelled as a portion

of a torus. This computation must be very fast, in order to save CPU cycles for the implementation of the cues.

4.1 Review of related works

The problem of intersection computation between surfaces and curves is fundamental in computer graphics and geometric modelling. For low degree algebraic surfaces such as quadrics, a number of applications and specialized algorithms have been developed for intersection computations by analytic methods [43–46]. An arc-shaped needle touching a plane is simple. More complex case is to touch or approach a curved surface with needle.

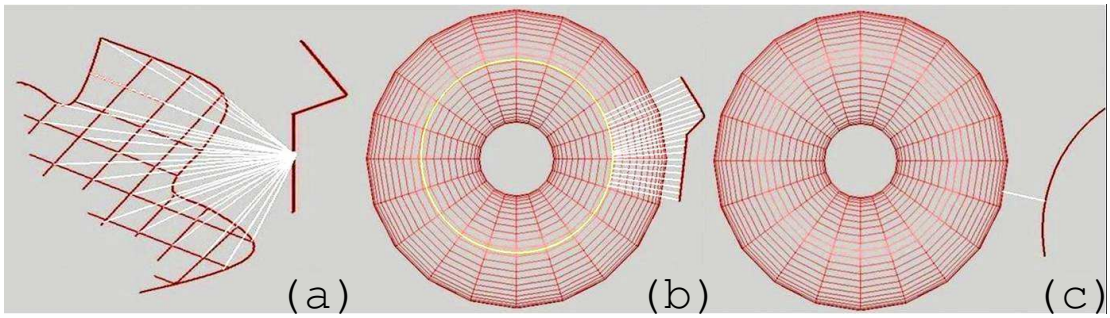

Figure 4.3: Three types of method to detect collision between a surface and a needle : (a) distance computation to many surface points for each needle point; (b) For an implicitly defined surface, the iteration can be reduced to the one-dimensional needle; (c) For algebraically simple surface and needle, the distance can be computed analytically.

If both needle and surface have complex shapes, the objects can be decomposed into triangles and the distance can be computed from all the distances between the triangles approximating these two objects (Fig. 4.3a). If the surface is describable by an equation $f(x, y, z) = c$ for a rapidly computable f such as a polynomial, a trivial algorithm evaluates $f(\cdot) - c$ for regularly spaced points on the needle (Fig. 4.3b) and tests for proximity to 0 (or for a change of sign, which implies crossing 0). Finally, if both surface and needle have tractable shapes, the relative distance can be computed analytically (Fig. 4.3c). This approach is the least general, depending on the specific shapes involved, but (analogously to assembly

programming for critical tasks) allows the highest speeds. This section develops such a method [7], and compares the three approaches.

4.2 Fast algorithm for torus-arc distance

4.2.1 Principle of the algorithm

To train in maneuvering an arc needle along the circle C it lies on, we surround the radius-R circle in the (x, y)-plane by a piece of regular torus. (This will be rotated into various world coordinates, varying the muscular patterns involved in the required hand motion.) There are various ways to describe this by an implicit equation. A torus uniformly r from the circle is given by the fourth degree equation

$$g(x, y, z) = \left(x^2 + y^2 + R^2 + z^2 - r^2\right)^2 - 4\left(x^2 + y^2\right) R^2 = 0 \qquad (4.1)$$

which leads to unhandy algebra: quartic equations do have closed form solutions, but they are rarely used and involve roots of various orders. To make the distance analytically tractable, we use a stereographic projection and lift this torus to S^3, where it becomes a cone described by a single second order equation. Solving a quadratic is a high school problem, and requires only square roots.

Stereographic projection (SP) is a correspondence between spherical and flat geometry, known to geographers since ancient Greek times and widely used in mathematics: a point on a plane tangent to the sphere at its South Pole corresponds to a point on the sphere if the line passes through the North Pole. It has the useful property that circles in the sphere project to circles in the flat domain, and *vice versa* (except that circles through the North Pole correspond to straight lines, 'circles through ∞') [47]. The same property follows for the projection P one dimension higher, between the sphere S^3 of quadruples (l, m, p, q) with $l^2 + m^2 + p^2 + q^2 = 1$ to flat coordinates (x, y, z) in flat three-dimensional space $\mathbb{R}^3$ — where our tube exists — though the centers of the spherical circles do *not*

map to the centers of the circles in $\mathbb{R}^3$. Setting $\rho^2 \equiv x^2 + y^2 + z^2$,

$$P(l, m, p, q) = \left(\frac{l}{1-q}, \frac{m}{1-q}, \frac{p}{1-q} \right) \tag{4.2}$$

$$P^{-1}(x, y, z) = \left(\frac{2x}{\rho^2+1}, \frac{2y}{\rho^2+1}, \frac{2z}{\rho^2+1}, \frac{\rho^2-1}{\rho^2+1} \right). \tag{4.3}$$

Using the stereographic projection, the distance between the needle and the torus can be computed with the following steps:

- Lift the torus $g = 0$ of Equ.4.1 to S^3 where it is the intersection with a quadratic cone $G = 0$, around the unit (l, m)-plane circle (§4.2.2).
- The circle C lifts to a circle $\widehat{C}$ in S^3. Work in the plane π containing $\widehat{C}$ (§4.2.3).
- The torus equation restricts to an inhomogeneous quadratic on π, which vanishes on $\widehat{C}$ where $\widehat{C}$ meets T (§4.2.4).

4.2.2 Lift the torus to S^3

Define a toroidal tube in S^3 by the quadratic equation

$$\alpha^2 \left(l^2 + m^2 \right) - \left(p^2 + q^2 \right) = 0 \tag{4.4}$$

Setting $\alpha = 0$ giving the degenerate case of the unit circle in the (l, m)-plane, which projects to the unit circle U in the (x, y)-plane: $\alpha > 0$ thickens it. (See Appendix for details.) Tori and their circular cross-sections go by P to tori and to circles around (but not centred) on U. Tori are best parametrized in S^3 coordinates (ϕ, ψ, α), setting

$$(l, m, p, q) = (\cos\phi, \sin\phi, \alpha\cos\psi, \alpha\sin\psi). \tag{4.5}$$

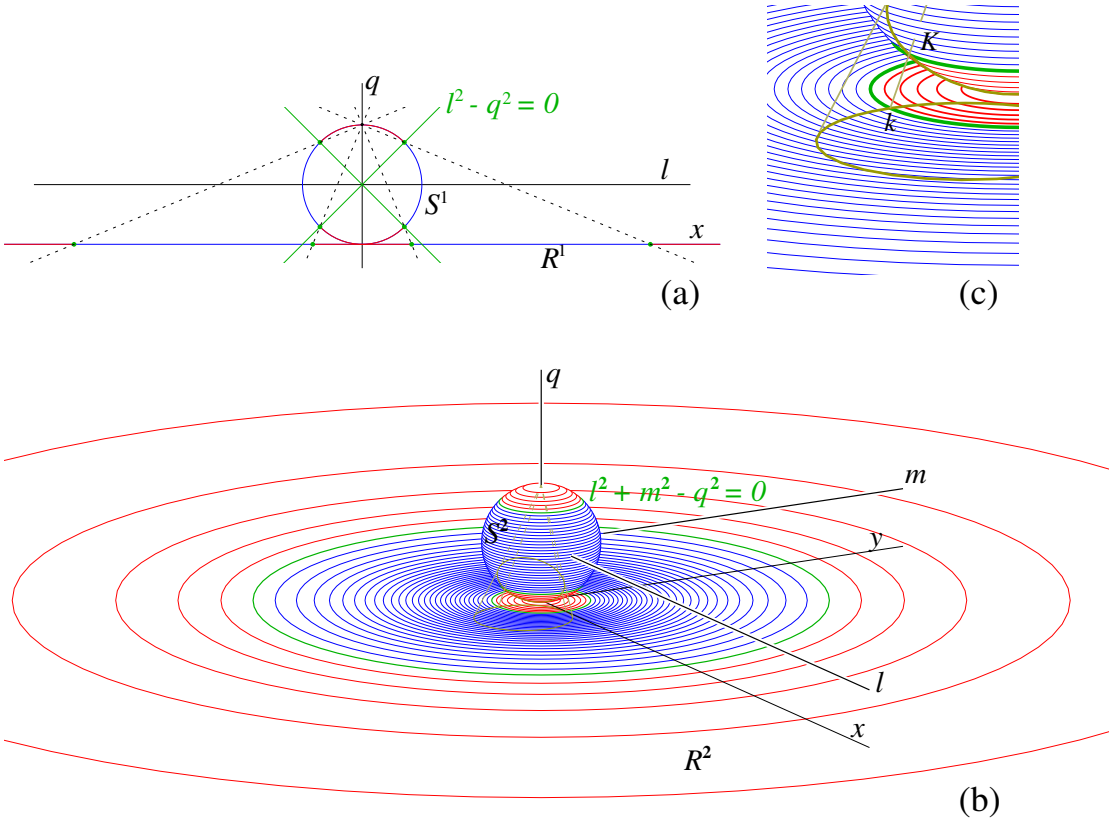

Figure 4.4: To detect the intersection of a torus with the needle (circle arc), we lift to S^3 and consider the intersection between the (quadratic) cone corresponding to the torus and the plane defined by the circle on S^3 (§4.2.1). As it is difficult to represent this in 4 dimensions, this figure illustrates this algorithm in one dimension (a) and two dimensions (b,c), with the stereographic projection is onto S^2. The detail in (c) shows the correspondence between the intersection k of an arc (needle) with the projected quadric (torus), and the intersection K of its lift with the quadric set in S^3.

For drawing, a torus or piece of torus is easily triangulated in these coordinates, and the mesh projected by P to $\mathbb{R}^3$.

