DEVELOPMENT OF A VIRTUAL REALITY BASED MICROSURGERY TRAINER

TEO CHENG YONG, WILLIAM

(B.Eng.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2005

Acknowledgments

I would like to express my most sincere gratitude and appreciation to my two supervisors, Dr. TEO Chee Leong and Dr. Etienne BURDET for their valuable insights and guidance on my research.

I would also like to thank Dr. LIM Kian Meng, Dr. Tim POSTON, Dr. LIM Beng Hai, Ankur DHANIK, James RAPPEL, Roger GASSERT, TEE Keng Peng and WANG Fei for their collaboration and help on the micromanipulation learning project.

I would also like to thank Mr. YEE, Mrs. OOI, Ms. TSHIN, Ms Hamiah, Mr. ZHANG, Mr. TAN, Mr. Roger and the numerous lab technicians at the National University of Singapore.

Last but not least, I could not have completed my research without the financial assistance provided by the National University of Singapore.

Table of Contents

Summary

This thesis describes a project to develop a virtual environment for simulating the grasping and inserting of a micro-needle using a Virtual Reality (VR) based training system. The project is part of the development of a Virtual Reality (VR) based training system for microsurgery in collaboration with the National University Hospital (NUH).

While a realistic environment is important for any virtual learning, a complex task such as microsurgery can be better taught by decomposing it into several simple dexterity primitives. And these primitives are designed based on identified features which are significant to the given task. In particular, the ability to properly grasp a suturing needle is identified as an important skill/subtask in microsurgery. A suturing needle is a tiny curved needle used to suture blood vessels and tissues in microsurgery. It is easily deformable because of its minute size. In addition, the curvature inherent to a suturing needle further increases the complexity when grasping with a needleholder. Hence, significant skill is required to grasp a suturing needle without deforming or breaking it. Also, as the workspace for any microsurgerical operation is restricted, the surgeons are usually required to suture at a variety of orientations. And when suturing at an unnatural angle, the surgeons are required to be able identify a suitable grasping orientation and position, and to perform a proper grasp before inserting the micro-needle.

In this project, a physics engine to simulate a virtual environment of the microsurgery is developed and a set of teaching modules is implemented to train the suturing subtask of grasping and inserting of a micro-needle. In addition, a haptic needleholder is fabricated and used by the user to manipulate the micro-needle and suture in the virtual environment. Computationally efficient and realistic models of the needle, needleholder

and the suture/thread are developed and implemented in a series of tutorials to train grasping and insertion of the virtual needle in the virtual environment. In addition, a library of APIs which permit rapid generation of custom modules is developed. These APIs are able to generate the various customized virtual objects and co-ordinate their interactions with the needleholder.

The haptic needleholder is modified from an actual needleholder and integrated with the Phantom^{tm} haptic device. The main modification done to the needleholder-stylus is the integration of a displacement sensor to provide sensory information on the state of the jaws of the needleholder and an estimate of the pressure applied on the grasped object. This haptic needleholder is used as an interface to the VR system.

The final setup together with the training modules developed using the library is tested on some subjects. Preliminary observations and results obtained show that extremely curved needles are harder to grasp and manipulate, but with more practice on the VR system, the difficulties arising from the differences between needles decrease.

List of Tables

Lists of Figures

1 Introduction

1.1 Microsurgery

Microsurgery is a form of surgery performed on minute body structures or cells with the aid of a microscope and other specialized instruments, such as a micromanipulator [1]. It is one of the leading medical practices and is used in vascular and neurosurgery, in traumatology, eye surgery and other branches of medicine where microscopes are used. In particular, microsurgery is used extensively for complex reconstruction of limbs in trauma, tumor resection, bone and joints, nerve, blood vessels, and for birth deformities.

An operating microscope, fine instruments, micro-suture and intensive trainings are required in order to perform precision acts required in a microsurgery. A typical microsurgery for hand reconstruction (example, trauma due to industrial accidents) involves trimming of severed blood vessels with typical diameter of 1mm and rejoining or modifying them to re-establish blood flow that is required for healing. The entire operation is usually done under the microscope with a magnification of x10 ~ x20.

Operating under magnification is a unique experience as the surgeon has to rely on purely visual cues in an entirely different environment. This can lead to mistakes and also requires a significant amount of training before the surgeon can perform any act with any precision. In this environment, tremors are greatly amplified and any mistakes will prove especially grave, as the tissues under operation are especially small and fragile. Hence, it is very

important that techniques are developed to minimize the tremor and for training to be conducted to accustom the surgeons to the unique conditions in microsurgery.

There are numerous factors which contribute to a good suture in microsurgery, one of which is the correct grasping of the micro-needle. The micro-needle is easily deformed; hence it is important that excessive force is not used when grasping the micro-needle. Moreover, it is important that the micro-needle is grasped at a proper orientation and on a specific region. This is because both the grasp orientation and region will greatly influence the ability to perform a proper suturing motion.

1.2 Motivation

1.2.1 Training of Microsurgery

The current training system for microsurgery generally involves a combination of practical observation of actual microsurgery by experienced practitioners, educational videos on proper techniques and postures, and actual practices on lab rats and medical cadavers.

There are several problems with the current system. It is difficult for the students to properly observe any surgery as the viewing ports are still constrained by the available workspace for the operation. There may be numerous cases where the hand or instrument of the surgeon will block the view port of the camera whereas in a virtual environment, only the tips of a forceps or needleholder are rendered while non-essential objects such as the surgeon's hand and the main body of the forceps or needleholder are filtered out.

Another problem is that preparation of a microsurgery practical requires significant amount of setup, which include booking of the surgical room, and pre-preparation of the surgical instruments. However with a virtual reality system, the training can be conducted with minimal preparations such as powering up the workstation and initializing the software.

Considerable expenses are also required for even a basic microsurgery course. This is mainly because expensive consumables, like micro-needles, micro-sutures, lab rats are used in every single lesson. Moreover, obtaining medical cadavers for practice is relatively expensive and difficult. A typical basic microsurgery course lasts 40 hours and cost US$1500 [3]. On the other hand, a virtual reality system does not have any consumables other than electricity. Hence, the only expenses for virtual reality training are the initial setup cost and general maintenance.

1.2.2 Training Grasping and Inserting of Micro-needle

The main motivation of the entire project is to develop a more effective complementary or alternative training system for microsurgery. And the project described in this thesis is a part of the bigger project as mentioned above.

As described previously, the main task in microsurgery is to rejoin severed blood vessels which require the use of micro-manipulators, micro-suture and micro-needles. And one of the important prerequisites in good suturing is a proper grasping of the micro-needle with the needleholder. Thus the motivations of this project are to develop a system to train the student in recognizing and performing the correct grasping of a micro-needle.

1.3 Decomposition of a Complex Microsurgical Task

In order to develop a cost effective and feasible training system, the key micro-manipulation tasks required in microsurgery were identified and broken down into several unique dexterity primitives [4,5,10,28]. The project described in this thesis is to develop the dexterity primitive to train grasping and inserting of a micro-needle. This approach is vastly different from established methodology employed by most groups [6,7].

1.3.1 Complex Microsurgical Task

A microsurgical task typically involves suturing of a severed blood vessel with diameter of 1 mm. The microsurgeon will first grasp the micro-needle with the needleholder. Next, he will insert the micro-needle into specific locations on the soft tissues. And finally, he will tie off the suture using the forceps and the needleholder. The entire task is done under a microscope.

The complex suturing task was broken into various dexterity primitives because studies suggest [8] that humans may form internal models of these primitives, in which these primitive can then be combined to perform more complex tasks. It is generally easier and faster to learn simple primitives before actually performing complicated tasks, and is being used in numerous teaching methodologies. A good example would be typing, where the students are required to practice typing individual sets of seemingly nonsensical letters before practicing typing out complete sentences.

Hence as shown in Figure 1.3-1, three main subtasks had been identified from this complex suturing procedure. They are namely,

- Grasping and Inserting of the micro-needle
- Precision Manipulation of the micro-instruments
- Loop formation and knot tying of micro-suture

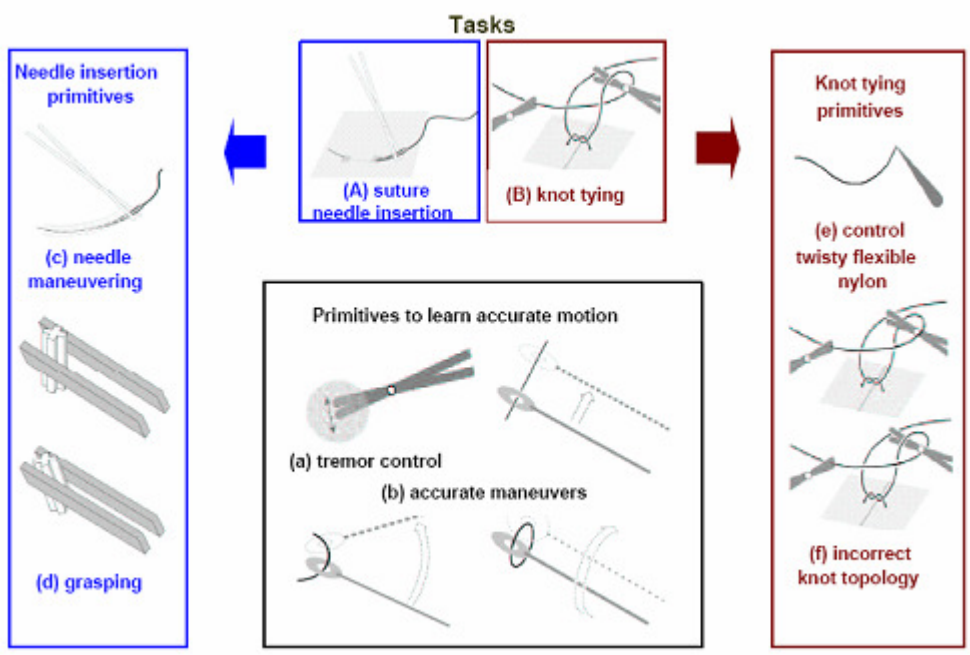


Figure 1.3-1 Breakdown of complex microsurgical task into dexterity primitives [4]

1.3.2 Grasping and Inserting of the micro-needle

Thus far, an analytical collision detection between a needle and a tube, together with a multi-scaled mesh had been developed by Wang [4,5]. The needle-tube collision is required to train the curved motion required in suturing with the micro-needle, while the multi-

scaled mesh improves the efficiency of the general tissue-needle collision required in the trainer. This project complements what Wang had done so far by developing the dynamic behaviors of the micro-needle, which is required to simulate the grasping of the micro-needle; in what Wang had done thus far, the micro-needle was always assumed to be grasped initially.

1.3.3 Precision Manipulation of the Micro-instruments

As the surgery was done under microscope, tremor and a precise manipulation of the micro-instruments play a significant part in a good suture. Several simple dexterity primitives like manipulating a ring around a hoop had been developed previously.

1.3.4 Loop formation and knot tying of micro-suture

Loop formation and knot tying of the suture is another important aspect of microsurgery. Significant works had been done by Ankur Dhanik to model a realistic behavior of a micro-suture [28].

1.4 Grasping and Inserting in Microsurgery

The main objective in this project is to develop a training system to teach grasping and inserting of the micro-needle. Hence it is necessary to understand the correct technique in grasping and inserting the micro-needle.

1.4.1 Anatomy of a Micro-Needle

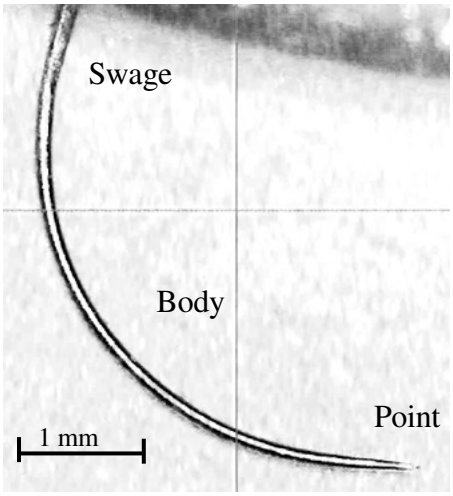


Figure 1.4-1 Photograph of a suturing micro-needle

As shown in Figure 1.4-1, a suturing micro-needle can be broken down into 3 main parts, the point, the body, and the swage. The point of a needle consists of the portion from the tip of the needle to the maximum cross-section of the body. The body of the needle consists of the majority of the needle. It is important for interaction with the needle holder and the ability to transmit the penetrating force to the point. The needle factors affecting this interaction include needle diameter and radius, body geometry, and stainless steel alloy. These components determine the needle-bending moment, ultimate moment, surgical-yield moment, and needle ductility. The swage is the continuation of the needle onto the suture, as in the needle and suture is one continuous entity [14]. The typical shape of a needle can be from ¼ ~ 5/8 of a full arc.

1.4.2 Grasping Procedure

A typical method to grasp the needle is to grasp it at the distal portion of the body, one half to three quarters of the distance from the tip of the needle, depending on the surgeon's preference. Grasping of the needle at its proximal or distal extremities is avoided in order to prevent damage to the suture. The needle should be held vertically and longitudinally perpendicular to the needle holder. The pressure applied should be sufficiently firm that the needle do not slip during insertion, but not overly excessive such that straightening or deformation of the needle occurs [13]. The ideal pressure would be a light but firm contact with the needle throughout the suturing process.

The grasping of the needle at an incorrect position and/or pressure could potentially damage the needle, making it unusable (and thus increasing the overall operation time). The grasping of the needle at an incorrect position and/or orientation on the other hand will result in difficulties when maneuvering to a proper insertion position.

The procedure to a good grasping of a micro-needle can be summarized as follow:

- Identify the proper grasping location on the needle
- Approach at a proper orientation such that the needle is vertically and longitudinally perpendicular to the needle holder
- Grasp the needle with the minimal pressure required to prevent slippage during insertion

1.4.3 Inserting Procedure

The needle should always penetrate the tissue at a 90° angle, which minimizes the size of the entry wound and promotes eversion of the tissue edges. The needle should be inserted 1-3 mm from the wound edge, depending on tissue thickness. The depth and angle of the suture depends on the particular suturing technique. In general, the 2 sides of the suture should become mirror images, and the needle should also exit the tissue perpendicular to its surface [13].

The procedure to a good insertion of a micro-needle can be summarized as follow:

- Identify suitable insertion locations on the tissue
- Position the needle such that it will penetrate the tissue at 90° angle
- Insert the needle in a curvilinear motion such that the needle body in contact with the tissue is always perpendicular
- Release the needle when it is firmly attached to the tissue, and when further insertion is difficult
- Grasp the needle on the other side of the tissue and continue the insertion with a curvilinear pulling motion where the needle would exit the tissue perpendicularly

1.5 Methodology

A Virtual Reality (VR) based system was selected as the alternative/complementary medium to train microsurgery. The various techniques required in microsurgery were broken down into simpler dexterity primitives, which are then taught separately in a virtual environment. In order to improve learning, a Phantom^{tm} Desktop in a Reachin setup was selected as the template for the training system. The Reachin setup imitates an actual microsurgical setup relatively closely, and the Phantom^{tm} Desktop is used as a spatial input device for system. The Phantom^{tm} Desktop is a haptic device capable of detecting 6 DoF movement and producing 3 DoF haptic cues, which can be used to generate pseudo-realism environment to influence learning [9,10,11]. A surgical needleholder was modified with a displacement sensor and integrated with the Phantom^{tm} Desktop to provide addition inputs, such as jaws position and pressure of the needleholder. The needleholder is one of the major instruments used in microsurgery, to grasp and manipulate various microsurgical objects such as micro-needles and micro-sutures.

In this project, other than developing a device driver for the DVRT displacement senor, various algorithms were investigated to simulate the virtual dynamics of various microsurgical objects. Collision detection was identified to be a bottleneck for computational time. Hence optimization techniques were investigated to improve detection efficiency. These algorithms and techniques were then implemented into a physics engine in a form of a C++ static library. The static library comprises of a lists of APIs which can be used to quickly develop customized virtual environments for microsurgical training. The library is able to generate and co-ordinate the interactions between the computationally

efficient and realistic models of the needle, needleholder and the suture/thread. A series of customized modules was developed using the static library. These tutorial modules were designed to train the students in the dexterity primitive of grasping and inserting a micro-needle. A combination of visual and haptic cues was also used in the simulator to improve the effectiveness of the training system. In addition, stereoscopic graphics were also implemented for the system [21]. The developed training modules were able collected several key data such that performance reviews can be done after a training session. A trial run was conducted with the developed system to test the functionality of the software.

1.6 Contributions

In this project, the following objectives have been achieved,

- A physics engine, in a form of a C++ static library, to simulate and generate virtual environments for microsurgery has been developed.
- A series of tutorial modules to train the subtask of grasping and inserting a micro-needle has been developed using the above mentioned static library.
- A fixture to integrate the needleholder, DVRT, and Phantom^{tm} Desktop, has been fabricated.
- A driver for DVRT to communicate with the physics engine has been developed.
- A trial run to verify the functionality of the system has been conducted.