To visualize this, consider the analogues in lower dimensions. Fig. 4.4*a* shows stereographic projection from the circle S^1 to the line $\mathbb{R}^1$. The restriction $l^2 - q^2$ of the quadratic (4.4) with $\alpha = 1$ divides the circle into positive regions (blue) and negative (red), vanishing on four boundary points, corresponding to points (not where $l^2 - q^2 = 0$) also marked green in $\mathbb{R}^1$. Moving to projection from the sphere S^2 to the line $\mathbb{R}^2$ (Fig. 4.4 *b*) spins the first figure, with pairs of points where $l^2 - q^2 = 0$ becoming circles where $l^2 + m^2 - q^2 = 0$. Going to (4.4) spins the picture in the (p, q) plane, so that the pair of circles becomes one torus: each pair

of points with the same (l, m) and opposite q becomes a (p, q) circle with fixed (l, m).

A typical circle (brown in Fig. 4.4*b*) in the plane $\mathbb{R}^2$ crosses one of the green circles if and only if the corresponding circle meets $l^2 + m^2 - q^2 = 0$, which is a quadratic calculation also usable for an arc, modelling a needle in the plane. Similar principles apply in $\mathbb{R}^3$.

4.2.3 Lift the needle to S^3

By convention, the following developments use bold vector $\mathbf{v}$ and bold capital matrices $\mathbf{M}$. To detect intersections in $\mathbb{R}^3$, we lift a circle to the corresponding circle in S^3, and look for intersections with (4.4). Explicitly, a circle in S^3 is the intersection of some plane π with S^3, which we first find. Let the needle arc in $\mathbb{R}^3$ be given in needle-model coordinates by $(r\cos s, r\sin s, 0)$ from $s = 0$ to $s = S$. Name the start point $\mathbf{g}_1 = (r, 0, 0)$, the end point $\mathbf{g}_2 = (r\cos S, r\sin S)$ and the mid-point $\mathbf{g}_3 = \left(r\cos\frac{S}{2}, r\sin\frac{S}{2}\right)$. Using homogeneous coordinates, the needle arc is mapped into $\mathbb{R}^3$ by a 4×4 position matrix

$$\mathbf{L} = \begin{bmatrix} l_{11} & l_{12} & l_{13} & l_{14} \\ l_{21} & l_{22} & l_{23} & l_{24} \\ l_{31} & l_{32} & l_{33} & l_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

which takes the three points to

$$\mathbf{f}_1 = \mathbf{L}\mathbf{g}_1 = \begin{bmatrix} l_{14} + rl_{11} & l_{24} + rl_{21} & l_{34} + rl_{31} & 1 \end{bmatrix}^T$$

and similarly for $\mathbf{f}_2 = \mathbf{L}\mathbf{g}_2$ and $\mathbf{f}_3 = \mathbf{L}\mathbf{g}_3$. Write their coordinates as (l_1, m_1, p_1, q_1), (l_2, m_2, p_2, q_2) and (l_3, m_3, p_3, q_3) respectively by applying P to $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$. These

define the hyperplane

$$\begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix} \begin{bmatrix} l \\ m \\ p \\ q \end{bmatrix} = 0\,, \qquad \text{where}$$

$$h_1 = \begin{vmatrix} m_1 & m_2 & m_3 \\ p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \end{vmatrix}, \quad h_2 = -\begin{vmatrix} l_1 & l_2 & l_3 \\ p_1 & p_2 & p_3 \\ q_1 & q_2 & q_3 \end{vmatrix},$$

$$h_3 = \begin{vmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ q_1 & q_2 & q_3 \end{vmatrix}, \quad h_4 = -\begin{vmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ p_1 & p_2 & p_3 \end{vmatrix}.$$

i.e., (h_1, h_2, h_3, h_4) is the triple cross-product of $(l_1, m_1, p_1, q_1) \times (l_2, m_2, p_2, q_2) \times (l_3, m_3, p_3, q_3)$. A vector through the lifted circle's centre is given by the triple cross-product

$$\mathbf{k} = (h_1, h_2, h_3, h_4) \times (\mathbf{f_2} - \mathbf{f_1}) \times (\mathbf{f_3} - \mathbf{f_1}). \tag{4.7}$$

We adjust the length of $\mathbf{k}$ to get the centre in $\mathbb{R}^4$ of the lifted circle $\widehat{C}$

$$\mathbf{c} = (c_1, c_2, c_3, c_4) = \frac{\mathbf{k} \cdot (l_1, m_1, p_1, q_1)}{\mathbf{k} \cdot \mathbf{k}} \mathbf{k} \tag{4.8}$$

The squared radius of $\widehat{C}$ is

$$R^2 = ((l_1, m_1, p_1, q_1) - \mathbf{c}) \cdot ((l_1, m_1, p_1, q_1) - \mathbf{c})\,. \tag{4.9}$$

In the plane of $\widehat{C}$ we choose a first unit basis vector

$$\mathbf{b}_x = \frac{1}{R}\left((l_1, m_1, p_1, q_1) - \mathbf{c}\right), \tag{4.10}$$

with coordinates $(b_{xl}, b_{xm}, b_{xp}, b_{xq})$. Orthogonal to this and in the plane π of $\widehat{C}$ is the triple cross-product

$$\mathbf{b}_y^\star = -\mathbf{b}_x \times \mathbf{c} \times (h_1, h_2, h_3, h_4)\,, \tag{4.11}$$

giving us our second basis vector

$$(b_{yl}, b_{ym}, b_{yp}, b_{yq}) = \mathbf{b}_y \equiv \frac{1}{\sqrt{\mathbf{b}_y^\star \cdot \mathbf{b}_y^\star}} \mathbf{b}_y^\star\,. \tag{4.12}$$

A general point (x, y) in π is then

$$\mathbf{c} + x\,\mathbf{b}_x + y\,\mathbf{b}_y = \begin{bmatrix} c_1 + x\,b_{lx} + y\,b_{ly} \\ c_2 + x\,b_{mx} + y\,b_{my} \\ c_3 + x\,b_{px} + y\,b_{py} \\ c_4 + x\,b_{qx} + y\,b_{qy} \end{bmatrix} \tag{4.13}$$

in $\mathbb{R}^4$ coordinates. If we parametrize the lifted circle by $(R\cos\theta, R\sin\theta)$ the lifted arc still starts (by construction) at $\theta = 0$, but the end angle θ_ω needs not be equal to S. To find it we need to use

$$\theta_\omega = \operatorname{arctan2}\left(\frac{(\mathbf{f}_2 - \mathbf{c}) \cdot \mathbf{b}_y}{R}, \frac{(\mathbf{f}_2 - \mathbf{c}) \cdot \mathbf{b}_x}{R}\right) \tag{4.14}$$

In the plane of $\widehat{C}$, the torus (4.4) becomes

$$0 = ax^2 + bxy + cy^2 + dx + ey + f \tag{4.15}$$

with

$$\begin{aligned}
a &= \alpha^2 \left(b_{lx}^2 + b_{mx}^2\right) - b_{qx}^2 - b_{px}^2 \\
b &= \alpha^2 \left(2b_{lx}b_{ly} + 2b_{mx}b_{my}\right) - 2b_{qx}b_{qy} - 2b_{px}b_{py} \\
c &= \alpha^2 \left(b_{ly}^2 + b_{my}^2\right) - b_{qy}^2 - b_{py}^2 \\
d &= \alpha^2 \left(2c_1b_{lx} + 2c_2b_{mx}\right) - 2c_4b_{qx} - 2c_3b_{px} \\
e &= \alpha^2 \left(2c_1b_{ly} + 2c_2b_{my}\right) - 2c_4b_{qy} - 2c_3b_{py} \\
f &= \alpha^2 \left(c_1^2 + c_2^2\right) - c_4^2 - c_3^2
\end{aligned}$$

4.2.4 Computing the distance

In the lifted circle plane coordinates above, common tangents for contours of

$$\begin{aligned}
T(x,y) &= ax^2 + bxy + cy^2 + dx + ey + f \\
C(x,y) &= x^2 + y^2
\end{aligned} \tag{4.16}$$

occur at the vanishing points

$$\begin{aligned}
\frac{1}{2}\begin{bmatrix} \frac{\partial}{\partial x}T(x,y) \\ \frac{\partial}{\partial y}T(x,y) \end{bmatrix} \times \begin{bmatrix} \frac{\partial}{\partial x}C(x,y) \\ \frac{\partial}{\partial y}C(x,y) \end{bmatrix} &= 0 \\
\frac{1}{2}\begin{bmatrix} 2ax + by + d \\ bx + 2cy + e \end{bmatrix} \times \begin{bmatrix} 2x \\ 2y \end{bmatrix} &= 0 \\
b\left(x^2 - y^2\right) + 2\left(c - a\right)xy + ex - dy &= 0
\end{aligned} \tag{4.17}$$

Look first at the homogeneous zeroes of

$$b\left(x^2 - y^2\right) + \left(c - a\right)2xy = 0 \tag{4.18}$$

Substituting new (u, v) coordinates into this equation and abbreviating one coefficient to $c - a = g$ yields

$$\begin{aligned} 0 &= b\left(x^2 - y^2\right) + g\left(2xy\right), \\ x &= u\cos\psi - v\sin\psi, \\ y &= u\sin\psi + v\cos\psi \end{aligned} \tag{4.19}$$

yields

$$\left(u^2 - v^2\right)\left(b\cos 2\psi + g\sin 2\psi\right) + 2vu\left(g\cos 2\psi - b\sin 2\psi\right) = 0 \tag{4.20}$$

If we set

$$\begin{aligned} 0 &\equiv b\cos 2\psi + g\sin 2\psi\,, \\ \psi &\equiv \frac{-1}{2}\arctan\frac{b}{d}\,, \end{aligned} \tag{4.21}$$

(4.20) becomes

$$2\left(d\cos 2\psi - b\sin 2\psi\right)uv = 0 \tag{4.22}$$

and the inhomogeneous quadratic for the torus

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

becomes

$$\beta uv + \gamma u + \delta v = 0 \tag{4.23}$$

where

$$\begin{aligned} \beta &\equiv 2\left(g\cos 2\psi - b\sin 2\psi\right) \\ \gamma &\equiv d\cos\psi + e\sin\psi \\ \delta &\equiv e\cos\psi - d\sin\psi \end{aligned}$$

This gives (unless $\beta = \delta = 0$, which reduces (4.23) to $u = 0$ and we're done)

$$v\left(\beta u + \delta\right) + \gamma u = 0 \tag{4.24}$$

$$\frac{-\gamma u}{\beta u + \delta} = v \tag{4.25}$$

while $x^2 + y^2 = R^2$ rotates to $u^2 + v^2 = R^2$, so

$$\begin{aligned} R^2 &= u^2 + \left(\frac{-\gamma u}{\beta u + \delta}\right)^2 \\ 0 &= u^2\left(\beta^2 u^2 + 2\beta u\delta + \delta^2 + \gamma^2\right) - R^2\left(\beta u + \delta\right)^2 \\ 0 &= \beta^2 u^4 + 2\beta\delta u^3 + \left(\gamma^2 + \delta^2 - \beta^2 R^2\right)u^2 - 2\beta\delta R^2 u - \delta^2 R^2 \end{aligned} \tag{4.26}$$

Get the real roots between $\pm R$ of (4.26) numerically or with a standard routine [48], plug in to (4.25), put u and v into (4.19) and we have the points at which (4.16) has its extrema[1] on the radius-R circle. We find the θ values of these points in the standard parametrisation $(R\cos\theta, R\sin\theta)$ of the circle.