1.7 Organization

The organization of the thesis is as follows:

In Chapter 1, a brief introduction to microsurgery, and the motivation and methodology of the project were presented. Related sub-projects were also briefly described.

In Chapter 2, an overview on the VR-based system and its physical setup were described.

In Chapter 3, the development of a modified needleholder for the VR-based trainer was described.

In Chapter 4, the various collision detection algorithms and virtual dynamics used in the physics API library were described.

In Chapter 5, the implementation of the various algorithms into a physics API library was described.

In Chapter 6, the performance of the physics library was presented. In addition, the various modules developed for training grasping and inserting were described. The key factors for performance evaluation of the students were also described and a trial run was conducted with the training software.

In Chapter 7, the conclusion and future works for this project are described.

2 Virtual Reality Based Training System

In this project, virtual reality integrated with haptic feedback was selected as the alternative mode of training for microsurgery. In this chapter, the various advantages and disadvantages of a virtual reality training system are being discussed. In addition, the training system setup and its components are also presented.

2.1 Background

Computer related technologies have advanced significantly in the past few decades. Virtual reality (VR) environment is one application of such technology and it has tremendous potential for surgical trainings [6, 7, 15, 16]. A VR environment is a simulation of the real world environment that is generated by computer software. The human user interacts with the VR environment via an interface generally consisting of the keyboard, mouse, monitor and perhaps a haptic device.

There are numerous advantages in using a VR trainer over the traditional training methods of practicing on cadavers and lab rats. A VR trainer requires very little setup time – one just needs to switch on the machines and run the program. A VR-based system also not requires significant resources and space – one just need a workstation with its human-machine interface, and the only significant expense is the electricity consumed by the setup. A VR-based system can also keep track of the progress of individual students, which then can be used as teaching aids and to improve the training methodology of the system. Moreover, it is entirely safe for the students, as there is little or no negative consequences when mistakes

are made in a VR environment – no precautions or clean up required for a VR-based trainer. Although a VR trainer has numerous advantages, one of the main challenges is to develop a realistic tactile interface with the virtual environment. Figure 2.1-1, shows pictures of microsurgical training on rats versus on a VR training system.

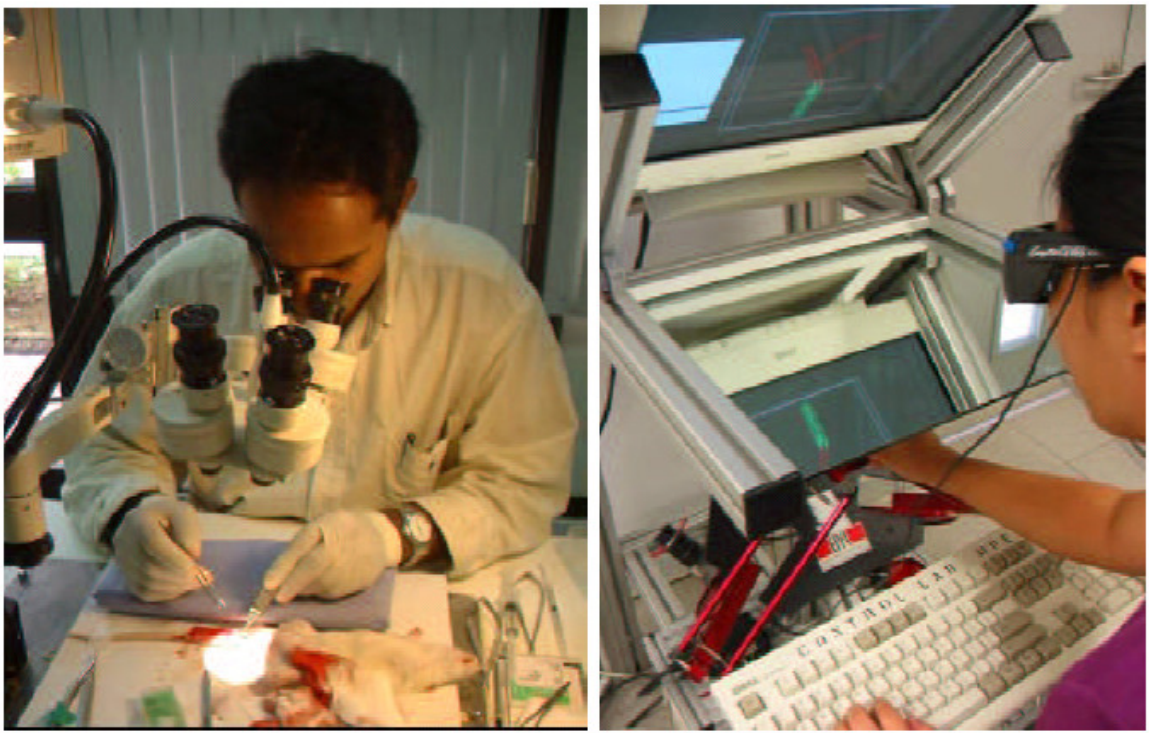


Figure 2.1-1 Microsurgical training on rats versus on VR training system

Other than the various interface devices described in previous chapters, a VR-based trainer also requires a physics engine to simulate dynamic behaviors of virtual objects used in a virtual microsurgery. A physics engine is essentially a library of APIs which detect and co-ordinates the numerous collisions between various virtual objects and implements their resultant dynamic behaviors.

2.2 System Setup

The main component of a VR-based trainer is the workstation and the typical user input devices are the keyboard and/or the mouse. However, to properly train the students with virtual reality, a realistic environment is essential. Hence, an input device that can simulate the feel of holding a needleholder and/or forceps (which are the tools commonly used in microsurgery) is required. To further improve the realism of the training, the simulator was developed as a stereoscopic software setup in the form of a Reachin Display, where with the aid of a pair of stereoscopic glasses, the students will be able to experience the virtual environment in 3D.

2.2.1 Reachin Setup

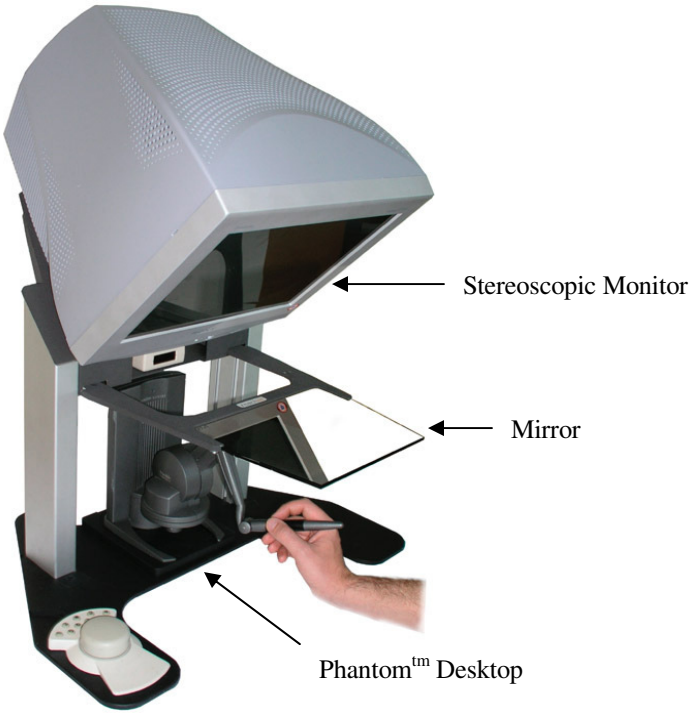


Figure 2.2-1 Reachin Display [31]

The Reachin Display was selected as the interface setup for the VR-based trainer. A Reachin Display setup is as shown in Figure 2.2-1. It consists of a multi-scan stereoscopic monitor positioned in such a way that the image generated will be reflected by a high grade surface mirror (to reduce ghosting effects). A haptic device is then positioned below the mirror such that the actual spatial position of its stylus synchronizes with its reflected image in the mirror. This setup is able to replicate the actual microsurgery setup quite closely.

2.2.2 Phantom^{tm} Desktop

Figure 2.2-2 Sensable Phantom^{tm} Desktop

As shown in Figure 2.2-2, the Phantom^{tm} Desktop by Sensable was selected as the haptic device for the Reachin Display. The main function of a haptic device in this VR-based trainer is to provide 6 DoF spatial inputs (3 for position, 3 for orientation) to the virtual environment, which can then simulate and display the virtual interactions between the user and the environment.

The Phantom^{tm} Desktop was selected because it possesses 3 DoF positional sensing, 3 DoF orientation sensing, portable design and a compact workspace ideal for microsurgery. In addition, the Phantom^{tm} is able to provide a force feedback of up to 1.75 N. This force feedback with a suitable bandwidth is required to provide a more realistic environment and potentially assist learning.

The user can interact with the virtual environment via the Phantom^{tm} desktop's stylus, such that the user provides the virtual position and orientation of the virtual needleholder via the stylus. However, in microsurgery there is an additional DoF – the movement of the jaws of the needleholder. Hence, a surgical needleholder was modified and integrated with the Phantom^{tm} Desktop. The development of the modified needleholder will be further described in Chapter 3 of the thesis. Additional information of the Phantom^{tm} Desktop can be found in Appendix A.

3 Modified Needleholder

The haptic device was not sufficient to provide a good interface for the virtual reality based training system. An actual surgical needleholder was modified with a displacement-sensor and integrated into the haptic device. In this chapter, the need for a customized input device was discussed. The design of the fixture to integrate the DVRT, needleholder, and Phantom^{tm} Desktop together was presented. In addition, the methodology to estimate jaw angle and pressure is presented. The chapter concludes with a description of a program developed to calibrate the modified needleholder.

3.1 Background

While the Phantom^{tm} Desktop is capable of providing the spatial position and orientation of the virtual needleholder, the stylus however does not have the "feel" of an actual needleholder used in microsurgery. Hence, an actual needleholder integrated to the stylus would probably improve the realism of the simulation.

Although the Phantom^{tm} stylus is equipped with a button which can be adapted to simulate the jaws movement of the virtual needleholder, it does not provide the full degree-of-freedom required. The button is only capable of providing a binary input (jaws open / jaws close), whereas we need an input that is able to represent the analog movement of the jaws.

Moreover, one of the training objectives of the simulator is train the students in their control of the applied jaws pressure. Excessive force in grasping will result in a poor grip

or a deformed micro-needle. Hence, it is obvious that a binary button input simulating the closing of the jaws of the needleholder is wholly insufficient and unrealistic.

As show in Figure 3.1-1, a subminiature Differential Variable Reluctance Transducer (DVRT) by MicroStrain was introduced into the system. The main function of the DVRT is to measure the displacement of the needleholder's grips, and thus the jaws separation angles can be computed. The DVRT has a stroke length of 8 mm and resolution of 1.9 micron (See Appendix B). Hence, an actual surgical needleholder was modified to include a DVRT, and then integrated with the Phantom^{tm} Desktop, to provide a more realistic interface with the virtual environment.

Figure 3.1-1 MicroStrain subminiature DVRT and DVRT reader

3.2 Modified Needleholder

The surgical needleholder is a tool used by microsurgeons to manipulate and grasp various surgical objects such as micro-needles and microsutures, in order to perform various techniques required in a microsurgery. As mentioned previously, in order to improve realism in the input device, an actual surgical needleholder was modified to be adapted onto the stylus of the Phantom^{tm}. The DVRT sensor was also integrated into the needleholder, such that, the displacement sensor is able to provide real-time input of the jaws separation angles. Thus a fixture to hold the DVRT and mount the needleholder onto the stylus was designed and fabricated as shown in Figure 3.2-3 .

The fixture comprises of three main components, the stylus adaptor, the DVRT adaptor, and the needleholder adaptor as shown in Figure 3.2-1 and Figure 3.2-2. The stylus fixture is a two-piece clamps with circular jaws, which is used to tighten around the body of the stylus. The dimension of the jaws was designed such a way that it slightly smaller than the diameter of the stylus's body. Additionally, screw threads are included such that the adaptor can be bolted onto the needleholder adaptor.

The DVRT adaptor is made up of a pair of swiveling holders with a through-bore. The core and shaft of the DVRT are tightened to the holders through the use of screws at the side of the holders. The main reason that the holders are free swiveling (free rotation in the z-axis) is to permit the freedom of movements for the DVRT shaft and core when the needleholder is being manipulated. This is because the needleholder is pivoted as shown in Figure 3.2-1, and any movements of the needleholder handles will be circular in nature, thus the adaptors

must be able to rotate to maintain a smooth sliding motion between the shaft and the core of the DVRT. Similar to the stylus adaptor, the holders are pivoted onto the needleholder adaptor.

The last component in the fixture is the needleholder adaptor. As its name suggests, the main function of the adaptor is to attach the rest of the components onto the needleholder. It comprises of a pair of holders which can be screwed onto the handles of the needleholder (screw threads are drilled onto the handles). The holders each has a platform to serve as attachment base for the other two components of the fixture. The detailed technical drawings for the fixtures can be found in Appendix C.