From this we can determine whether an arc from θ_1 to θ_2 of the radius-R circle meets the set

$$ax^2 + bxy + cy^2 + dx + ey + f = 0\,. \tag{4.27}$$

Writing

$$\phi\left(\theta\right) = aR^2\left(c\theta\right)^2 + bR^2\left(c\theta\right)\left(s\theta\right) + R^2 c\left(s\theta\right)^2 + Rd\left(c\theta\right) + Re\left(s\theta\right) + f \tag{4.28}$$

with $c\theta \equiv \cos\theta$, $s\theta \equiv \sin\theta$, we find $\phi\left(0\right)$ and $\phi\left(\theta_\omega\right)$. If there is no maximum or minimum between θ_α and θ_ω, we have an intersection with (4.27) if and only if 0 is between $\phi\left(0\right)$ and $\phi\left(\theta_\omega\right)$. If the arc contains n extrema $\theta_1, \ldots, \theta_n$ (for $1 < n \leq 4$), there is an intersection if 0 is between $\phi\left(0\right)$ and $\phi\left(\theta_1\right)$, or between $\phi\left(\theta_1\right)$ and $\phi\left(\theta_2\right)$ or … or between $\phi\left(\theta_n\right)$ and $\phi\left(\theta_\omega\right)$.

[1] Which are minima and which are maxima appears easily, by inspecting the values. If there are two solutions, the greater and lesser are the unique maximum and minimum. If there are four, both maxima are above both minima. However, we do not need this information.

Also, the θ of the smallest absolute value of $\phi(\theta)$ between $\phi(0)$ and $\phi(\theta_\omega)$ can be used to find the point of the needle arc which is nearest (in the sense of lowest G) to the surface of the curved tube in $\mathbb{R}^3$. Put $(R\cos\theta, R\sin\theta)$ to (4.13), we get the coordinate of the nearest point in π. Then use (4.3) to get the coordinate in $\mathbb{R}^3$. Computing the distance between the nearest point and the core curve of torus decides how far the arc and tube are from collision and whether inside ($G < R$) or outside ($G > R$). The distance can be used for warning signals such as force, sounds or color gradients, useful in training.

The discussion above applies to the whole torus, to simplify the first reading and checking. For a segment of torus, contained between the planes $\{(x, y, z), x = 0\}$ and $\{(x, y, z), (\cos\Omega)\, x + (\sin\Omega)\, y = 0\}$, it is straightforward to find the part of the needle arc that also lies between the two planes, and test that rather than the whole arc for collision with the torus.

4.3 Performance and Comparison

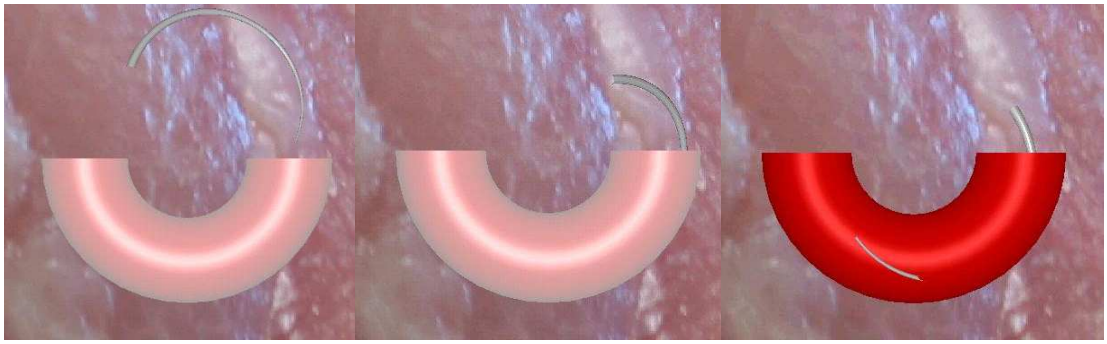

Figure 4.5: The analytical algorithm enables fast computation of the separation of needle and torus.

We implemented the SP algorithm of section 4.2 (on a Dell PC with 2.4GHz Pentium CPU, 256 MB) to detect collision between a circular needle and a toroidal blood vessel (Fig. 4.5). We measured the computation time as a function of the accuracy, defined as the end criterion of the Newton algorithm used to solve the fourth order equation (4.26). We see in Fig. 4.6 that the computation time is as

low as $0.015ms$ and does not increase with accuracy, an important property to simulate the microworld [7].

The SP algorithm was first compared with a general detection algorithm, the RAPID interference detection system [49]. For a fair comparison we excluded the set-up time of this algorithm, as well as the visualization effort which common to all the algorithms. The precision of RAPID depends on the diameter of the largest triangle. This is a slightly different measure of precision than for SP, however the computation time is order of magnitudes larger than with SP, so this difference is inessential. Similarly, the algorithm of discretizing the needle and computing the distance to the torus for all these discrete points (Fig. 4.3b) requires computation time growing proportionally with the precision. In that case the precision corresponds to the discretization step.

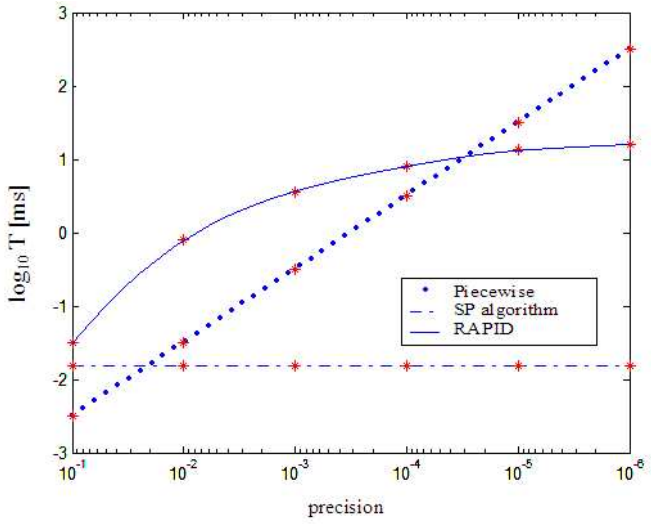

Figure 4.6: Computation time for torus-needle separation for the analytic method versus two numerical ones, comparing to the other two approaches. The shaded area corresponds to the precision required in the actual task to learn microsurgery.

With this algorithm, a good collision detection and distance computation requires to discretize the needle with at least a hundred points. At such precision, the SP algorithm is orders of magnitude faster than numerical approaches at accuracies below 10^{-2}, and can be used to implement all audio, visual and haptic cues of section 4.

4.4 Discussion

To microsurgeon, needle maneuvering is a task requiring very fine control. In suture, a curved needle suturing together two membrane edges, must pass through at carefully planned places. Both position and orientation must change simultaneously, in six-degree-of-freedom motion.

For the needle maneuvering skill, we introduced a method using stereographic projection to compute the distance between the curved needle and a curved vessel. This analytical algorithm was shown to be faster by orders of magnitude than numerical ones. Furthermore the computation time barely increases with increasing precision, a critical condition to simulate the microworld. We used geometric properties specific to the shapes involved, such as the circularity of the arc allowing it to lift to a planar circle in S^3, as an ellipse for instance does not. This limits the generality of the specific algebra used, but still shows that geometrically special coding can be as powerful as is rewriting an algorithm in system-specific assembler.

Chapter 5

Real-time knot-tying simulation

Knot tying is a bi-manual task requiring skillful control of the nylon and knowledge of the correct topology, which will be trained using two dexterity primitives. Probably the most obvious component of knot tying is to realize correct knot topology. Whether a knot is going to be correct or not depends on its topology. The suturing process consists of creating a loop, maneuvering one end of the thread, and then pulling the other end through this loop. The creation of this loop and then pulling the other end across from the right direction is the criterion which decides the correctness of the knot. A less obvious critical component of knot tying consists of skillful manipulation of the nylon. Both in our experience as trainee and according to experimented surgeons experience) this component is more difficult to acquire. In this chapter, we focus on the modelling of a real-time thread simulator, which is developed to train the above subskill.

5.1 Review of related works

While a kinematic scheme such as "follow the leader" [10] may be used to visualize knot tying in real time, however it cannot give an idea of the dynamics involved in manipulation of real nylon threads. Bending and twisting are essential components of nylon thread manipulation, which cannot be realized using "follow the

leader" scheme. Further, while the typical force level in microsurgery is below perception, experiencing the tread dynamics using feedback of scaled force may help learning. "Follow the leader" scheme is based on geometry and is not suitable for our application, as we wish to analyze the effect of haptic cues in the learning process. All of this requires computation of the nylon thread dynamics. In [50], a spline of linear springs with control points at their extremities was used for knot tying simulation. The focus was on dealing with self-collisions when the knot is being pulled and tied, while conserving energy, mass and momentum of the system. The authors realize the limitation of their model as they do not model bending or twisting. Moreover, it's not a real-time simulation. According to surgeons, in microsurgery twisting the thread plays an extremely important role. In this chapter we present a scheme for modelling nylon thread dynamics, which includes modelling of stretching, bending and twisting of thread in real time. Forces are modelled and we have done experiments with Phantom device, which show that we have obtained sufficient visual, haptic and dynamic realism.

5.2 Thread dynamics

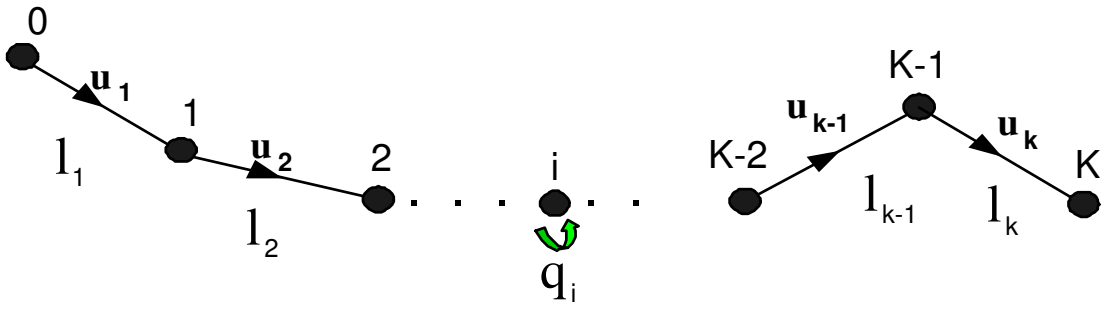

Figure 5.1: Structure of the thread

The topological structure of the thread is represented by a list of $k+1$ nodes N_i, $i \in \{0, 1, \cdots, k\}$ at positions $\mathbf{x}_i$ with torsion angle q_i connected by k links L_i, $i \in \{1, 2, \cdots, k\}$. Let $\mathbf{u}_i$, $i = 1, \cdots, k$ represent the vectors pointing from node N_{i-1} to N_i.

The dynamics structure is realized by attaching a mass m_i and an inertia momentum I_i to each node N_i, and a group of (massless) springs to each link L_i. This means that the links produce stretching, compressing, bending and twisting between and on the nodes. The dynamics of each node N_i is determined by:

$$\begin{aligned} \mathbf{F}_i &= m_i \cdot \ddot{\mathbf{x}}_i \\ \tau_i &= I_i \cdot \ddot{q}_i \end{aligned} \tag{5.1}$$

where F_i is the sum of external forces applied on mass m_i and τ_i the torsion force applied along the thread.

The velocity $(\dot{\mathbf{x}}_i, \dot{q}_i)$ and position $(\mathbf{x}_i, q_i)$ of each node are numerically (Euler) integrated with

$$\begin{aligned} \dot{\mathbf{x}}_i(t+dt) &= \dot{\mathbf{x}}_i(t) + \ddot{\mathbf{x}}_i \cdot dt \\ \dot{q}_i(t+dt) &= \dot{q}_i(t) + \ddot{q}_i \cdot dt \end{aligned}$$

$$\begin{aligned} \mathbf{x}_i(t+dt) &= \mathbf{x}_i(t) + \dot{\mathbf{x}}_i(t+dt) \cdot dt \\ q_i(t+dt) &= q_i(t) + \dot{q}_i(t+dt) \cdot dt \end{aligned} \tag{5.2}$$

In Equ.(5.1), $\mathbf{F}_i$ is the sum of external forces applied on mass m_i:

$$\mathbf{F}_i = \mathbf{F}_i^s + \mathbf{F}_i^b + \mathbf{F}_i^t + \mathbf{F}_i^r + \mathbf{F}_i^c + m_i\,\mathbf{g} \tag{5.3}$$

where $\mathbf{F}_i^s$ is the force resulting from stretching and compression of the links connected to node N_i, $\mathbf{F}_i^b$ is the force resulting from bending the link, $\mathbf{F}_i^t$ is the force resulting from twisting of the link, $\mathbf{F}_i^r$ corresponds to friction, $\mathbf{F}_i^c$ is the reaction force when two links are in contact, and $m_i\,\mathbf{g}$ the gravity force.

Stretching and compression force F^s

All nodes (except the first and last) experience stretching and compression of the springs bounded to them according to:

$$\mathbf{F}_i^s = -\kappa_i^s \cdot \Delta_i \cdot \mathbf{u}_i - \kappa_{i-1}^s \cdot \Delta_{i-1} \cdot \mathbf{u}_{i-1} \tag{5.4}$$

where $\Delta_i = l_i - r_i$ is the difference between the current length of the link and its resting length, and κ_i^s is the stiffness constant of spring link.

In microsurgery the minute force when pulling the suture can be ignored, however a relatively strong force can be felt when a knot is tightened or the suture is stretched. Therefore the nodes mass is set to be low and the stiffness constant is set to be high.

Bending force F^b

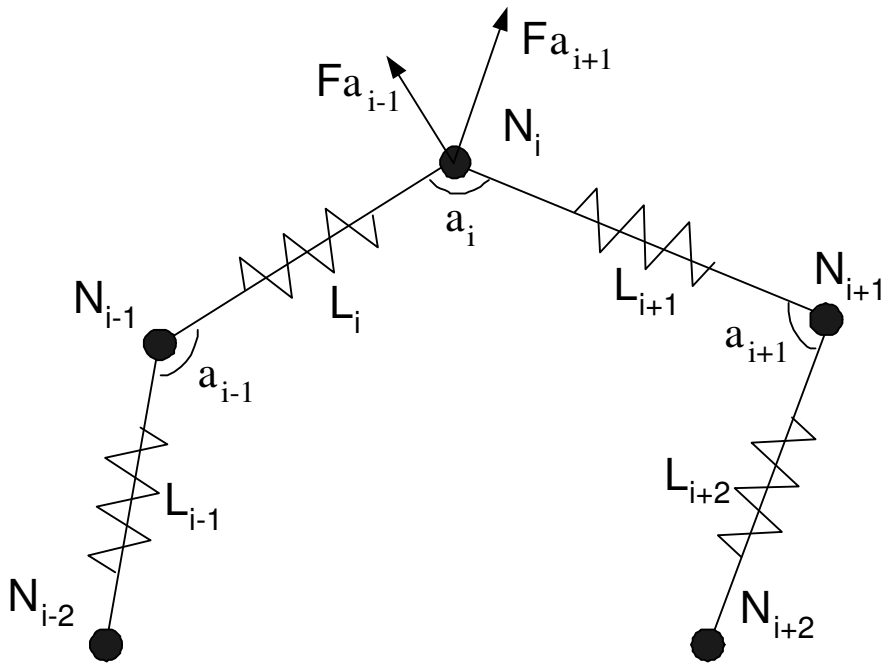

Figure 5.2: Bending force at node N_i.

Fig.5.2 illustrates how the bending force $\mathbf{F}_{\alpha_{i-1}}$ and $\mathbf{F}_{\alpha_{i+1}}$ applied on mass m_i is computed using the bending angles α_{i-1} and α_{i+1}. The bending force is computed

as

$$\begin{aligned}
\mathbf{F}_{\alpha_{i-1}} &= -\kappa^b_{i-1} \cdot (\pi - \alpha_{i-1}) \cdot \mathbf{u}_{i-1} \times (\mathbf{u}_{i-2} \times \mathbf{u}_{i-1}) \\
\mathbf{F}_{\alpha_{i+1}} &= -\kappa^b_{i+1} \cdot (\pi - \alpha_{i+1}) \cdot \mathbf{u}_{i} \times (\mathbf{u}_{i+1} \times \mathbf{u}_{i}) \\
\mathbf{F}^b_i &= \mathbf{F}_{\alpha_{i-1}} + \mathbf{F}_{\alpha_{i+1}}
\end{aligned} \tag{5.5}$$

where κ^b_{i-1} is the stiffness constant of the angular spring, the bending angle is measured by $(\pi - \alpha_{i-1})$, and the orientation of the bending force is specified by $\mathbf{u}_{i-1} \times (\mathbf{u}_{i-2} \times \mathbf{u}_{i-1})$. Fig.5.3 shows the difference of a thread with(out) high bending property.