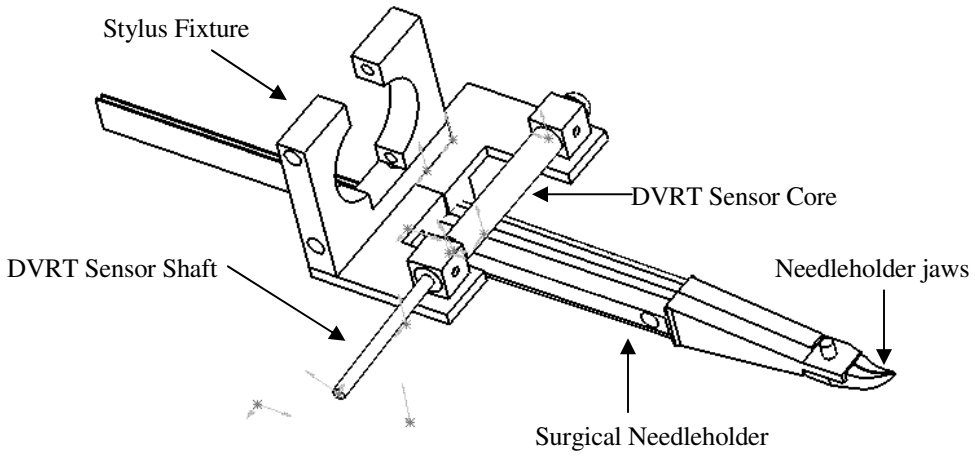


Figure 3.2-1 Engineering Drawing of the Modified Needleholder

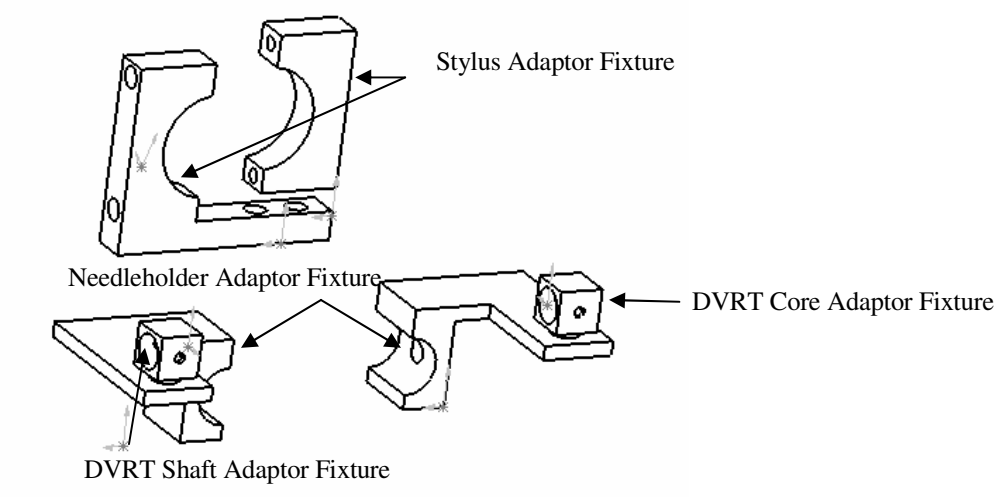


Figure 3.2-2 Engineering Part Drawing of the Fixture Components

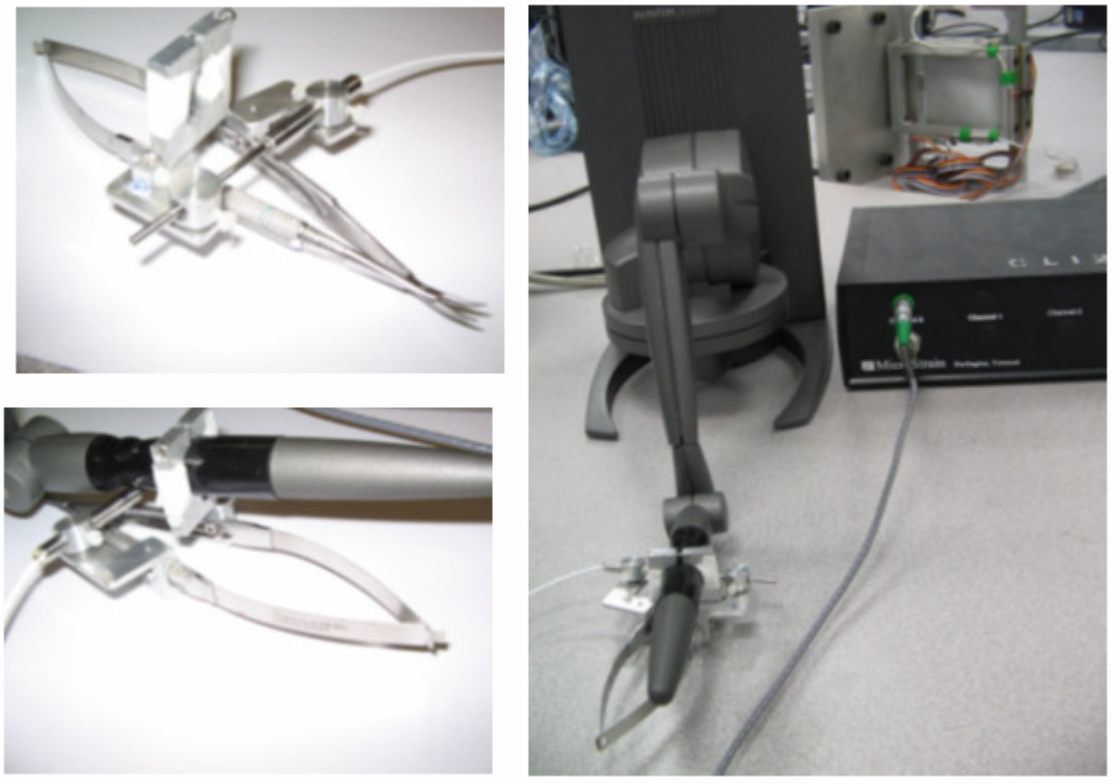


Figure 3.2-3 Modified Needleholder integrated with Phantom^{tm} Desktop and DVRT

3.3 Jaws Angle and Pressure Inputs

Other than the 6 DoF spatial inputs (position and orientation) from the Phantom^{tm} Desktop, there are two additional inputs from the DVRT: the jaws angle and jaws pressure. A win32 API device driver was written such that the voltage output of the DVRT can be obtained via the RS232 port and translated into the jaws angle and pressure.

It is obvious that the needleholder grips separation is directly related to the jaws angle. However, it isn't as obvious how the jaws pressure can be predicted from this separation value. As shown in Figure 3.3-1, when the jaws are being closed entirely, there is still a slight gap between the two grips. If sufficient force is being applied to the grips, the gap separation decreases. Thus, the jaws pressure can be predicted from the decrease.

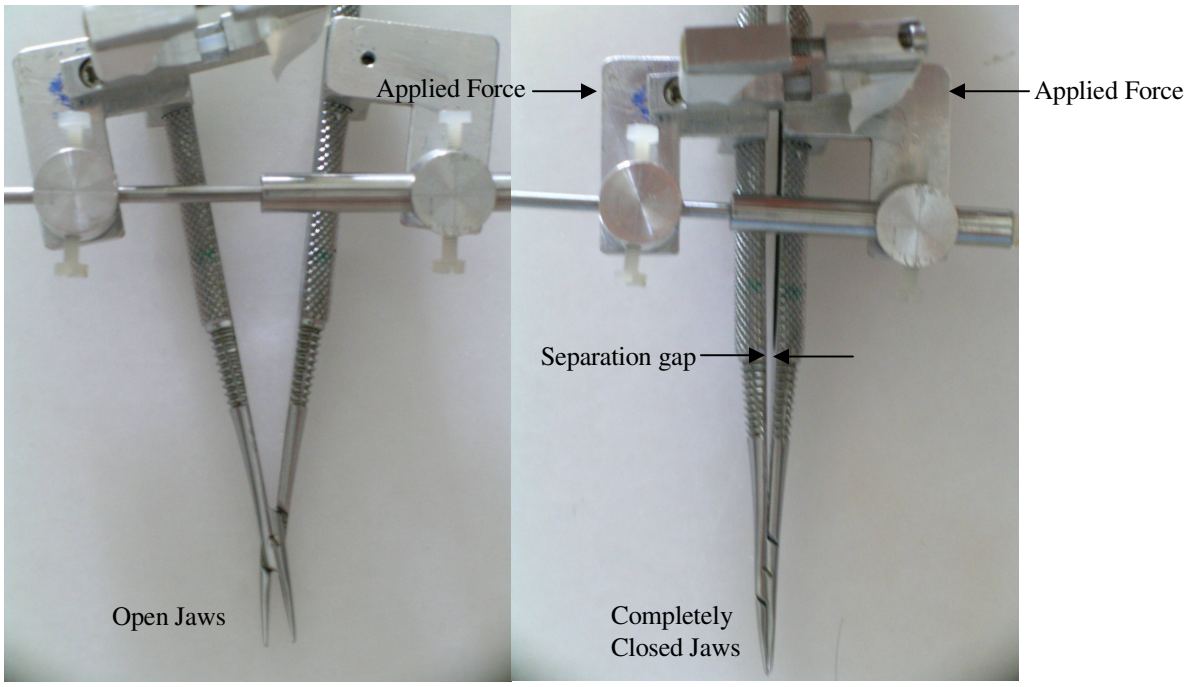
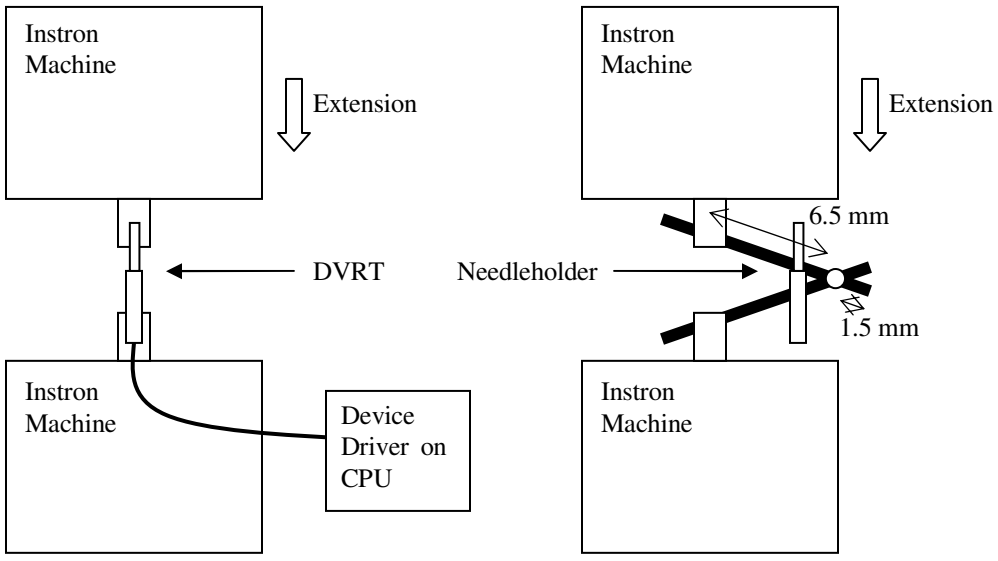


Figure 3.3-1 Estimating Jaws Pressure



(a) Calibration of DVRT (b) Force Calibration of Needleholder

Figure 3.3-2 Setup for Calibration of DVRT and Needleholder

As shown in Figure 3.3-2, an Instron machine was used to calibrate the DVRT and the needleholder. In order to calibrate the DVRT, the DVRT core and shaft were fitted onto the immobile and mobile fixture of the Instron machine respectively. The Instron machine was programmed to move the DVRT shaft with a constant displacement step. The voltage reading from the DVRT was recorded via a device driver, for each displacement step of the shaft. These voltage values were than plotted against the displacement values as shown in Figure 3.3-3. From the plot, it was verified that both the linearity (≈100%) and resolution of the DVRT were comparable to its specifications and within the requirements of this project. The irregularity at the beginning of the graph was due to the fact that the effective stroke length of the DVRT is 8 mm while the displacement measurement was conducted over a length of 10 mm. The voltage-extension factor obtained from the plot was used in the device driver written for the DVRT so that real-time displacement values can be

provided.

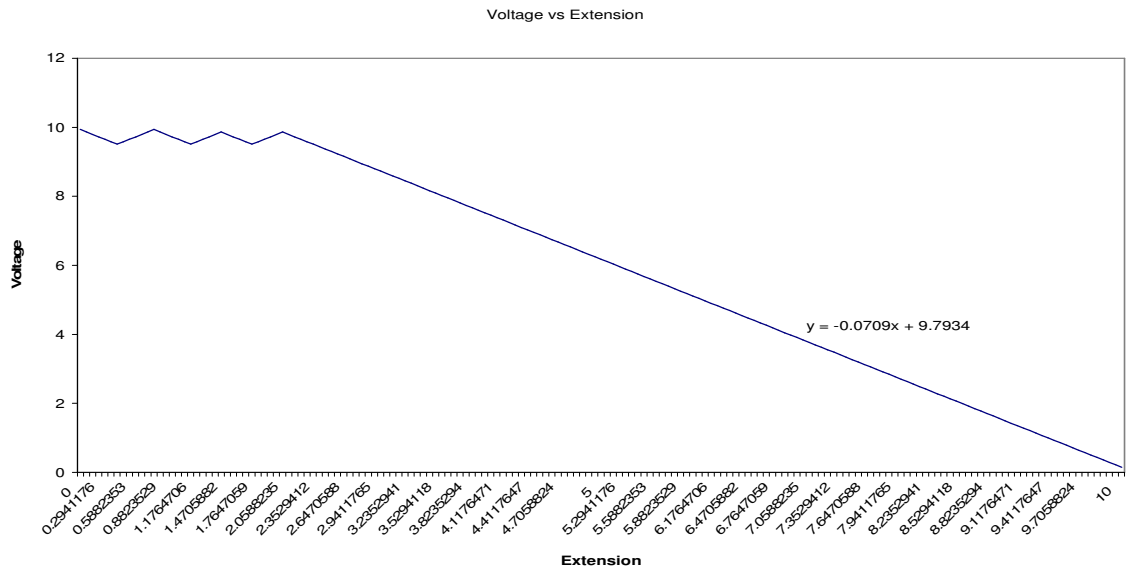


Figure 3.3-3 DVRT Calibration Graph (in volts versus millimeters)

Similarly, a calibration was done on the needleholder to determine the force exerted by the jaws of the needleholder. As shown in Figure 3.3-2, the Instron machine was used to measure the reaction force on the handle of the needleholder as it was depressed down. It can be found that about 0.2 N of force is required to just close the jaws as shown in Figure 3.3-1 while approximately 10 N of force is required to close the needleholder's handles completely. The reaction force from the Instron machine was plotted against the the movement of the needleholder's handles as shown in Figure 3.3-4.

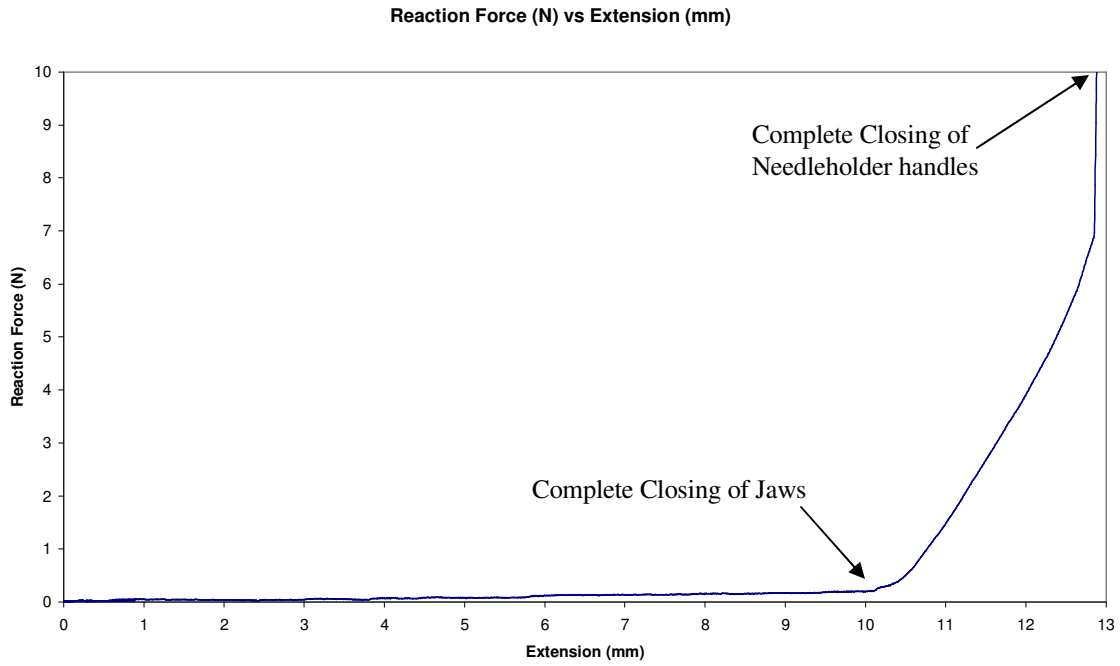


Figure 3.3-4 Force Calibration Graph (in newton versus millimeters)

Using basic geometry calculations and the conservation of moment, the approximate force experienced at the tip of the needleholder and the corresponding displacement of the DVRT, can be calculated and is plotted as shown in Figure 3.3-5. And by using the graph as a lookup table, the approximate force exerted by the needleholder on the needle can be predicted by the device driver. And by observing the force readings while grasping an actual micro-needle, it was found experimentally that a good grasping pressure would be approximately 0.6 N.

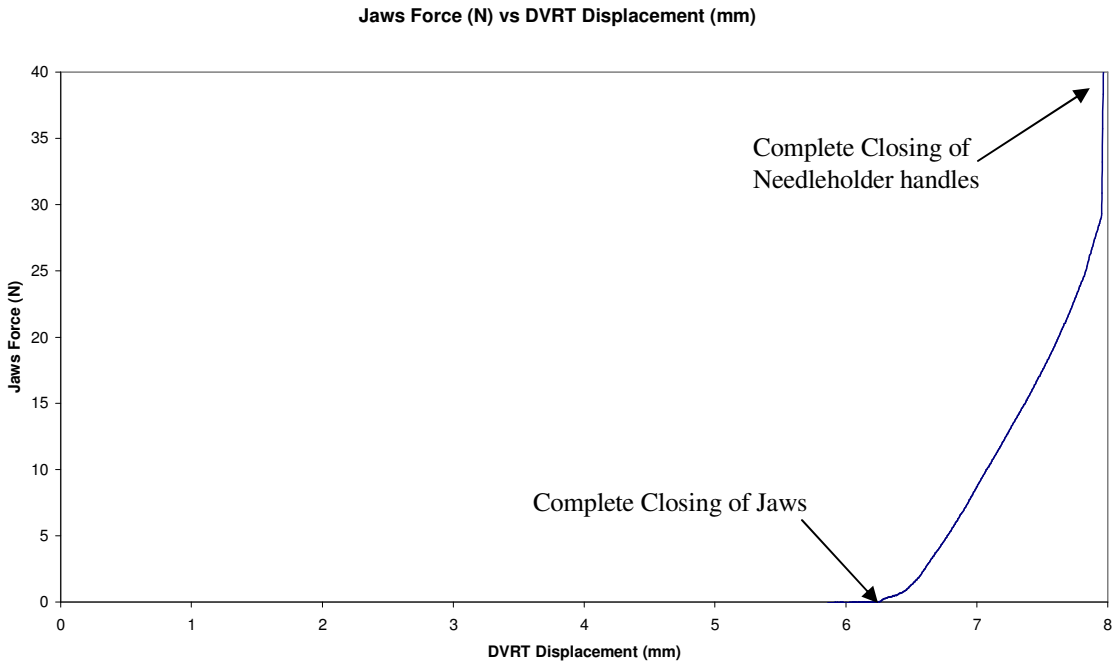


Figure 3.3-5 Jaws Force Calibration Graph (in newton versus millimeters)

4 Dynamics and Algorithms

Rigid body and particle mechanics are used to simulate the dynamic behaviors of the virtual objects found in the training system. This chapter is separated into two main sections, dynamics and algorithms. The various assumptions and equations used to simulate the virtual dynamics are described in this chapter. In addition, the various collision and data management algorithms are also presented.