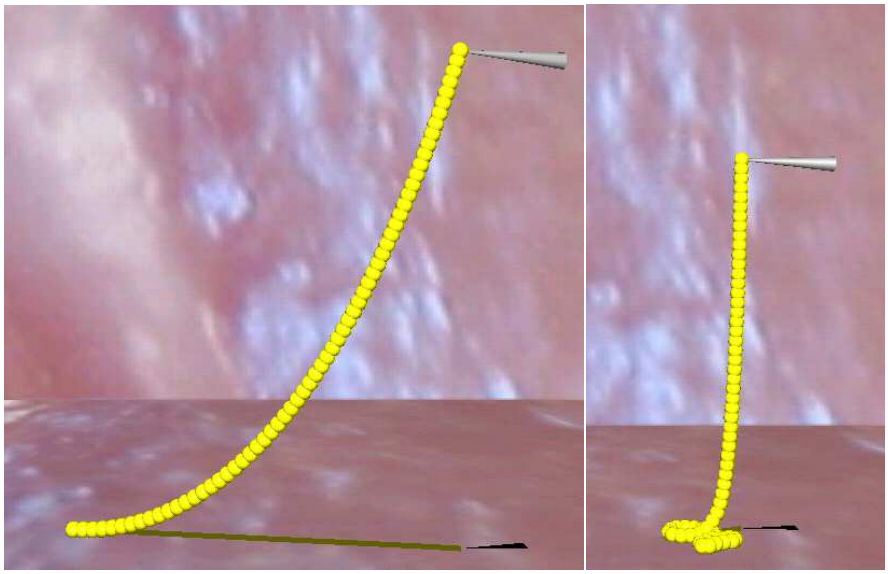

Figure 5.3: The thread with high bending (left) and without bending (right)

Twisting force F^t

An angular spring is associated to each spring link, which produce the torsion torque (5.1)

$$\tau_{i-1} = \kappa^d_{i-1} \cdot (q_i - q_{i-1}) \tag{5.6}$$

The torsion angle q_i is propagated from the grasping node to its connected nodes iteratively using (5.2) and (5.6). Torsion of link L_{i-1} produces a twisting force $\mathbf{F}^t_i$

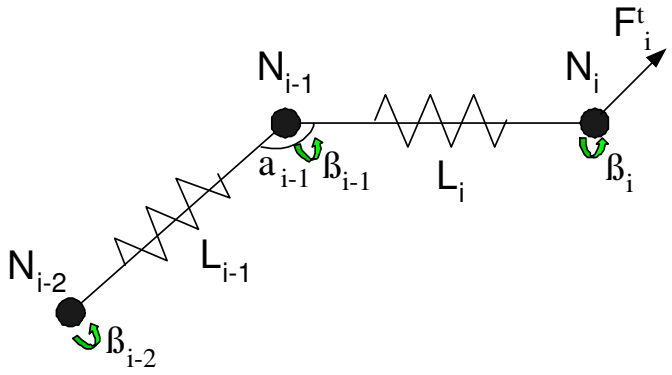

Figure 5.4: Twisting effect

on the mass m_i given by

$$\mathbf{F}_i^t = -\kappa_{i-1}^t \cdot (\beta_{i-2} - \beta_{i-1}) \cdot \sin(\pi - \alpha_{i-1}) \cdot (\mathbf{u}_{i-2} \times \mathbf{u}_{i-1}) \tag{5.7}$$

where κ_{i-1}^t is the stiffness constant of the angular spring, $(\beta_{i-2} - \beta_{i-1})$ is the torsion angle of spring link L_{i-1} and $(\mathbf{u}_{i-2} \times \mathbf{u}_{i-1})$ gives out the orientation vector of the twisting force. Fig.5.5 shows the twisting thread model.

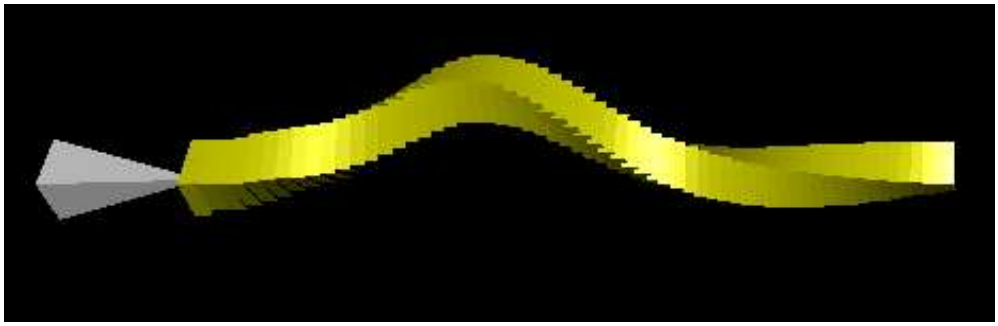

Figure 5.5: The twisting thread model

Damping (F^r,τ^r)

Damping is brought to the system using friction force

$$\begin{aligned} \mathbf{F}_i^r &= (\kappa^{fx} + \kappa^c) \cdot \dot{\mathbf{x}}_i(t) \\ \tau_i^r &= \kappa^{fq} \cdot \dot{q}_i(t) \end{aligned} \tag{5.8}$$

Damping corresponding to internal dynamics is realized trough κ^{fx} and κ^{fq} ('f' stands for free), and κ^c is used to realize contact with the environment. Realistic viscous effect is achieved using high κ^{fx} and κ^{fq}.

Contact force F_i^c

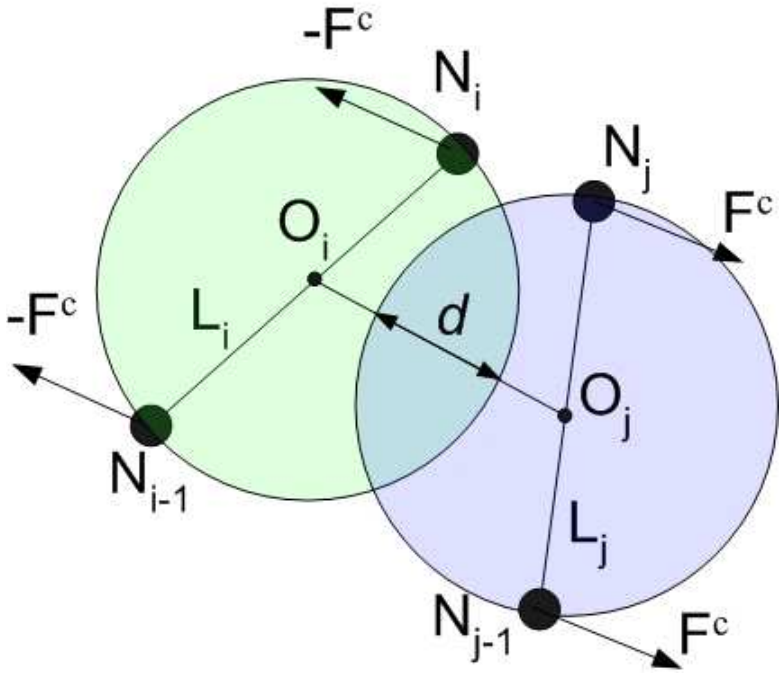

Figure 5.6: Reaction force in self-collision

The contact of the thread with itself or other objects such as surgery tools or skin is handled through the force $\mathbf{F}_i^c$. Here we describe in particular the force for self-collision. Each link is bounded with a sphere to represent the volume (Fig.5.6). The reaction force between two colliding links, acting on the four neighboring nodes, is computed as

$$\mathbf{F}^c = -\kappa^c e^{-\kappa^o d} \cdot \mathbf{u}_i \times (\mathbf{u}_i \times \mathbf{u}_j) \tag{5.9}$$

The parameter κ^o must be set carefully. A too larger reaction force will let the links bounce off too far away such that the knots can not be tightened. Too small reaction force will not prevent the links from passing through each other.

The complete treatment of collision is described in section 5.3. For collision with objects d is determined from the penetration of the sphere associated with the link into the object.

Gravity

In microsurgery, the suture is very light and usually in viscous environment, and the motion of the suture is only slightly affected by gravity. However, the gravity can still be observed in suture's dynamics i.e. without it the thread simulation is not realistic. In the simulation $\mathbf{m}$ of Equ.(5.3) needs to be adjusted to a small value.

5.3 Collision detection

In microsurgery simulation, a suture often collides with itself and with other objects (surgery tools, organs). All collisions must be detected completely at every loop and then the reaction force (5.9) can be correctly set to prevent the thread particles from passing through obstacles or through each other.

Collision detection is a particular important problem in 3D VR simulation. Extensive research has been devoted to checking collisions between rigid objects. However, less effort has been spent on collision detection between deformable objects, especially self-collisions in a deformable object.

Possible methods to detect self-collision in a thread have been compared in [10]. The "grid method" decomposes a deformable object to large groups of small rigid pieces, then defines a uniform grid of cubes over the 3D workspace, computes the intersection between each piece and the cube, and records the results to a hash table. At each simulation loop, the hash table is refreshed, and self-collisions are found by checking every intersected cube which contain two or more pieces. If all its pieces have the same size and the centers of the minimum enclosing spheres of

any two pieces cannot come closer than some absolute constant, the grid method always takes O(n) time at each loop, which is optimal in the worst case. The bounding-volume hierarchy method (BVH) constructs a hierarchical bounding representation based on spheres. The key to efficiency is a simple search routine that uses the bounding representation to ignore most of the possible pairs of components. A BVH is best at representing geometric proximity inside an object at various levels of detail. However the geometric proximity is not invariant in a deforming object, and constant rebuilding of a BVH to maintain geometric proximity can be very costly. On the other hand, the sequence of the links of our thread model is fixed, which lead us to build a BVH for the thread that has an invariant topology, meaning that when the thread deforms we only need to update the locations and volumes of each bounding sphere.

As we have mentioned in 5.2, minimum spheres are used to enclose each link (with two nodes) and represent the volume of the thread pieces. The spheres are then filled to the BVH as the leaves (in lowest level), after which the BVH can be built from bottom to up by bounding pairs of successive spheres to new spheres to form each of higher level. Hence the resulting BVH is a balanced binary tree. Each intermediate sphere tightly bounds its two children and also encloses all the leaf spheres below it. The root sphere encloses the entire thread and all the bounding spheres. To find the collision between root spheres T1 and T2, a recursive algorithm is defined:

Algorithm **FindCollision** (T1, T2)
If (T1 and T2 have null intersection)
    Return;
Else
    If (T1 and T2 are both leaf spheres)
        If (T1 and T2 are neither same one nor neighbor)
            Save T1 and T2 as a collision pair;

Return;

If (T1's level higher than T2's)

Switch T1 and T2;

FindCollision (T1, T2→left_item);

FindCollision (T1, T2→right_item);

Return;

To find self-collision in T, we use function **FindCollision**(T, T) after update the BVH tree. Then all the collision pairs are stored into a list, which will be used to set the reaction force as described in 5.2. The topology structure of BVH is pre-computed once at the beginning of the simulation and then remained fixed. During the simulation, only the position and radius of the bounding spheres are updated in each loop, since the update is done bottom up, totally only $(2n-1)$ update for each loop. It's shown that the BVH method is only slightly less efficient in the worst case than the grid method in detecting self-collisions; it is very efficient in the average case, especially when other objects are involved.

5.4 Simulation algorithm

We have developed a real-time thread simulator. Assume the node C is the control node, which is grasped by surgery tool (here we used keyboard or haptic device to simulate the tool). The simulator supports one or more control nodes. The C is usually set to be two nodes since suture is a bimanual task in microsurgery.

The positions of the control nodes are obtained in high frequency ($500 \sim 1000Hz$) by reading the positions/orientations of the tracking devices. From the current position of each node, we will be able to calculate the total external force from (5.3). This external force will be applied to the each of the nodes in the next loop with the period of time step dt. Because the dt is very small, we assume the system is continuous. During each simulation loop, all the links are updated using

(5.2).

Algorithm Flow:

1. Read the positions of all the control nodes C;
2. For each node, calculate the total external force applied on the node with (5.3);
3. Read the time step dt between the two consecutive loops;
4. Update the new position of each node by using (5.2).

The simulator has considered most of the physical properties and achieved realistic dynamic effect. Most of the parameters are related to each others and also depending on the computational power of the workstation. They must be set very carefully to make sure the system is stable, especially in simulating the knot tying. The safe bound of the parameters are found by trial and error. By modifying the stiffness constants and damping factors, the simulator can simulate the dynamic behaviors of different types of threads, from soft necklace to stiff nylon.

5.5 Application and results

We have implemented our thread simulator to a real-time knot tying simulation (Fig.5.7). Our physics-based thread model has essential advantages over the geometry-based model of [10]. In Joel's simulation, a considerable attention has been given to collision management. When a knot has been done, it is firstly identified and then moves as a single rigid cluster according to the grasping nodes. Our dynamics based simulation is free from such considerations. And we do not need to consider special tricks for special scenarios. The collision management is simplified by setting appropriate reaction forces (5.9), such that it is no longer necessary to consider the geometry when a knot is tightened. The reaction forces, bending and

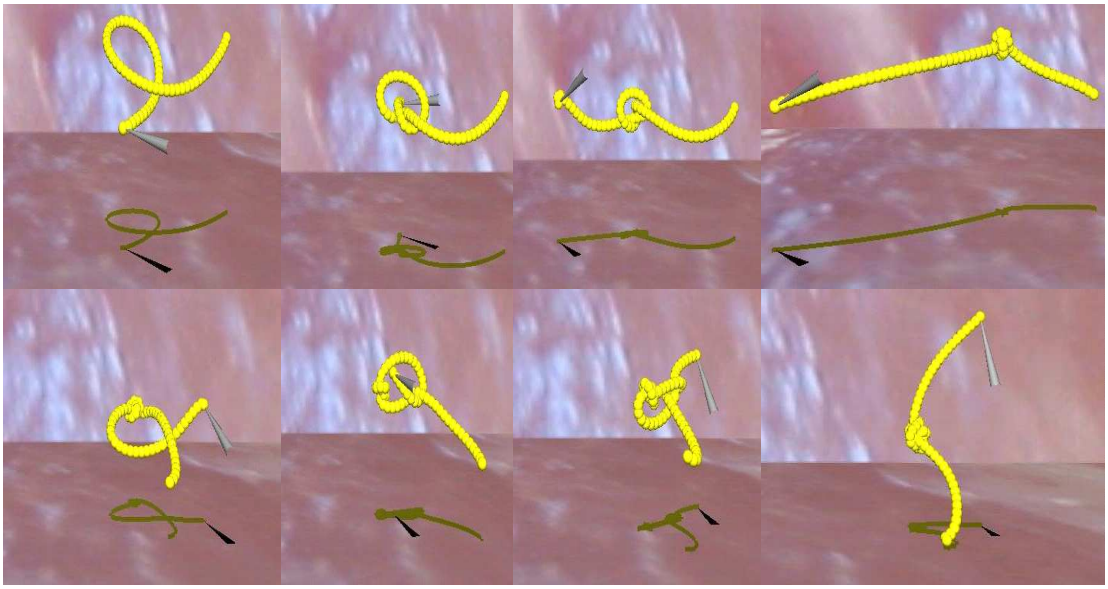

Figure 5.7: Key steps in the tying of a double overhand knot

twisting make them bounce away from their neighbors while being pulled back by the connected springs at the same time. Every piece of the thread follows Newton's physical law and so behaves naturally. Even the details of the motion could be simulated, such as a sudden pull of the thread resulting in the rest of the parts swinging and bouncing very quickly and then resting. Also, gravity and friction were added in a very simple and natural way. In contrast, geometry-based model will only produce stiff motion. As shown in Fig.5.8, we compare the motion of the geometry-based approach to our physics-based approach. In geometry-based approach, we take use of FTL (Follow The Leader) algorithm [10]. Each figure shows 4 different sates, which we swing the thread up and down rhythmically. Apparently, the motion of FTL is bounded inside an envelop no matter how we increase the acceleration, which is not as realistic as the physics-based one.

For our initial experiment, 65 nodes with 64 links are used to model the thread. This number of links is sufficiently long enough to make 6 overhand knots and can be computed at haptic speed. Usually we only need a double-overhand knot in microsurgery simulation. More links can be used for a pure visual realistic simulation. Users can easily tied different kinds of knot.

The Phantom device is used to simulate the surgery tool to control the thread.

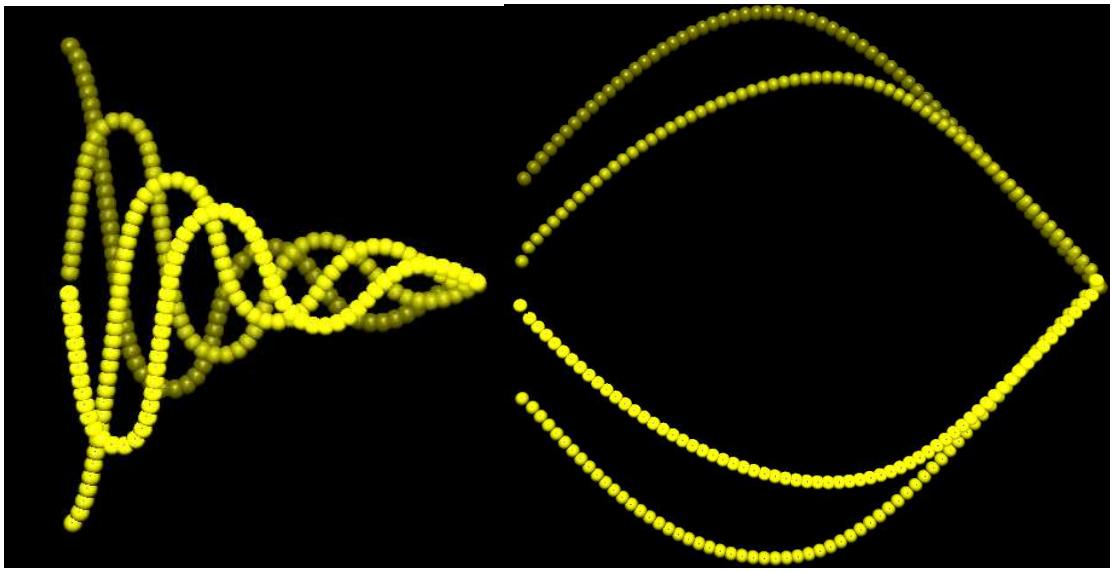

Figure 5.8: Compare the dynamic behavior the geometry-based approach (left) to the physics-based approach (right)

Any nodes of the thread can be grasped. As soon as the control node is grasped, set the motion of the control node follow the phantom, and also transfer the force which is applied on the control node to the phantom at each loop.

The simulator runs on a DELL workstation with dual 2.6G $XEON$ processors. The graphics rendering is run at 30Hz. A stereo glasses is used to provide the true binocular display, with the shadow projected on the surface of the skin, a realistic stereographic effect is realized. Users report that they have no difficult to distinguish the depth. The simulation process with haptic rendering is run at 500 to $1000Hz$, depending on the number of (self)-collisions. We sample the position of the control node, solve dynamics, and update the position of the rest of the nodes. And then we sample next position of control node. If solving dynamics takes too much time, the next position sampled could be at quite a distance from previous position if we doing fast manipulation. This will lead to lag in motion display and force rendering. Because our thread simulator run at very high frequency, there is no lag found while manipulating the phantom device.

5.6 Discussion

This chapter represents a physics-based thread simulator that enable realistic knot tying intersection in haptic rendering rate. The model follows Newton's law and behaves naturally. It has considered most of the properties of the real thread, such as stretching, compressing, bending and twisting by assuming the thread will retrieve to its nature state either quickly or slowly. The structure of the system has essential advantages over geometrically based models. Compared with the geometrically based method of [10], our simulator starts with a different thread configuration, similar collision detection methods and simplified collision management; it easily overcomes some of the limitation of non-physical simulation system while provides more realistic effect both in visual and haptic aspects. Moreover, in micro-world simulation, details cannot be ignored. Even in real world, a thread needs to take some time to rest after you stop pulling it. Simply using damping factor to affect the motion of the thread in non-physical system is not able to reproduce the swinging and bouncing effects which can be observed in the motion of real thread. Knot identification could be added in the same way than was done in [10].

However, not all the dynamic behaviors of the suture in microsurgery can be realistically modelled, such as "snap", which often happens in microsurgery environment. Such a sudden jump is possible to be simulated till we investigate higher order energy-based models [51].

Chapter 6

Conclusion

We are developing a virtual Reality micromanipulation learning system to train microsurgeon, implemented on the dedicated workstation, with collocated visual and haptic workspaces. Conventional approaches judge a virtual environment by its resemblance to the real environment, and often the "sense of immersion" or "presence". In contrast, we are only concerned with efficiency in learning. We train simple dexterity primitives corresponding to selected features of the complex task, and also investigate suitable multisensory learning cues systematically. To this end we use simple environments with only selected features of the complex tasks, and develop fast algorithms to investigate the learning of these dexterity primitives with various multi-sensory cues.