4.1 Virtual Dynamics

4.1.1 Rigid Bodies Representation

Most of the virtual objects in the VR-based trainer are modeled as rigid bodies. A rigid body occupies a volume of space and has a particular shape, both of which are fixed. And since a particle is defined as a body whose spatial extent and internal motion and structure, if any, are irrelevant in a specific problem, any rigid body can be represented as a system of particles, at which, particle mechanics can then be applied to model rigid body behaviors [20, 27]. In short, a rigid body can be represented by the position vectors of a set of particles.

As a rigid body can only undergo translations and rotations, we can define the shape of the rigid body in a form a system of particles in a fixed space called body space. Given the geometric description (reference positions of all the individual particles) of a rigid body previously, we can then transform this description (positions of all the individual particles) into the world space using the translation vector $x(t)$ and the rotation matrix $R(t)$

experienced by the body. Hence, the position vectors of all the particles in the rigid body can be obtained using the Equation 4.1 below.

$$r_i(t) = R(t) \times p_i + x(t) \tag{4.1}$$

where

$r_i(t)$ is the position vector of the particles of the rigid body in world space at time t,

p_i is the reference vector of the particles of the rigid body in body space (with reference to a the center of mass of the body),

$R(t)$ is a 3x3 Rotation matrix defining the total rotation (about the center of mass) experienced by the object in world space at time t,

And $x(t)$ is the translation vector defining the total linear movement experienced by the body in world space at time t.

Typically, the center of mass of the rigid boy is used as the reference origin to describe its geometric shape (reference positions of the particles). As such, the translation vector $x(t)$ and rotational matrix $R(t)$ corresponds to the position vector, and orientation of the rigid body respectively.

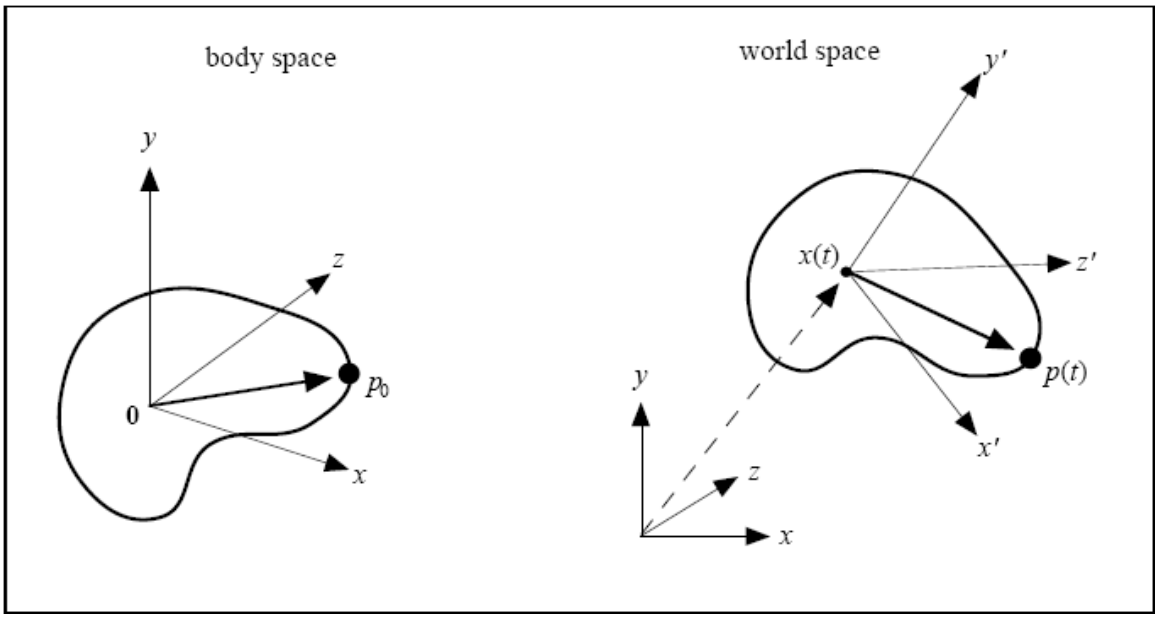


Figure 4.1-1 Representation of a Rigid Body in world space via body space description [27]

4.1.2 Dynamic Behaviors

In order to simulate the dynamics of a virtual object, we need to be able to predict its behavior (e.g. position and orientation) at each discrete time step. And in order to predict the position and orientation of an object with time, the dynamic properties of the objects that are required. These properties include the linear acceleration, the angular acceleration, the linear velocity, and the angular velocity. Hence, the dynamic behaviors of the virtual object can be typically represented with several ordinary differential equations (ODE) of the form

$$\dot{x} = f(x,t) \tag{4.2}$$

where

f(x,t) is a known function (i.e. it can be evaluated given *x* and *t*),

x is the state of the system (e.g. position vector of the center of mass, orientation of the object),

and $\dot{x}$ is *x*'s time derivative (e.g. velocity vector of the object, angular velocity of the object).

Hence, predicting the dynamic behavior of a virtual object is an initial value problem. For the case of simulating a virtual environment, numerical solutions with discrete time steps are used, such that, we can use the derivative function *f* to calculate the approximate change in *x*, Δx, over a time interval, Δt, then increment *x* by Δx to obtain the new value. Hence, we have to perform derivative evaluation at each time step to predict the state of the virtual object at the next time step. One of the simplest and commonly used numerical methods is called the Euler's method.

4.1.3 Euler's Method

Euler's method simply computes $x(t_0+h)$ by taking a step in the derivative direction,

$$x(t_0 + h) = x_0 + h\dot{x}(t_0) \tag{4.3}$$

where

x_0 is the initial state (e.g. position vector at time t_0),

and *h* is the step size parameter (e.g. magnitude of the time interval, *dt*).

4.1.4 Linear Velocity

As a rigid body can be represented by a position vector and orientation matrix, the next step is to be able to express these state variables over time. And as mentioned previously, we can define the position derivative using Euler's method. This variable is known as the linear velocity.

$$\dot{x}(t_0 + \Delta t) = v(t_0 + \Delta t) = v(t_0) + \Delta t\dot{v}(t_0) \tag{4.4}$$

where

$v(t_0)$ is the linear velocity vector at time t_0,

$\dot{v}(t_0)$ is the linear velocity vector derivative (acceleration) at time t_0,

and Δt is the time step size for each computational loop.

4.1.5 Angular Velocity

Other than translational motion imparted by the linear velocity, a rigid body can also spin or rotate. Just as the linear velocity describes the position of the rigid body over time, the angular velocity vector, $\omega(t)$, describes the orientation of the rigid body over time. The angular velocity, $\omega(t)$, comes in a form of a vector where its unit direction represents the direction of the axis of rotation, while its magnitude represents the how fast the rigid body is rotating. And the orientation derivative can be computed as follows,

$$\dot{R}(t) = w(t) \times R(t) \tag{4.5}$$

where

$\omega(t)$ is the angular velocity vector at time t,

$R(t)$ is the 3x3 orientation matrix at time t,

And $\dot{R}(t)$ is the derivative of the 3x3 orientation matrix.

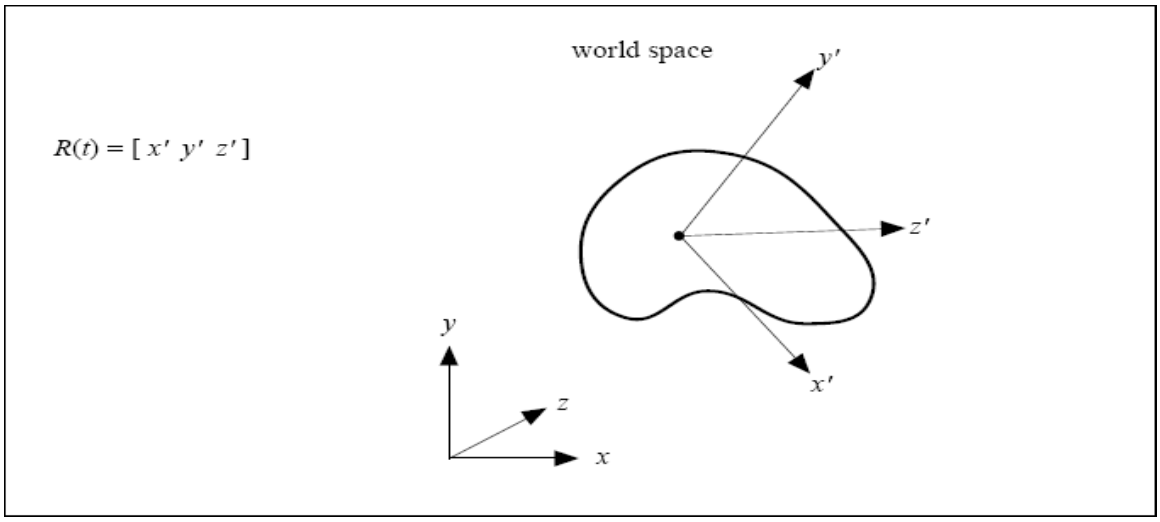


Figure 4.1-2 Physical representation of a Rotation Matrix, R(t) [27]

4.1.6 Particle Velocity

In our representation, a rigid body is represented as a system of constrained particles. So other than the linear velocity, which essentially describes the motion of the center of mass of the rigid body, we also need to find the velocity of individual particles in the system.

$$\dot{p}_i(t) = w(t) \times p_i(t) + v(t) \tag{4.6}$$

where

$\dot{p}_i(t)$ is the derivative of the position vector of the particles of the rigid body at time *t*,

$p_i(t)$ is the reference vector of the particles of the rigid body in body space at time *t*,

$\omega(t)$ is the angular velocity vector of the rigid body at time *t*,

and $v(t)$ is the linear velocity vector of the rigid body at time *t*.

4.1.7 Force and Torque

A rigid body can be represented by a 3x3 orientation matrix and the position vector of its center of mass. When subjected to multiple external moments or forces at different points on the body, the problem can be simplified by reducing these moments and/or forces into a single resultant torque and force at its center of mass. The dynamics of the object can be then computed by simply applying Newton's 2nd Law of motion as shown in sections 4.1.8 and 4.1.9.

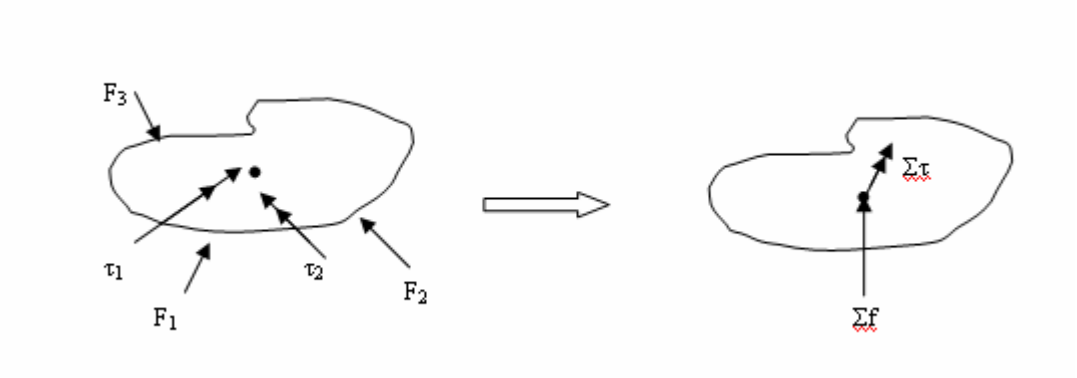


Figure 4.1-3 Representation of Force and Torque on rigid body

The torque $\tau_i(t)$, experienced by the rigid body is the same throughout the particle system, while a force $f_i(t)$, acting on particle p_i, is reduced into a force $f(t)$ acting on the center of mass and a torque $\tau(t)$.

$$f(t) = f_i(t) \tag{4.7}$$

$$\tau(t) = p_i \times f_i(t) \tag{4.8}$$

Hence the total force and torque experienced by the rigid body can be calculated as follows,

$$\Sigma f(t) = \Sigma f_i(t) \tag{4.9}$$

$$\Sigma \tau(t) = \Sigma(p_i \times f_i(t)) + \Sigma \tau_i \tag{4.10}$$

4.1.8 Linear Acceleration

We can see that the linear velocity is influenced by its time derivative, also know as the linear acceleration. And the linear acceleration is directly related to the forces experienced by the rigid body.

$$a(t) = \dot{v}(t) = \frac{\Sigma f_i(t)}{mass} + gravity \tag{4.11}$$

4.1.9 Angular Momentum

Angular velocity is used to describe the orientation of the rigid body over time, and is directly related to the angular momentum. Angular momentum measures the rigid body's tendency to continue rotating. The angular momentum is defined as,

$$L(t) = I(t)\omega(t) \tag{4.12}$$

where

$L(t)$ is the angular momentum vector of the rigid body,

$I(t)$ is a 3x3 matrix for the moment of inertia of the rigid body in world space,

And $\omega(t)$ is the angular velocity vector of the rigid body.

Hence the angular velocity at time t, can be determined as follows,

$$\omega(t) = I^{-1}(t)L(t) \tag{4.13}$$

And using Euler's method, the angular momentum L at time $t_0+\Delta t$, can be determined as follows,

$$L(t_0 + \Delta t) = L(t_0) + \Delta t\dot{L}(t_0) \tag{4.14}$$

where

$L(t_0)$ is the angular momentum vector at time t_0,

$\dot{L}(t_0) = \Sigma\tau(t_0)$ is the derivative of the angular momentum vector at time t_0,

$\Sigma\tau(t_0)$ is the vector representing the total torque experienced by the rigid body,

and Δt is the time step size for each computational loop.

4.1.10 Damping

As we are simulating a real world environment, damping should be implemented. The effect of damping is to bring a moving object to rest in the absence of other influences. A side-benefit is that damping improves numerical stability. In this case, we applied an ideal viscous drag force to damp linear motion of the object as follows.

$$f(t) = -k_d v(t) \tag{4.15}$$

where

$f(t)$ is the linear drag force vector experienced by the rigid body,

k_d is the linear drag coefficient,

and $v(t)$ is the linear velocity vector of the rigid body.

Similarly, an ideal angular drag force is also applied to damp the rotational motion of the object as follows.

$$\tau(t) = -k_\tau \omega(t) \tag{4.16}$$

where

$\tau(t)$ is the angular drag force vector experienced by the rigid body,

k_τ is the angular drag coefficient,

and $\omega(t)$ is the angular velocity vector of the rigid body.

4.2 Collision Detection

Having described the representation of a rigid body in virtual environment, the next problem is to model the interactions between the various virtual objects, namely the micro-needle and the needleholder. However, before we can model the interactions of the virtual

objects, we need to be able to determine if there is any interaction in the first place. Hence, there is a need for real-time interactive collision detection [25].

4.2.1 Sphere-Sphere Collision

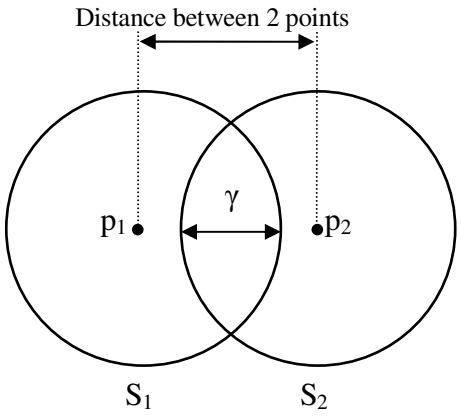


Figure 4.2-1 Sphere-Sphere Collision

The most basic of all collision detection is the sphere-sphere collision. Given a sphere s_1 with a radius r_1, and a sphere s_2 with a radius r_2, we can determine if there will be collision between the 2 spheres, and if so, the penetration value γ.

$$\gamma = (r_1 + r_2) - \sqrt{(p_2 - p_1) \bullet (p_2 - p_1)} \tag{4.17}$$

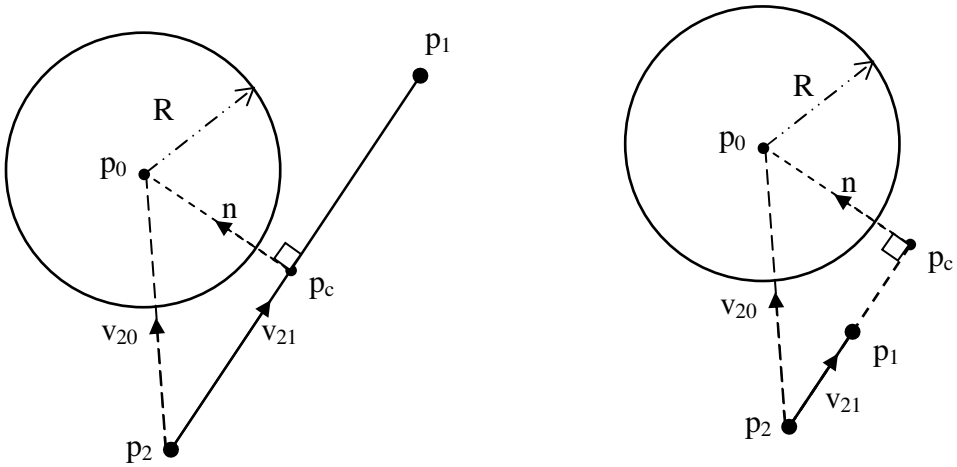
where

γ must be greater than zero, and

p_1 and p_2 are the position vectors of the spheres respectively.

Collision occurs when γ is greater or equal to zero. This collision detection is used for determining if there are collisions between the bounding spheres of the various objects in the virtual environment. It is also used to determine if there are collisions between the idealized point masses of the virtual suture.

4.2.2 Sphere-Edge Collision



(a) Case 1: p_c lies within the line segment (b) Case 2: p_c lies outside the line segment

Figure 4.2-2 Sphere-Edge Collision

The shortest distance between two geometries is usually used to determine if collision occur. The shortest possible distance between a sphere and an infinite straight line is simply the perpendicular distance between the two. As shown in Figure 4.2-2, this is however not always true between a sphere and a line segment/edge as it is possible for p_c the position vector nearest to the sphere, to lie outside of the line segment. And thus the nearest point to the sphere lies on p_1 or p_2 of the line segment.

Hence, the first step in the algorithm is to determine if p_c lies within the edge. As shown in Figure 4.2-2, v_{20} is the vector from p_2 to p_0, and v_{21} is the vector from p_2 to p_1, while n is the normal vector between the two geometries. It can be observed that if $0 < v_{20} \bullet v_{21} < |v_{21}|$ then p_c will lie along the edge. And if $|n| < R$, then collision will occur, where p_c is approximated to be the point of collision. The value of $|n|$ and p_c can be calculated using the following equations.

$$|n| = |v_{20} - (v_{20} \bullet v_{21})v_{21}| \tag{4.18}$$

$$p_c = p_2 + (v_{20} \bullet v_{21})v_{21} \tag{4.19}$$

In contrast, if $v_{20} \bullet v_{21} < 0$, then p_c lies nearer to p_2 and we do a sphere-sphere collision of the sphere with the point p_2 (with radius zero).

Similarly if $v_{20} \bullet v_{21} > |v_{21}|$, then p_c lies nearer to p_1 and we do a sphere-sphere collision of the sphere with the point p_1 (with radius zero).

The main application of this algorithm is to detect the collision between the bounding spheres of the virtual needle and the edges of the virtual needleholder. The discretisation of the virtual needle will be described in Section 4.2.9.

4.2.3 Sphere-Plane Collision

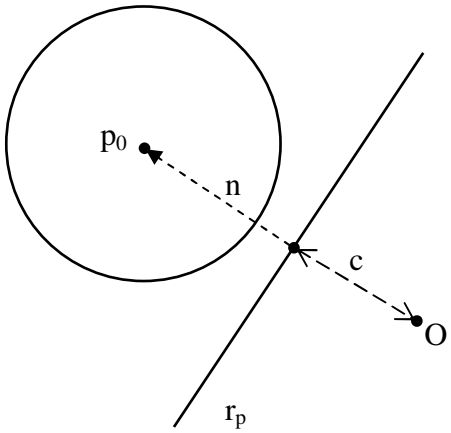


Figure 4.2-3 Sphere-Plane Collision

A plane r_p can always be defined by a unit normal vector n, and a plane constant c.

$$r_p \bullet n = c \tag{4.20}$$

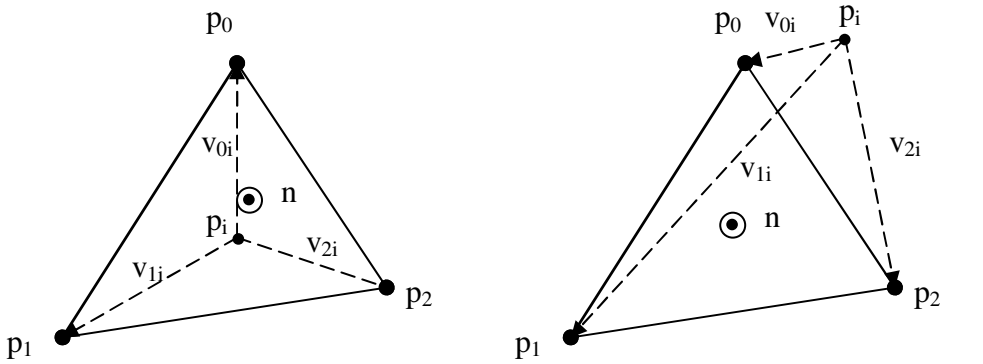
Hence the shortest distance of a point to the plane s,

$$s = p_0 \bullet n - c \tag{4.21}$$

And if $s \leq r_0$, then collision occurs.

The main application of this algorithm is to detect the collision between the bounding spheres of the virtual needle and the planar surfaces of the virtual needleholder. The discretisation of the virtual needle will be described in Section 4.2.9.

4.2.4 Point in Polygon Collision Detection [25]



(a) Case 1: p_i lies inside polygon (b) Case 2: p_i lies outside polygon

Figure 4.2-4 Point-in-Polygon Detection

As seen in Figure 4.2-4 above, it can be seen that if the point *p* lies within the triangle, the cross product between the successive vectors v_{0i}, v_{1i}, v_{2i}, etc … in an anti-clockwise direction will all produce a normal vector in the direction of the plane normal vector *n*, such that,

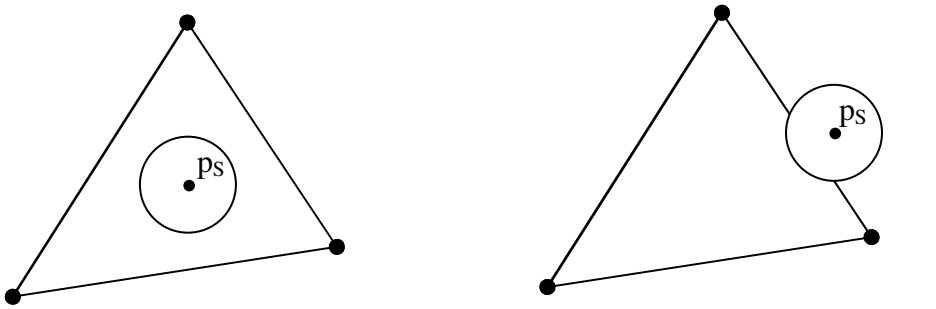
$$(v_{xi} \times v_{x+1i}) \bullet n > 0 \tag{4.22}$$

However, this is not true if the point lies outside the plane, such that at least one of the cross product vectors will be in the opposite direction to the normal vector *n*. For the example shown in Case (b) of Figure 4.2-4, it can be obviously seen that the direction of cross product vector of v_{2i} and v_{0i} is into-the-plane, in contrast to out-of-plane of the polygon's normal vector *n*.

$$(v_{2i} \times v_{oi}) \bullet n < 0 \tag{4.23}$$

In short, if none of the angles between successive vectors are more than 180º then the projection of the point p_i onto the plane of the polygon will lies within the polygon boundaries. Next, the perpendicular distance of the point to the polygon plane is determined (as in Section 4.2.3). And if the value is less than a pre-determined threshold value, the point can be considered to have collided with the polygon plane. The point-in-polygon collision detection is important as few of the virtual surfaces are of the infinite plane variety.

4.2.5 Sphere-Polygon Collision Detection



(a) Case 1: p_s lies inside polygon (b) Case 2: p_s lies outside polygon

Figure 4.2-5 Sphere-Polygon Detection

The actual collision detection for the discretised needle and the jaws of the needleholder are done using the sphere-polygon collision detection, which is actually a combination of the sphere-edge, sphere-plane, and point-in-polygon collision detections. The discretised needle is essentially a collection of bounding spheres (see Section 4.2.9) while the

needleholder is broken down into a collection of polygon planes. Each polygon plane is defined as a planar surface constrained by three or more edges.

As shown in Figure 4.2-5, it is possible for a sphere to touch a polygon plane while its center of mass is outside the polygon. Hence in order to properly perform Sphere-Polygon collision detection, additional checks must be performed to detect the collision between the sphere and edges of the polygon when the sphere center of mass is outside the polygon. Hence the pseudo codes for the Sphere-Polygon collision detection algorithm is as follow:

1. Perform Sphere-Plane check
2. If true, perform Point-in-Polygon check, else no collision
3. If true, collision exists, else perform Sphere-Edge check for all the edges of the polygon
4. If true, collision exists, else no collision

4.2.6 Dynamic Sphere-Polygon Collision Detection

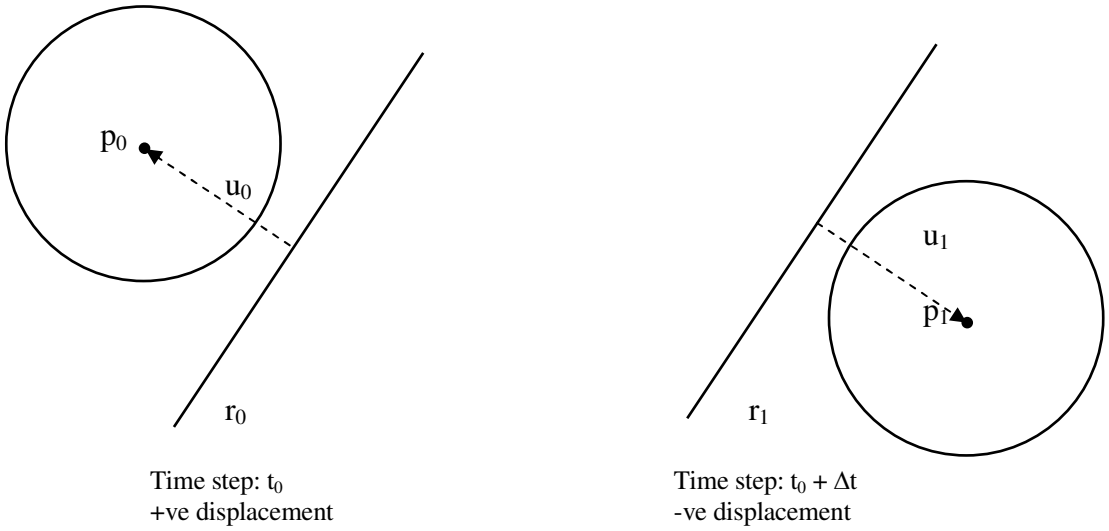


Figure 4.2-6 Dynamic Sphere-Polygon Collision

As the dynamic behaviors of virtual objects are calculated in discrete time steps, the main problem of the sphere-polygon collision detection algorithm is that if the velocity of the sphere is very high, the sphere might pass through the plane without getting close to the plane (as the distance moved by the sphere is very large in a single time step). Hence, this collision detection uses a through ray algorithm instead.

As shown in Figure 4.2-6, if the sphere is within the polygon and the displacement of the sphere to plane is positive for the previous time step, and the same sphere is still within the polygon and the displacement is now instead negative for the current time step, then collision can potentially occur. And the time of collision t_c, can be approximate to

$$t_c = t_0 + \frac{|u_0|}{|u_0| + |u_1|}\Delta t \tag{4.24}$$

where

t_c is the approximated time of collision,

t_0 is the time of collision before collision,

Δt is the discrete time step,

If the approximated time of collision is known, the approximated position and orientation of the polygon plane can be extrapolated, and the appropriate approximations for the Sphere-Polygon collision response can be then performed.

4.2.7 Line-Line Collision Detection [22]

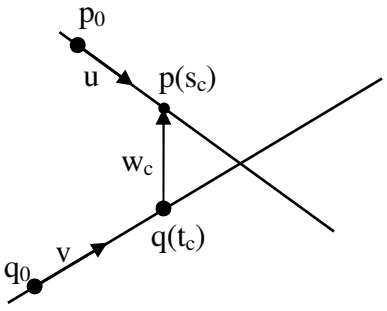


Figure 4.2-7 Line-Line Collision

Given two infinite lines L_1 and L_2,

$$L_1 : p(s) = p_0 + su \tag{4.25}$$

$$L_2 : q(t) = q_0 + tv \tag{4.26}$$

where

p_0 and q_0 are some arbitrary position vectors,

u and v are some arbitrary directional vectors,

and s and t are some arbitrary constants

Defining s_c and t_c as some unique arbitrary values for the constants s and t such that the points $p(s_c)$ and $q(t_c)$ are closest to each other, it can be observed that

$$w_c = p(s_c) - q(t_c) = w_0 + s_c\, u - t_c\, v \tag{4.27}$$

where

$\mathbf{w}_0 = p_0 - q_0$,

And if the two lines are not parallel, then the vector w_c is uniquely perpendicular to both the lines such that,

$$u \bullet w_c = 0 \tag{4.28}$$

$$v \bullet w_c = 0 \tag{4.29}$$

Hence, we can derive the following equations,

$$(u \bullet u)s_c - (u \bullet v)t_c = -u \bullet w_0 \tag{4.30}$$

$$(v \bullet u)s_c - (v \bullet v)t_c = -v \bullet w_0 \tag{4.31}$$

Solving the above equations, we can obtain the unique points $p(s_c)$ and $q(t_c)$ where

$$s_c = \frac{(u \bullet v)(v \bullet w_0) - (v \bullet v)(u \bullet w_0)}{(u \bullet u)(v \bullet v) - (u \bullet v)(u \bullet v)} \tag{4.32}$$

$$t_c = \frac{(u \bullet u)(v \bullet w_0) - (u \bullet v)(u \bullet w_0)}{(u \bullet u)(v \bullet v) - (u \bullet v)(u \bullet v)} \tag{4.33}$$

Hence, we can obtain $|w_c|$. And if we assume the two lines are actually cylinders with radius r_1 and r_2. Then collision occurs between the two cylinders if $|w_c| < r_1 + r_2$.

However, this is not a very useful algorithm as infinite cylinders do not actually exist.

4.2.8 Edge-Edge Collision Detection [22]

We can define the edges E_1 and E_2 as

$$E_1 : p(s) = p_0 + su \quad \text{where } 0 < s < 1 \tag{4.34}$$

$$E_2 : q(t) = q_0 + tv \quad \text{where } 0 < t < 1 \tag{4.35}$$

where

p_0 and q_0 are some arbitrary position vectors,

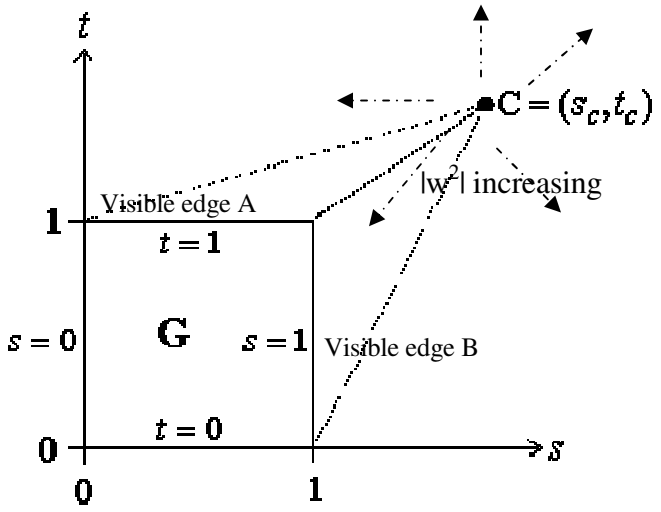
u and v are some arbitrary directional vectors (not unit vectors),

$|u|$ and $|v|$ are the lengths of E_1 and E_2 respectively,

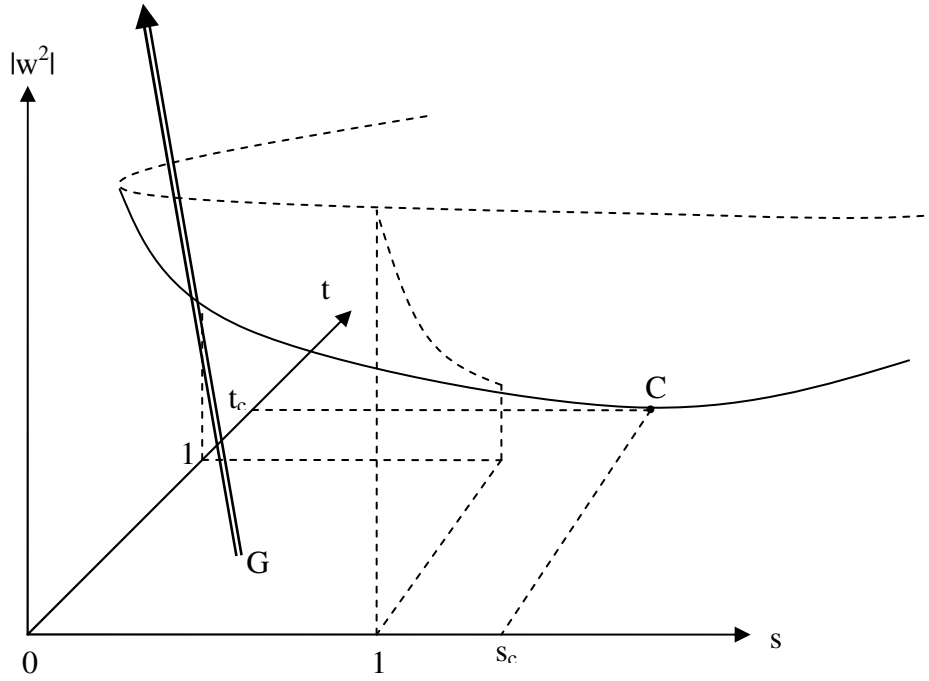
and s and t are some arbitrary constants with values ranging from 0 to 1

Similar to Section 4.2.7, the two mutually closest points $p(s_c)$ and $q(t_c)$ in E_1 and E_2 respectively can be calculated. However unlike collision between two infinite lines, it is possible for the points $p(s_c)$ or/and $q(t_c)$ to lie(s) outside their respective line segment, resulting in that the shortest perpendicular distance between the two line segments to be impossible.