For the modelling of soft tissue mechanics, we described a multi-scale FEM algorithm which uses elements from multiple levels in a hierarchy of mesh similar to the progressive mesh. We show that this algorithm computes fast compared to the traditional FEM algorithm with low loss of accuracy. For the needle maneuvering simulation, we introduced a method using stereographic projection to compute the distance between the curved needle and a curved vessel. This analytical algorithm was shown to be faster by orders of magnitude than numerical ones. Furthermore the computation time barely increases with increasing precision, a critical condition to simulate the microworld. For the knot-tying simulation, we

presented a physics-based thread simulator that enable realistic knot tying intersection in haptic rendering rate. The model follows Newton's law and behaves naturally. It has considered most of the properties of the real thread, such as stretching, compressing, bending and twisting. The effects of gravity and friction are also added in very simple way.

Overall, the contributions of this work are two-fold: for human learning aspects, we investigate the possibilities of using the primitives to train the subskills separately to achieve better training efficiency, which is different to conventional approaches; for VR simulation aspects, we developed some novel algorithms to speed up the real-time simulation, which makes the haptic rendering possible.

For future work, we might fulfill the VR microsurgery trainer to provide other primitive training tasks and evaluate its performance on real subjects.

References

[1] Singapore ministry of health statistics, http://app.moh.gov.sg.

[2] Microsurgery training programme, indiana university, http://www.iupui.edu/,micrsurg.

[3] RS Johansson et al. Eye-hand coordination in object manipulation. *Journal of Neuroscience*, 21:6917–6932, 2001.

[4] D. Ota, B. Loftin, T. Saito, R. Lea, and J. Keller. Virtual reality in surgical education, computers in biology and medicine. *computers in Biology and Medicine*, 25(2):127–137, 1995.

[5] E. Erel, B. Aiyenibe, and P.E.M. Butler. Microsurgery simulators in virtual reality: Review. *Journal of Microsurgery*, 23(2):147–152, 2000.

[6] RV O'Toole, RR Playter, et al. Measuring and developing suturing technique with a virtual reality surgical simulator. *Journal of American College of Surgeons*, 189:114–127, 1999.

[7] F. Wang, T. Poston, C.L. Teo, K.M. Lim, and E. Burdet. Multisensory learning cues using analytical collision detection between a needle and a tube. In *Proceedings of IEEE VR 04, Haptic Symposum*, pages 339–346, 2004.

[8] M. Haruno, D.M. Wolpert, and M. Kawato. Mosaic model for sensorimotor learning and control. *Neural Computation*, 13:2201–2220, 2001.

[9] J. Brown, K. Montgomery, J-C Latombe, and M. Stephanides. A microsurgery simulation system. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2001.

[10] J. Brown, J-C Latombe, and K. Montgomery. Real-time knot tying simulation. In *Dimacs Workshop on Medical Applications in Computational Geometry*, 2003.

[11] E. Burdet, C.L. Teo, and H.P. Lim. A robotic teacher for chinese ideograms. In *Proceedings of 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, page 335, 2002.

[12] E.S. Boy, E. Burdet, C.L. Teo, and J.E. Colgate. Experimental evaluation of the learning cobot. In *Proceedings of Eurohaptics 2003*, 2003.

[13] R.D. Shilling and B.G. Shinn-Cunningham. Virtual auditory displays. In *Handbook of Virtual Environment Technology*, 2001.

[14] S. Smith, RM Pickett, and MG Williams. Environments for exploring auditory representations of multidimensional data. In Kramer G, editor, *Auditory Display: Sonification, Audification and Auditory Interfaces. SFI Studies in the Sciences of Complexity.*, volume XVIII. Addison Wesley, 1994.

[15] E. Burdet, R. Osu, DW. Franklin, T. Milner, and M. Kawato. The central nervous system stabilizes unstable dynamics by learning optimal impedance. *Nature*, 414:446–449, 2001.

[16] E. Burdet, R. Gassert, F. Mani, F. Wang, Teo. C.L., and H. Bleuler. Design of a haptic forceps for microsurgery training. In *Proceedings of Eurohaptics 2004*, 2004.

[17] M. Kawato. Internal models for motor control and trajectory planning. *Current Opinion in Neurobiology*, 9:718–727, 1999.

[18] HD Zimmer et al. Memory for action: A distinct form of episodic memory? In *Counterpoints*. Oxford University Press, 2000.

[19] RN. Lemon, RS. Johansson, and G. Westling. Corticospinal control during reach, grasp, and precision lift in man. *Journal of Neuroscience*, 15:6145–6156, 1995.

[20] R. Bhushan, N.and Shadmehr. Computational nature of human adaptive control during learning of reaching movements in force fields. *Biological Cybernetics*, 81(1):39–60, 1999.

[21] T. Flash and E. Henis. Arm trajectory modifications during reaching toward visual targets. *Journal of cognitive Neuroscience*, 3(3):220–230, 1991.

[22] H. Zhang, E. Burdet, et al. Robotic micro-assembly of a scaffold/cell constructs with a shape memory alloy gripper. In *Proceeding of IEEE International Conference on Robotics and Automation*, 2002.

[23] F. Tendick, M. Downes, T. Goktekin, et al. A virtual environment testbed for training laparoscopic surgical skills. *Presence*, 9(3):236–255, June 2000.

[24] R. Alterovitz, J. Pouliot, et al. Simulating needle insertion and radioactive seed implantation for prostate brachytherapy. In J.D. Westwood et al, editor, *Medicine Meets Virtual Reality 11 (MMVR11)*, pages 19–25. IOS Press, January 2003.

[25] M. Mahvash and V. Hayward. Haptic simulation of a tool in contact with a nonlinear deformable body. In H. Delingette N. Ayache, editor, *International Symposium on Surgery Simulation and Soft Tissue Modelling, IS4TM, Lecture Notes in Computer Science*, Juan-Les-Pins, France, June 2003. Springer Verlag.

[26] D. Simone and A.M. Okamura. Haptic modeling of needle insertion for robot-assisted percutaneous therapy. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2085–2091, 2002.

[27] C.W. Kennedy, T. Hu, J.P. Desai, A.S. Wechsler, and J.Y. Kresh. A novel approach to robotic cardiac surgery using haptics and vision. *Cardiovascular Engineering: An International Journal*, 2002.

[28] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, Anacheim, California, 1987.

[29] JP. Gourret, N.M. Thalmann, and D. Thalman. Simulation of object and human skin de-formations in a grasping task. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 21–30, 1989.

[30] R. Baumann and D. Glauser. Force feedback for virtual reality based minimally invasive surgery simulator. In *Medicine meets virtual reality*, San Diego CA, 1996.

[31] P. Meseure and C. Chaillou. Deformable body simulation with adaptive subdivision and cuttings. In *Proceedings of the WSCG'97*, pages 361–370, 1997.

[32] UG. Kuehnapfel and B. Neisius. Cad-based graphical computer simultaion in endoscopic surgery. *End. Surg*, 1:181–184, 1993.

[33] M. Sagar, D. Bullivant, G. Mallinson, and P. Hunter. A virtual environment and model of the eye for surgical simulation. In *Computer Graphics Proceedings, Annual Conference Series*, pages 205–212, 1994.

[34] E. Keeve, S. Girod, P. Pfeifle, and B. Girod. Anatomy-based facial tissue modeling using the finite element method. In *IEEE Visualization 96*, pages 21–28, 1996.

[35] S. Peiper, J. Rosen, and D. Zeltzer. Interactive graphics for plastic surgery: A task-level analysis and implementation. In *Symposium on Interactive 3D Graphics*, 1992.

[36] O.R. Astley and V. Hayward. Multirate haptic simulation achived by coupling finite element meshes through norton equivalents. In *Internal conference on Robotics and Automation*, 1998.

[37] M. Bro-Nielsen and S. Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Proceedings of Eurographics*, volume 15 of *3*, pages 57–66, 1996.

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

[39] S.P. DiMaio and S.E. Salcudean. Simulated interactive needle insertion. In *Haptic Symposium, IEEE Int. Conference on Virtual Reality*, 2002.

[40] X. Wu and M.S. Downes. A adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Eurographics*, volume 20, pages 349–358, September 2001.

[41] K.M. Lim, F. Wang, T. Poston, L. Zhang, C.L. Teo, and E. Burdet. Multi-scale simulation for microsurgery trainer. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[42] K.M. Lim, T. Poston, L. Zhang, B.F. Liu, C.L. Teo, and E. Burdet. Multi-scale simulation for a robotic surgical trainer. In *ICBME*, 2002.

[43] C. Shene and J. Johnstone. On the planar intersection of natural quadrics. In *Proceedings of ACM Solid Modeling*, pages 234–244, 1991.

[44] J.R. Miller. Geometric approach to nonplanar quadric surface intersection curves. *ACM Transactions on Graphics*, 6(4):274–307, 1987.

[45] L. Piegl. Geometric method of intersecting natural quadrics represented in trimmed surface form. *Comput. Aided Des.*, 21(4):201–212, 1989.

[46] RT Farouki, CA Neff, and M. O'Connor. Automatic parsing of degenerate quadric-surface intersections. *ACM Transactions on Graphics*, 8:174–203, 1989.

[47] LV Ahlfors. *Complex Analysis - An Introduction to theTheory of Analytic Functions of One Complex Variable*. McGraw Hill, 1979.

[48] http://www.uni-koeln.de/math-nat-fak/phchem/deiters/quartic/quartic.html.

[49] S. Gottschalk, M. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *Proc. ACM Siggraph'96*, pages 171–180, 1996.

[50] J. Phillips, A. Ladd, and L.E. Kavraki. Simulated knot tying. In *Proc. IEEE International Conference on Robotics and Automation*, pages 841–846, Washington, DC, May 2002. IEEE Press.

[51] A. Dhanik, T. Poston, K.M. Lim, C.L. Teo, and E. Burdet. Virtual nylon thread manipulation for a haptic microsurgery training environment (submitted). 2004.