Hence, instead we need to perform a minimization problem on $|w|^2$, which is a quadratic function of s and t. This minimization can be represented as a parabaloid over the (s,t)-plane as shown in Figure 4.2-8, with the global minimum at $w(s_c,t_c)$ computed previously. The parabaloid is a monotonic increasing function with a global minimum C (found previously) which lies outside the region G (where the edges E_1 and E_2 are defined). Hence in order to determine if there are any contacts between the two edges, the local minimum in the region G must be found.



(a) $|w^2|$ Projection the (s,t)-plane



(b) Parabaloid over the (s,t)-plane

Figure 4.2-8 Parabaloid over the (s,t)-plane

This local minimum can be determined by observing the visible edges of the boundary of region G. This is because the parabaloid is monotonic increasing, thus the local minimum

for region G should lie on the boundary edges (e.g., Visible edge A and B). And the local minimum for each visible edge can be calculated as follows

$$\frac{d}{dt}\left|w(t,s=0)_l\right| = 0 \ , \ \frac{d}{ds}\left|w(t=0,s)_l\right| = 0 \tag{4.36}$$

And by comparing the absolute magnitudes of the local minimums of each edges, we can determine the local minimum for the region G. And , which is the vector describing the shortest distance between the two edges. Similarly, both the edges can be represented as cylinders with radius r_1 and r_2. And collision occurs if $\left|w\right| < r_1 + r_2$. This collision detection is used to determine the interaction of a cylinder with the needleholder's jaws.

4.2.9 Bounding Volume Hierarchy (BVH)

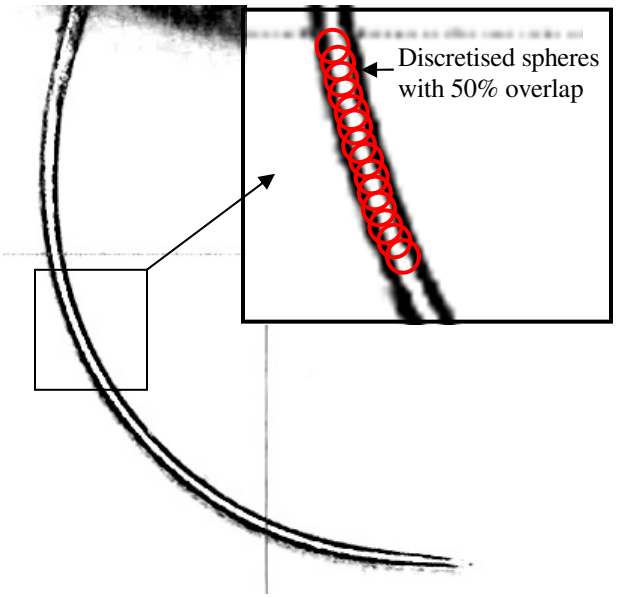


Figure 4.2-9 Discretisation of micro-needle

As it is very difficult to represent a 3D virtual needle as a continuous equation, the virtual needle is instead discretised into a string of overlapping spherical point masses as shown in Figure 4.2-9. The diameter of each sphere is of the exact same value as the corresponding point on the virtual needle, and each consecutive sphere are placed closely together (50% overlap) to create a smoother surface. Hence in order to perform complete collision detection, each and every point masses of the virtual needle need to undergo at least one collision detection check. This can prove to be very computationally expensive [19]. For example, the virtual jaws of the needleholder has 8 planes and 12 edges, and assuming the needle is discretised into 100 point masses, in a single loop will requires at least 2000 checks. And each collision detection check typical has more than 5 operations. Hence the number of mathematical operations required just for collision detection between the needleholder and needle would exceed tens of thousand. However, with the implementation of a Bounding Volume Hierarchy (BVH), collision detection checks can be significantly reduced.

A BVH is simply a tree of bounding volumes. The bounding volume at a given node encloses the bounding volumes of its children. The bounding volume at the lowest level in the tree usually encloses one or more geometric primitives. In this project, the BVH used is the Bounding Sphere Hierarchy (BSH). A bounding sphere is used to enclose the entire virtual needle. This bounding sphere is placed at the topmost level of the BSH tree, and a branch is allocated to the bounding sphere. The virtual needle is then broken up into segments and placed into two or more smaller bounding spheres. These bounding spheres are then placed on the branch of the topmost bounding sphere. And each virtual needle

segments in each leaf (bounding spheres) is further broken up and placed in successively smaller leaves. These smaller leaves are then placed on the branches of their precedent leaves. This is essentially repeated until a specified number of levels are reached or when the virtual needle cannot be further broken up. For example, in Figure 4.2-10, a 4-level BSH tree is constructed for the virtual needle. The top level of the tree comprises of two level 2 leaves, while each level 2 leaf comprises of two level 3 leaves, and each level 3 leaf contains two level 4 leaves. Finally, the lowermost leaf will contain a collection of point masses of the virtual needle.

As mentioned previously, the advantage of using the BSH is that the computation of collision detection can be significantly reduced. This is because if the initial collision detection check failed against the bounding sphere of a particular node, then there will not be any collision between the jaws and the children of the node, hence eliminated the need to perform further checks with the children.

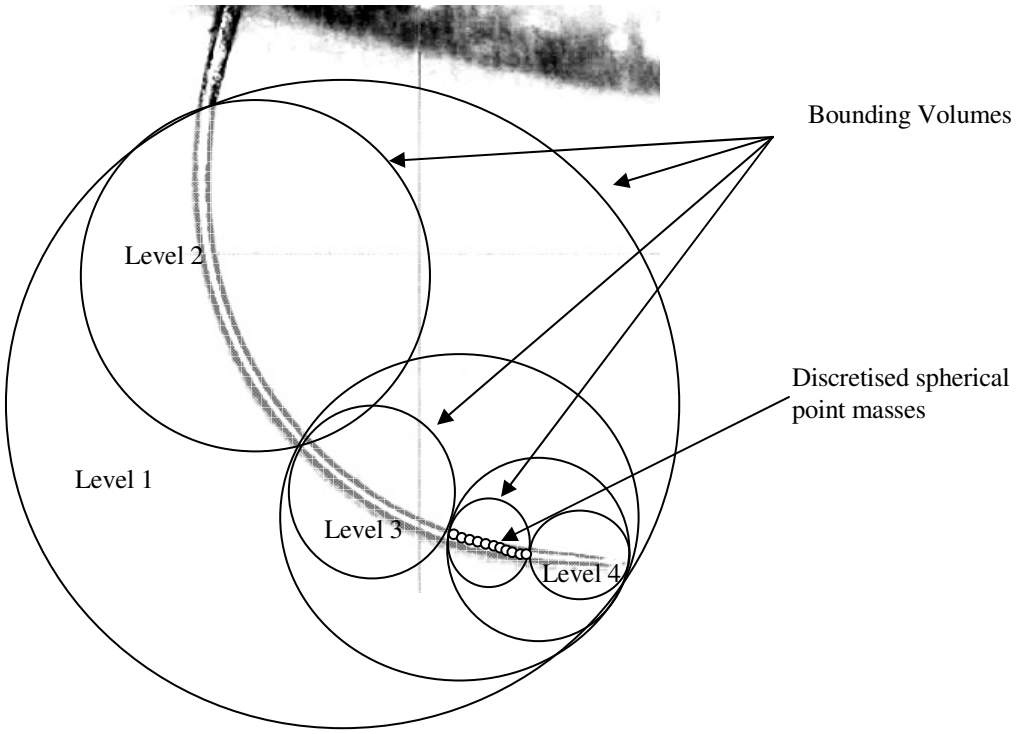


Figure 4.2-10 Four-Levels BSH on micro-Needle

4.2.10 Constructing a Bounding Sphere Hierarchy (BSH)

Given a system of particles, we can construct a BSH as shown in Figure 4.2-11, by first finding the center of mass of the system, then computing the minimum possible radius required to enclose the entire system. This resultant bounding sphere is the level 1 or parent node of the sphere tree.

Next, the particle that is furthest from the parent node is chosen as the starting point to grow the level 2 sphere leaf. All the nearby particles are tested to see if they are within the bounding sphere (with half the radius of its parent as the initial threshold value) surrounding the particle. The new local center of mass is computed and the sphere leaf

shifted to this new position. This is repeated until no new particles can be added into the leaf. After which the optimal (minimum) radius of the leaf is then calculated. There will be particles that are not enclosed by the 1st sphere leaf because of the initial threshold radius. Hence, an additional level 2 sphere leaf is generated in the similar manner. This process is repeated until all the particles are enclosed in level 2 sphere leaves.

One all the level 2 sphere leaves are generated, level 3 sphere leaves will be generated in each of the level 2 sphere leaves in the similar manner. This process is repeated for progressive levels until any one of the conditions is fulfilled,

- Only a single particle exists in the sphere leaf,
- The number of particles in the sphere leaf is less than an arbitrary value specified by the user
- The level of the sphere leaf has reached an arbitrary value specified by the user

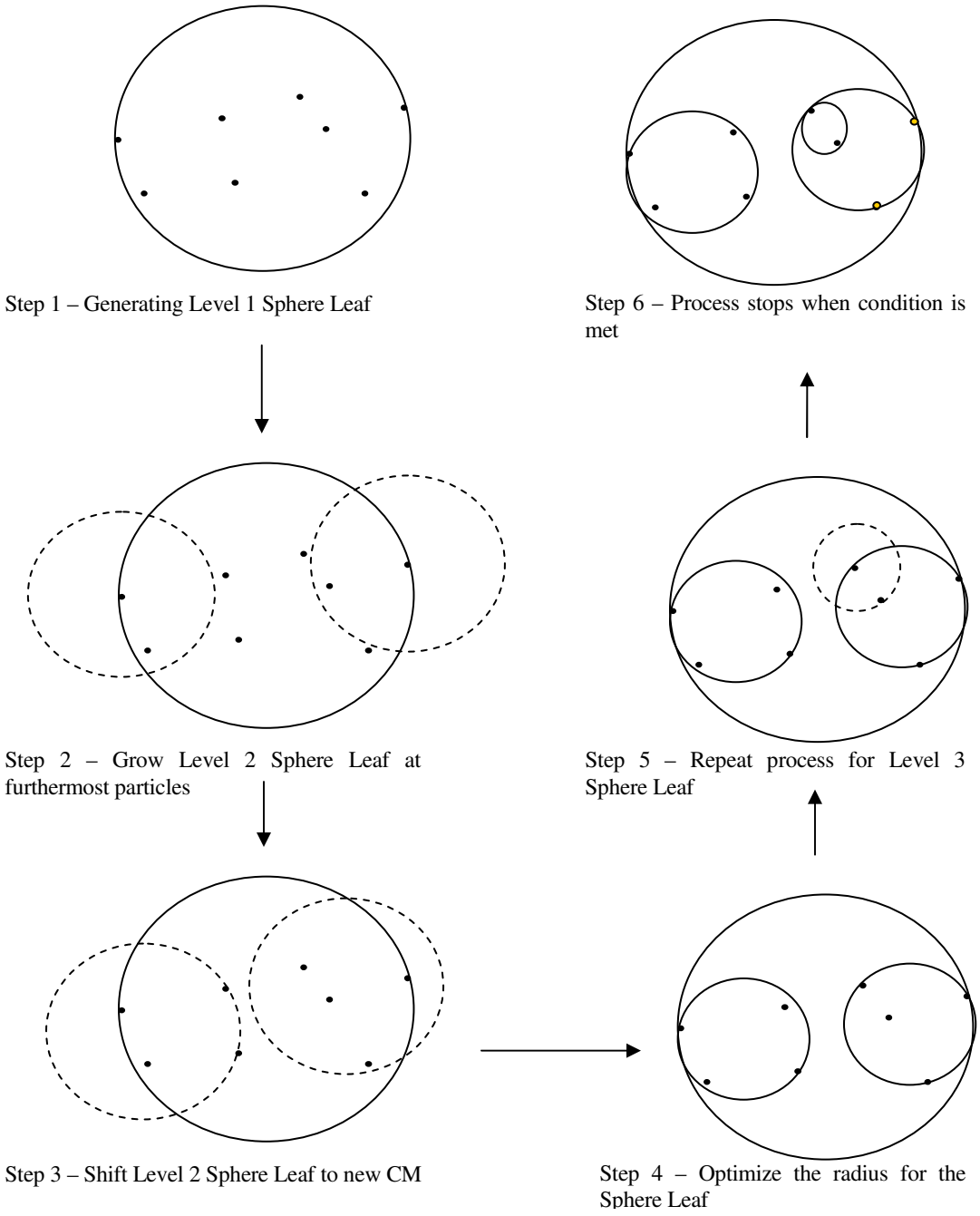
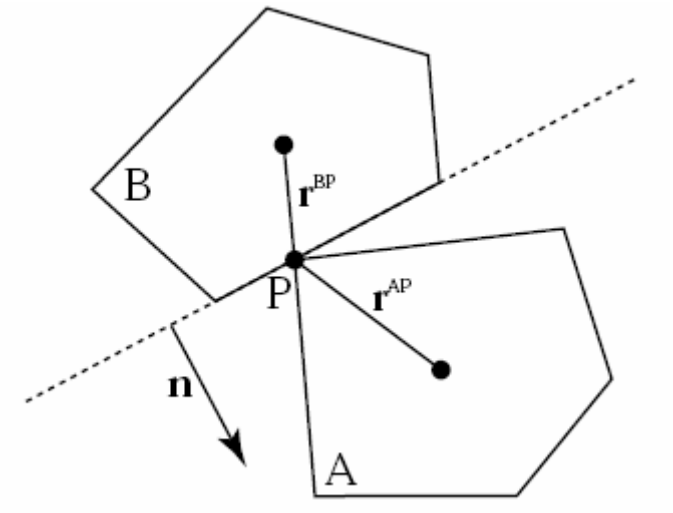


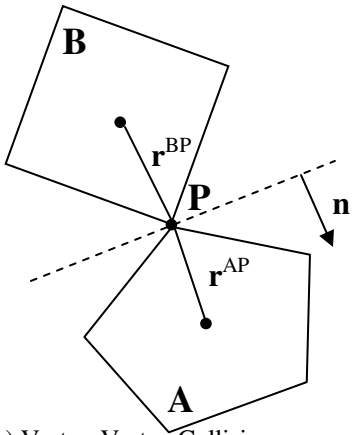
Figure 4.2-11 Constructing a Bounding Sphere Tree

4.3 Collision Response

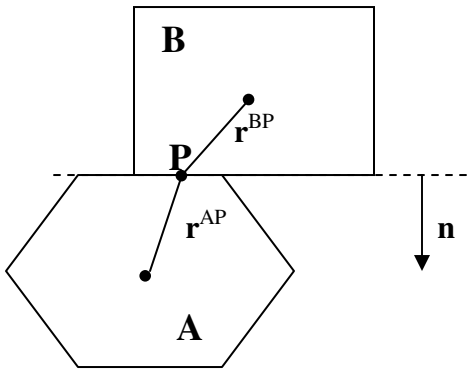
Having determined the interactions between the virtual objects, we now have to model the interaction behaviors of these objects. This is also known as the collision response. The main importance of collision response is to prevent penetration between objects, and impulsive behaviors are used to model these behaviors [20, 27].



(a) Vertex-Plane Collision



(b) Vertex-Vertex Collision



(c) Plane-Plane Collision

Figure 4.3-1 Collision between two objects

As shown in Figure 4.3-1, there are numerous ways for two objects to collide with each other. And in of the cases, by Newton's Law of Restitution, we can write out

$$(v_2^A - v_2^B) \bullet n = -e(v_1^A - v_1^B) \bullet n \tag{4.37}$$

$$v_2^{AB} \bullet n = -e\, v_1^{AB} \bullet n \tag{4.38}$$

where

v_1^A is the initial velocity of the center of mass of A,

v_2^A is the resultant velocity of the center of mass of A,

v_1^B is the initial velocity of the center of mass of B,

v_2^B is the resultant velocity of the center of mass of B,

v_1^{AB} is the initial relative velocity between the center of mass of A and B,

v_2^{AB} is the resultant relative velocity between the center of mass of A and B,

e is the coefficient of restitution,

and n the normal vector of the collision.

From Figure 4.3-1(a) & (c), it is obvious the normal vector n is parallel to the normal of the colliding planes. However for case (b), there is no clear collision vector, and in the case of

this project, the collision vector *n* is taken to be parallel to the relative velocity between the center of mass of the two objects. This is assumed for the cases of vertex-vertex, vertex-edge, edge-edge collisions.

As the objects are assumed to be rigid bodies, we can obtain

$$v_2^{AP} = v_2^A + \omega_2^A \times r^{AP} \tag{4.39}$$

$$v_2^{BP} = v_2^B + \omega_2^B \times r^{BP} \tag{4.40}$$

where

v_2^{AP}, v_2^{BP} are the resultant velocity of the respective objects at the collision point P (e.g. velocity of point P on Object A after collision occurs),

v_2^A, v_2^B are the result velocity of the respective objects at the center of mass,

ω_2^A, ω_2^B are the resultant angular velocity of the respective objects,

and r_{AP}, r_{BP} are the position vector of the collision point.

We also know that impulse is essentially the force experienced by the objects over a time Δt, hence we have the following equations

$$v_2^A = v_1^A + \frac{j}{m_A},\ \omega_2^A = \omega_1^A + I_A^{-1}(r_{AP} \times j) \tag{4.41}$$

$$v_2^B = v_1^B - \frac{j}{m_B},\ \omega_2^B = \omega_1^B + I_B^{-1}(r_{BP} \times j) \tag{4.42}$$

where

v_2^A, v_2^B are the resultant velocity of the respective objects,

v_1^A, v_1^B are the initial velocity of the respective objects,

ω_2^A, ω_2^B are the resultant angular velocity of the respective objects,

ω_1^A, ω_1^B are the initial angular velocity of the respective objects,

j is the impulse experienced by the colliding objects,

I_A^{-1}, I_B^{-1} is the 3x3 inverse matrix of the moment of inertia about the center of mass of the respective objects in world space,

and m_A, m_B are the mass of the respective objects.

Hence, by substituting all the above equations into Newton's Law of Restitution, we can obtain an expression for the Impulse j,

$$j = \frac{-(1+e)v_1^{AB} \bullet n}{n \bullet n(\frac{1}{m_A} + \frac{1}{m_B}) + [(I_A^{-1}(r_{AP} \times n)) \times r_{AP} + (I_B^{-1}(r_{BP} \times n)) \times r_{BP}] \bullet n} \tag{4.43}$$

Once the impulse *j*, is computed, then impulsive forces can be applied on both bodies, to determine the resultant linear velocities and angular velocities.

5 Implementations

The algorithms described in previous chapters were implemented into a library of APIs. In this chapter, the capability of the C++ static library developed is described. In addition, the implementation of various unique behaviors observable in microsurgery is also described.

5.1 Physics Engine

A physics engine is essentially a library of APIs to implement and co-ordinate the dynamic behaviors of various virtual representations of objects found in microsurgery. In this project, the physic engine was written into a form of a C++ static library. A static library is essentially just a collection of object files that are linked into the program during the linking phase of compilation, and is not relevant during runtime. In this form, it is easy and convenient for other software engineers to develop their own virtual environments by linking to the static library. In addition, the C++ Standard Template Library (STL), which is a general-purpose C++ library of algorithms and data structures, was used to develop a customized STL container for the purpose of holding the various data required for the physics calculation and implementation. The main input device for the system in this project is the Phantom^{tm} Desktop, and it is provided with the GHOST® (General Haptic Open Software Toolkit) SDK. The GHOST® SDK is a powerful C++ tool kit used to develop virtual and haptic environments. It also possesses the drivers and API to communicate with the haptic device. Hence, the physics engine was developed such that it is complementary with the GHOST® SDK.

The main concerns in developing a VR-based system are the realism of the environment and the computational expenses required. Various literatures on simulating dynamics of rigid bodies were studied [20,27] and implemented. Virtual objects like the micro-needle and needleholder were assumed to be rigid bodies. And by representing a rigid body as a system of particles, the virtual dynamics/behaviors of the virtual objects can be simulated using particle mechanics. In addition, interactions that will occur in microsurgery can be simulated by implementing collision detection and responses. Various algorithms for collision detections were studied and implemented [22,25,26,27]. An architecture was developed to co-ordinate and simulate the dynamic behavior between the various virtual objects in the VR based system.

The main reason for concern in computation expenses is because typical VR software runs in two loops, the graphics loop and the haptic loop, and in order to have a realistic haptic feel, it is a requirement that the haptic loop runs at 1 kHz or higher. This is equivalent of having a maximum computational time of 1 ms per loop. As computational time depends on the number of mathematical operations, it is essential that the number of operations in a single loop be kept to the minimum.

In the course of developing the engine, it was found that collision detections between objects took up most of the computational time, such that it took more than 1 ms (requirement for haptic interface) to check if there was collision between the needleholder and the needle. And it was found that a boundary sphere hierarchy can greatly reduce the computational time, by optimizing the amount of collision detection required [19].

5.2 Grasping and Needle Insertion

The main objective of the static library is to model the required behavior for the grasping of the micro-needle and the insertion of the micro-needle into the tissue. Hence, other than the generic collision between the needleholder and the micro-needle, customized functions are also written to simulate some of the special events in grasping and insertion.

5.2.1 Collision Detection between Needleholder and Needle

The needleholder is primarily modeled as a pair of customizable triangular jaws and the needle a set of particle with radius. The initial collision detection check is the bounding sphere of the jaws against the BSH of the needle. All probable collision particles are stored in a customized STL container, where dynamic sphere-polygon collision checks are carried out. And if all the grasping conditions are fulfilled, no collision responses are implemented and the collision detection routine ends.

The conditions for a successful grasp are

- There exists a left penetration and right penetration on a single particle
- The left penetration and right penetration of the jaws on the needle should not differ by more than 10%
- The particle with the greatest overall penetration is chosen as the grasped point

However, if no successful grasping occurs, the particle with the greatest penetration (earliest instant of collision) for each of the planes of the jaws, is selected as the colliding particle. In each case, collision response as mentioned in the previous section is implemented for each particle.

5.2.2 Realignment of needle during Grasping

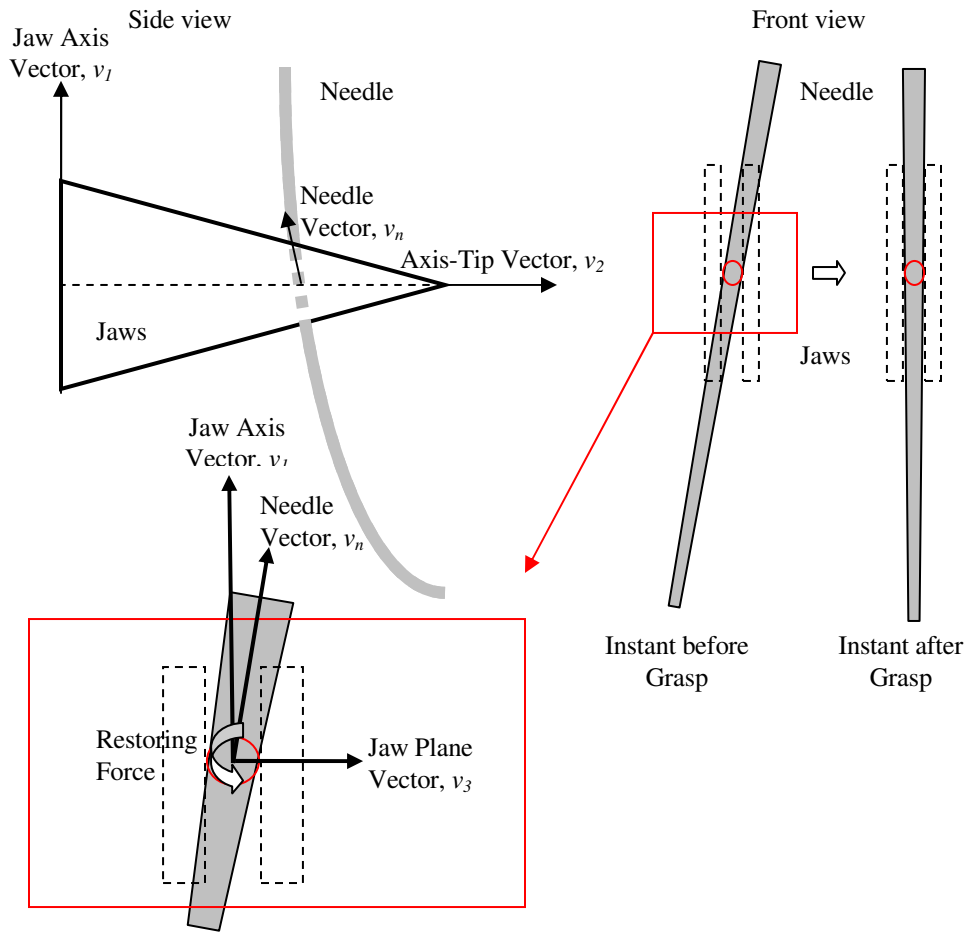
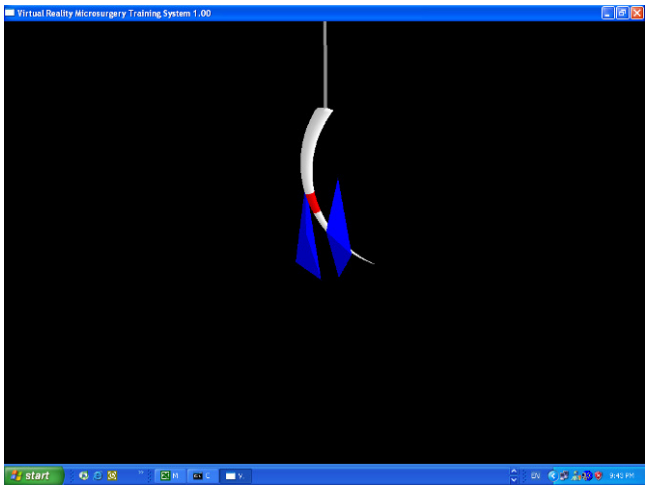


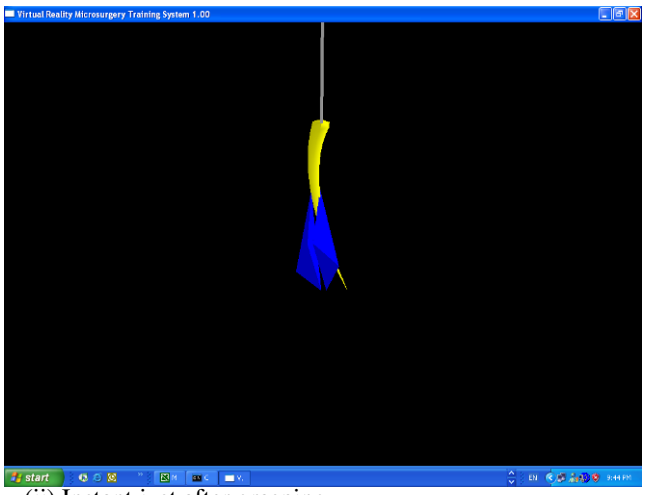
Figure 5.2-1 Twisting during Grasping

As shown in Figure 5.2-1, it is possible for a needle to be orientated differently from the jaws when grasped. This results in a snapping or realignment of the needle. In order to

simulate this effect, the differences in orientation have to be computed. The unit vectors of Jaw-Axis, Axis-Tip, and Needle vectors, v_1, v_2, v_3, and v_n respectively are retrieved from the physics engine. The vector component parallel to v_2, in v_n is removed such that $v_n = v_n - (v_n \bullet v_2)v_2$. This is because rotation in this direction is constrained and physically impossible (if the grip is firm). And by performing dot products $v_n \bullet v_1 = \cos\theta_1$ and $v_n \bullet v_3 = \cos\theta_3$, the differences in orientation with respect to v_1 and v_2 can be calculated and a restoring force implemented to simulate the snapping effect when the jaws closes on the needle. Figure 5.2-2 illustrates the realignment of the needle in the demo program.



(i) Instant just before grasping



(ii) Instant just after grasping

Figure 5.2-2 Realignment of needle during grasping

5.2.3 Twisting of needle during Insertion

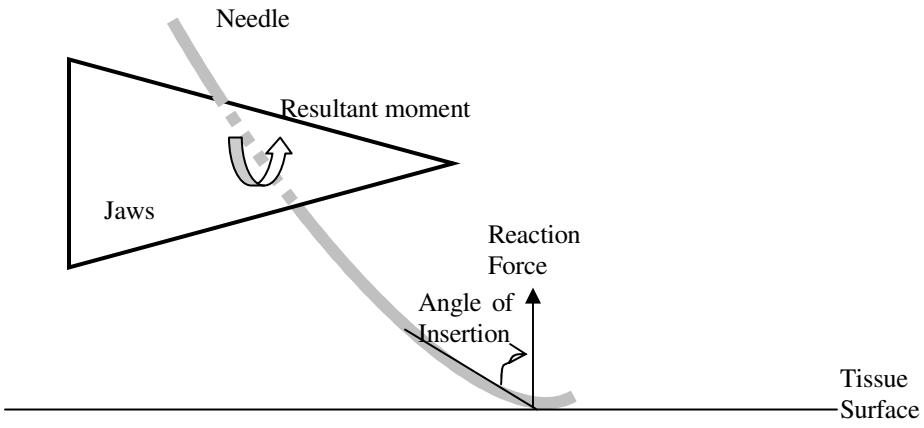
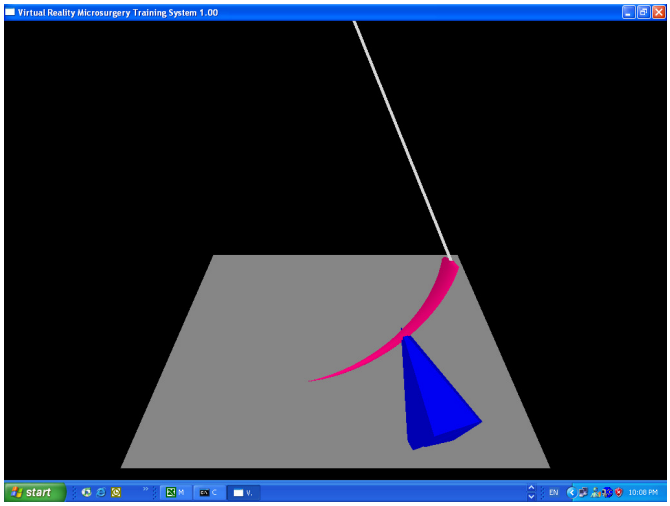


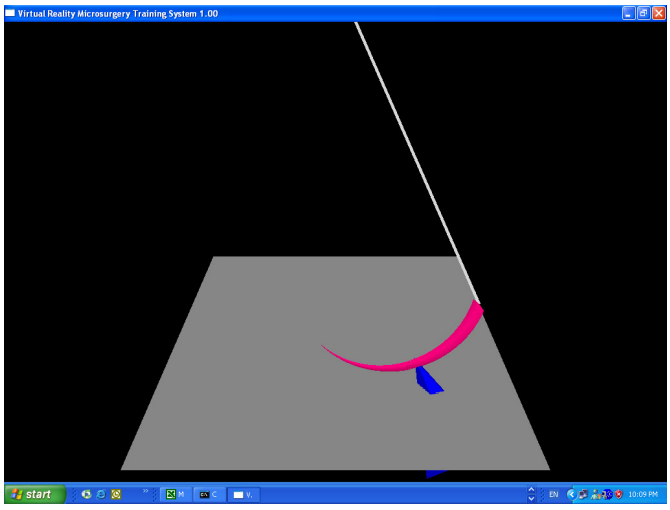
Figure 5.2-3 Reaction force on needle during Insertion

As shown in Figure 5.2-3, when the angle of insertion is large, a significant resistive force will be exerted on the needle by the tissue surface. And if the needle is not grasped sufficiently firmly, the moment which results will twist the needle out of alignment as

demonstrated in the demo program in Figure 5.2-4. This phenomenon is simulated by applying a reaction force on the needle about the grasped point. A constraining force is also applied to prevent the motion of the grasped point. This constraining force can be varied depending on the pressure applied on the needleholder (which is provided by the DVRT mounted on the needleholder).