Appendix A

Navier's Series Solution Formulation

The governing equation of anti-plane elasticity problem is:

$$\nabla(G\nabla w) + P = 0 \tag{A.1}$$

A series solution for LxB rectangular region may be found by Fourier expansion of the load and deflection

$$p(x,y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} P_{mn} sin\frac{m\pi x}{L} sin\frac{n\pi y}{B} \tag{A.2}$$

$$w(x,y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} W_{mn} sin\frac{m\pi x}{L} sin\frac{n\pi y}{B} \tag{A.3}$$

The coefficients of this double Fourier expansion are given by:

$$P_{mn} = \frac{4}{LB}\int_0^L \int_0^B p(x,y) sin\frac{m\pi x}{L} sin\frac{n\pi y}{B} dydx \tag{A.4}$$

For a line force,

$$P(x,y) = F\prod(x, x_1, x_2)\delta(y - y_1) \tag{A.5}$$

$$P_{mn} = \frac{4}{LB}\int_0^L \int_0^B F\prod(x, x_1, x_2)\delta(y - y_1)sin\frac{m\pi x}{L}sin\frac{n\pi y}{B}dydx$$
$$= \frac{4F}{m\pi b}sin\frac{n\pi y_1}{B}\left[cos\frac{m\pi x_1}{L} - cos\frac{m\pi x_2}{L}\right] \tag{A.6}$$

From equation A.3,

$$\nabla(G\nabla w) = G\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} -W_{mn}\left[\left(\frac{m\pi}{L}\right)^2 + \left(\frac{n\pi}{L}\right)^2\right] sin\frac{m\pi x}{L}sin\frac{n\pi y}{B} \tag{A.7}$$

Substituting this into A.1 gives:

$$\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\left\{-W_{mn}\left[\left(\frac{m\pi}{L}\right)^2 + \left(\frac{n\pi}{L}\right)^2\right] + \frac{P_{mn}}{G}\right\} sin\frac{m\pi x}{L}sin\frac{n\pi y}{B} = 0 \tag{A.8}$$

The coefficients of this equation are identically zero,

$$W_{mn}\left[\left(\frac{m\pi}{L}\right)^2 + \left(\frac{n\pi}{L}\right)^2\right] - \frac{P_{mn}}{G} = 0 \tag{A.9}$$

This gives:

$$W_{mn} = \frac{P_{mn}}{G}\frac{1}{\left(\frac{m\pi}{L}\right)^2 + \left(\frac{n\pi}{L}\right)^2} \tag{A.10}$$

Inserting equation A.6 and A.10 into equation A.3,

$$w(x, y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\frac{4Fsin\frac{n\pi y_1}{B}\left[cos\frac{m\pi x_1}{L} - cos\frac{m\pi x_2}{L}\right] sin\frac{m\pi x}{L}sin\frac{n\pi y}{B}}{GBm\pi^3\left[\left(\frac{m}{L}\right)^2 + \left(\frac{n}{L}\right)^2\right]} \tag{A.11}$$

The above series solution for the anti-plane elasticity problem is used to check the accuracy of our FEM results. Fig. A.1 shows the force values along the bar using the Navier's solution with different number of terms m, and the results converge rapidly as m increases.

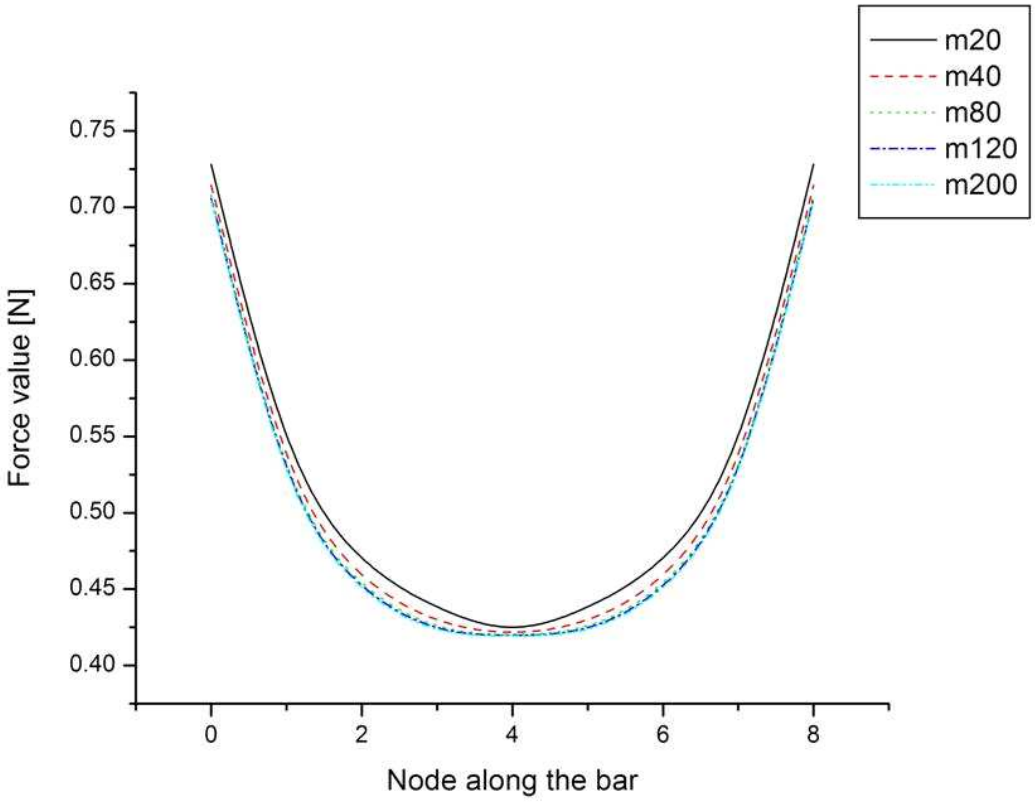

Figure A.1: Force value along the bar

Appendix B

Relations between R, r and α

To connect the torus form (4.1) with the projected α version, we substitute

$$l = \frac{2x}{x^2+y^2+z^2+1} \quad m = \frac{2y}{x^2+y^2+z^2+1}$$
$$p = \frac{2z}{x^2+y^2+z^2+1} \quad q = \frac{x^2+y^2+z^2-1}{x^2+y^2+z^2+1}$$

from (4.3) into (4.4). We have $\alpha^2\left(l^2+m^2\right)-\left(p^2+q^2\right)=0$ exactly at the zeroes of $\epsilon\left(x^2+y^2+z^2+1\right)^2\left(\alpha^2\left(l^2+m^2\right)-\left(p^2+q^2\right)\right)$ where ϵ is an arbitrary constant. We want to choose R and r such that

$$\left(\rho^2-D^2\right)^2-4\left(x^2+y^2\right)R^2 = C\left(\left(\rho^2+1\right)^2\left(\alpha^2\left(l^2+m^2\right)-\left(p^2+q^2\right)\right)\right)$$

where $\rho^2=x^2+y^2+z^2$, $D^2=R^2-r^2$. But then

$$\rho^4-2\rho^2D^2+D^4-4\left(x^2+y^2\right)R^2-C\left(\alpha^2\left(4x^2+4y^2\right)-4z^2-\left(\rho^2-1\right)\right)$$

vanishes at infinitely many points, which is only true if the coefficients of x^n, y^n, z^n vanish identically. Comparing the coefficients of ρ^4, we want $\epsilon=-1$, *i.e.*,

$$\left(2D^2-4R^2+2+4\alpha^2\right)x^2+\left(2D^2-4R^2+2+4\alpha^2\right)y^2+\left(2D^2-2\right)z^2+D^4-1=0$$

The vanishing of coefficients reduces to

$$\begin{aligned} D^4 - 1 &= 0 \\ 2D^2 - 2 &= 0 \\ 2D^2 - 4R^2 + 2 + 4\alpha^2 &= 0 \end{aligned} \tag{B.1}$$

We want a torus with $R > r$, so we choose

$$R^2 - r^2 = 1. \tag{B.2}$$

It then follows that

$$\begin{aligned} 4R^2 - 4\alpha^2 &= 4 \\ R^2 - \alpha^2 &= 1 \\ r &= \alpha \end{aligned} \tag{B.3}$$

For small values of α we get a thin torus containing the unit circle $x^2 + y^2 = 1$ in the $z = 0$ plane but centred on $x^2 + y^2 = 1 + \alpha^2$. For large values we get a fat torus, with a z-axis hole of throat radius

$$\begin{aligned} \sqrt{1+\alpha^2} - \alpha &= \alpha \left(\sqrt{1 + \frac{1}{\alpha^2}} - 1 \right) \\ &= \alpha \left(\left(1 + \frac{1}{2\alpha^2} + \ldots \right) - 1 \right) \\ &\approx \frac{1}{2\alpha} \end{aligned} \tag{B.4}$$

and outside radius approximately α.

Note that (B.2) forces $r = \sqrt{R^2 - 1}$, regardless of α. To get a general torus, with arbitrary $\widetilde{R}$ and $\widetilde{r}$, define their ratio $\beta = \widetilde{r}/\widetilde{R}$. We produce a torus with this

ratio by setting

$$\beta^2 = \frac{\alpha^2}{1+\alpha^2} \qquad \text{(B.5)}$$
$$\alpha^2 = \frac{\beta^2}{1-\beta^2}\,,$$

which gives $R^2 = 1/(1-\beta^2)$. If this is not the value $\widetilde{R}$ needed, scale (x, y, z) by the factor $\widetilde{R}/\sqrt{1-\beta^2} = \sqrt{\widetilde{R}^2 - \widetilde{r}^2}$. For instance, if $\widetilde{R} = 5$ and $\widetilde{r} = 3$, giving $\beta = 3/5$, we want to draw the $\alpha^2 = 9/16$ torus (which unscaled would have $R = \sqrt{1+9/16} = 5/4$ and $r = \sqrt{9/16} = 3/4$) enlarged by the factor $\sqrt{5^2 - 3^2} = 4$.