(i) Instant before contact with surface



(ii) Instant just after contact with surface

Figure 5.2-4 Twisting of needle during Insertion

5.2.4 Force feedbacks during Grasping and Insertion

The physics engine computes the impulsive forces experienced by the needleholder due to collision with other virtual objects using Equation (4.42). These forces are then transmitted to the haptic device after a scaling. The scaling constant is a variable specified by the user. This is because in reality, the forces experienced in microsurgery are minimal and can be generally ignored. However, it may be useful to scale up the force feedbacks to create a pseudo-realistic learning environment (e.g. a more sensitive environment).

As shown in Figure 5.2-3, a reaction force is exerted on the needle during insertion. The magnitude of this reaction force is set to be approximately proportional to the angle of insertion as shown in Figure 5.2-5. A threshold value is specified by the user at which the reaction force is render null to simulate a smooth and easy insertion. This threshold value represents the correct range of values for the angle of insertion, for a proper insertion. It is to be noted that different relationships between the reaction force and the angle of insertion can also be implemented to study their effects on learning. This force experienced by the needle is approximated onto the needleholder, which is then scaled and transmitted to the haptic device.

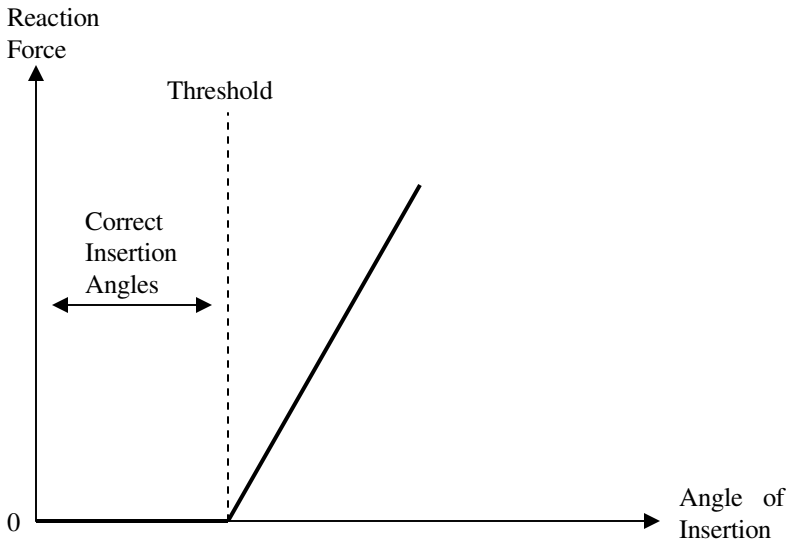


Figure 5.2-5 Reaction force vs. Angle of Insertion

5.3 Simulation Architecture

A typical update loop for the simulation would be as follows,

- Perform BSH collision detection between all the virtual objects to obtain objects that are in close proximity
- Perform detailed collision detection between these viable objects
- Implement collision responses to the colliding objects
- Update all the virtual bodies
    - Compute the total forces and torque experienced by the body
    - Apply damping to the body
    - Compute the linear acceleration of the body

    - Compute the linear velocity of the body
    - Update the position of the center of mass
    - Compute the angular momentum
    - Compute the angular velocity
    - Update orientation of the body
    - Compute the position vectors for all the particles in the system in world space
- Next loop

5.4 Library Architecture

A static library in C++ was developed for the VR-based trainer. The main function of the static library is to provide a collection of APIs to develop the individual modules for the VR-based trainer. Essentially, the main object classes that is required to develop a customized virtual environment, are as follow

- A Collider class which handles the interactions between the various virtual objects
- A Needle object class which represents the micro-needle in the VR environment
- A Suture object class which represents the micro-suture in the VR environment
- A Jaws object class which represents the needleholder in the VR environment
- A Surface object class which represents the soft tissue for suturing in VR environment

The typical procedure in modeling a module is as follows,

- Declare a Collider object
- Use the various API in Collider to generate the various virtual objects
- Insert the update function call at the haptic loop
- Insert the draw function call at the graphic loop

Collider Class - The collider class is the wrapper class to generate the various virtual objects and co-coordinating the collision detection and responses between various objects. It is also responsible for the graphic rendering of the objects and environment.

Jaws Class - The jaws class is the representation of the virtual needleholder. It is essentially compose of a list of dynamic planes and edges. Most of the collision detection are implemented in this class.

Needle Class - The needle class is the representation of the virtual needle. It is a child of the rigid body class. It also has a sphere tree.

Suture Class - The suture class is the representation of the virtual suture. It is compose of a list of point masses and spring. It also has a sphere tree and self collision is implemented.

Surface Class - The surface class is the representation of the virtual tissue. It is compose of a plane and the collision detection and response with the needle is implemented here.

Dynamic Plane Class - The dynamic plane class is essentially a container to hold two instants of a plane at any one time. It is required for the collision detection functions in the jaws class.

Plane Class - The plane class is just the basic representation of a plane.

Rigid Body Class - The rigid body class is the parent class for most of the virtual objects. The generic dynamic behaviors are implemented in this class.

6 Results and Analysis

In this chapter, the performance of the physic engine was tested and verified to be satisfactory. The effectiveness of the BSH algorithm was also studied and presented. As part of the project, a series of training modules were developed to train the subtask of grasping and inserting a micro-needle. In this chapter, the scenario and requirements for each module is presented, and the procedure to evaluate the performance of the student is also presented. A trial run with the software is also being described.

6.1 Performance of Physics Engine

As the computational ability of the workstation is finite, it is very important that the number of mathematical operations is kept at the minimum, while maintaining the high realism required for good virtual learning. In the following sections, tests were conducted to investigate the effectiveness of a BSH, and to verify that the physics engine is able to complete their calculations within a haptic loop (<1ms).

6.1.1 Bounding Sphere Hierarchy (BSH)

As shown in Table 6-1, it can be seen that the addition of a sphere tree, literally cuts down the mean collision computational time by 50%. Although higher depth sphere tree generally decreases the computational time, but it depends greatly on the type of interaction, as a greater depth signifies more checks but also more elimination.

Table 6-1 Computational time for Sphere Trees (sample size 10,000)

Sphere Tree	**0 Level**	**3 Levels**	**4 Levels**	**5 Levels**
Generation (ms)	NA	0.18609	0.27188	0.29920
Collision Detection (ms)	0.73~1.1	0.15~0.93	0.16~0.94	0.15~0.63
Mean Collision Detection (ms)	0.92713	0.38867	0.49025	0.29138

6.1.2 Computational Time per Haptic Loop

As the VR-based trainer is integrated with a haptic device, there is a requirement that the simulation is at least 1 ms, as realistic forces require at least a frequency of 1 kHz. As shown in Table 6-2, the computational time for simulating a virtual dynamical environment is only 0.61344 ms. This verify the capability of the physic engine.

Table 6-2 Mean computational time for 1 loop (sample size 10,000)

Type	**Computational time per loop (ms)**
Virtual Dynamic	0.61344
Graphic Rendering	2.86280

All the above tests are done on an Intel® Pentium® 4 CPU 2.6 GHz, with 1 GB RAM.

6.2 VR Training Software

The main objective of this project is to develop VR training software to teach the subtask of grasping and inserting a micro-needle. The following sections describe the various modules

that were developed (using the physics engine developed in Chapter 5) to train the subtasks. The entire training simulation is developed for both stereoscopic and non-stereoscopic display.

6.2.1 Grasping of a Needle

A demonstration module as shown in Figure 6.2-1 was developed to help the student familiarize himself with the virtual environment of the trainer. Two virtual spheres and a virtual cylinder were presented for practice for manipulating or grasping.

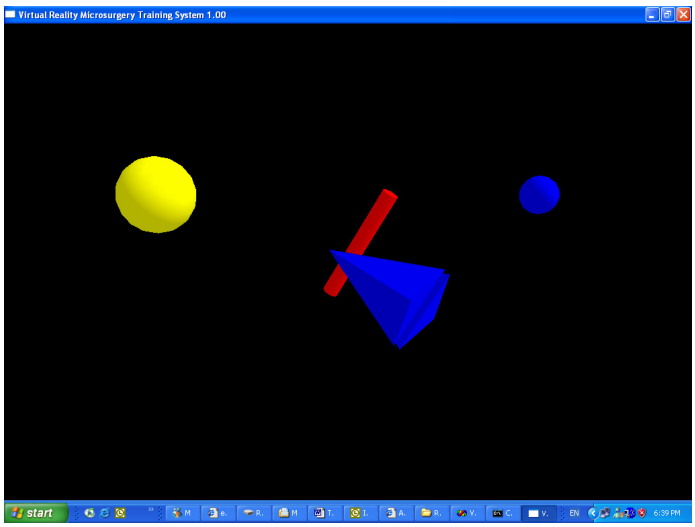


Figure 6.2-1 Demonstration Module

6.2.2 Basic Grasping Module

A Basic Grasping Module as shown in Figure 6.2-2 was developed as the next module of the grasping series. In this module, the key learning objective is to grasp the simplified needle at the correct location. The desired grasp location is customizable and is displayed

as the red region on the simplified model of the micro-needle. The main purpose of using a simple geometry in this module is to slowly introduce the student to the concept of grasping in virtual reality.

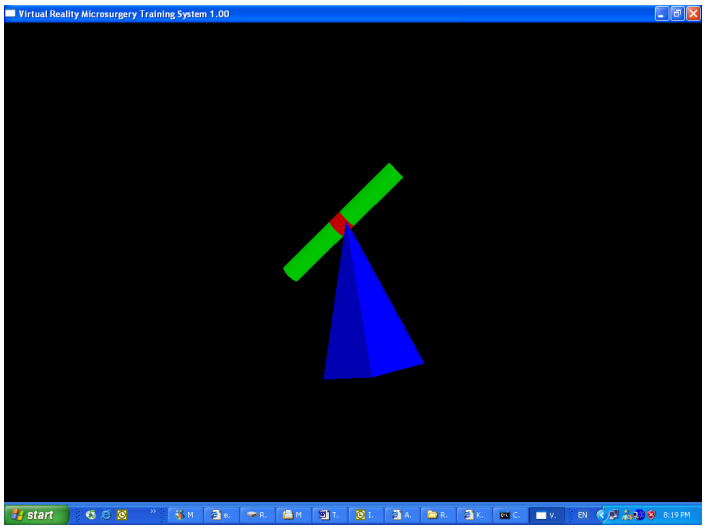


Figure 6.2-2 Basic Grasping of a simplified needle

6.2.3 Advanced Grasping Module I

The next module in the grasping series is the Advanced Grasping Module I as shown in Figure 6.2-3, Figure 6.2-4, Figure 6.2-5 and Figure 6.2-6. The purpose of this module is to train the student's coordinate such that he is able to accurately maneuver the needleholder to any desired position and grasp properly. A realistic needle is presented and the student is required to practice grasping the needle at the highlighted location. Three different types of micro-needles are available for practice. And as the basic module, the desired grasping location is customizable by the instructor.

However, unlike the basic module, the dynamics of the needle model is vastly different. Due to the curved geometric of the suturing needle, rotation of the needle is a problem. Moreover, it is desirable that the needle is grasped vertically and longitudinally perpendicular to the needle holder. And to prevent tissue damage, the needle should be grasped only using the tips of the needleholder.

Visual color cues are displayed when the needle is being successfully grasped.

- A red needle represents the grasp point is wrong and the student is not grasping with the jaws tips.
- A pink needle represents the grasp point is wrong but the student is correctly grasping with the jaws tips.
- A yellow needle represents the grasp point is correct but the student is not grasping with the jaws tips.
- A green needle represents the grasp point is correct and the student is grasping with the jaws tips.

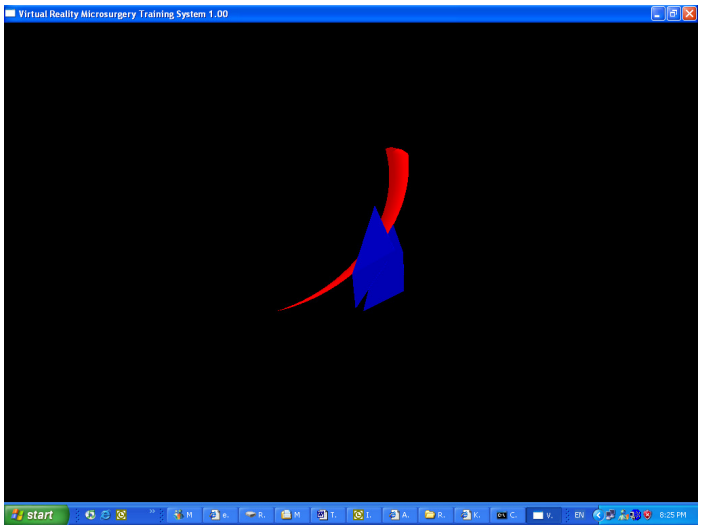


Figure 6.2-3 Incorrect grasp point and not grasping with tips

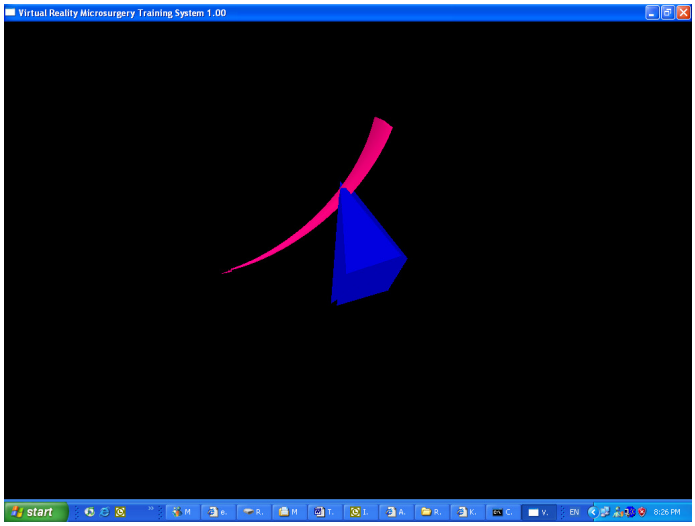


Figure 6.2-4 Incorrect grasp point but grasping with tips

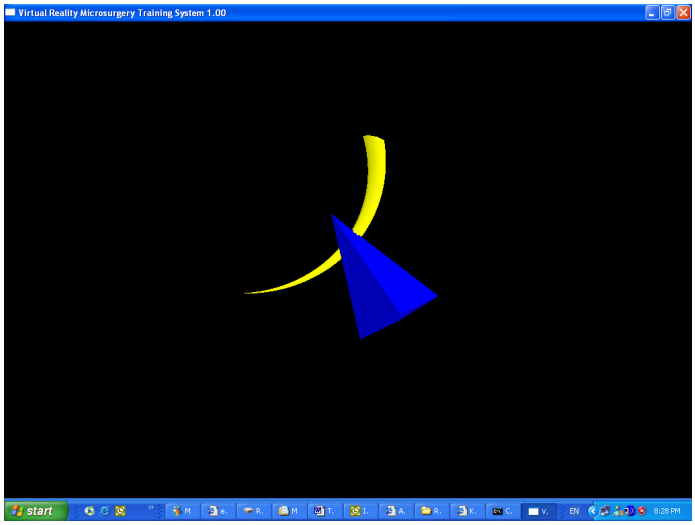


Figure 6.2-5 Correct grasp point but not grasping with tips

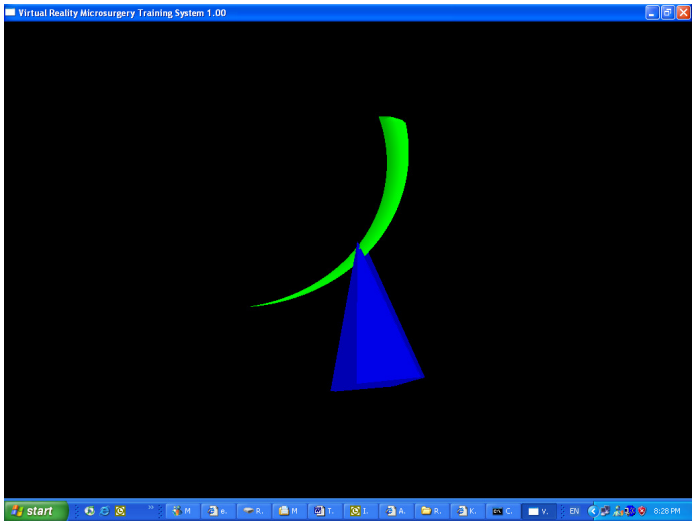


Figure 6.2-6 Correct grasp point and grasping with tips

6.2.4 Advanced Grasping Module II

The final module in the grasping series is the Advanced Grasping Module II as shown in Figure 6.2-7, a free hanging needle is presented and the student is to practice grasping the

needle under gravity. Similar to Advanced Module I, color cues are provided feedback about the student's grasp. In addition, the student has to control his grasping force, as show in Figure 6.2-8, excessive force on the needle will trigger a visual cue, whitening of the jaws.

After successfully grasping the needle, the student has to move the grasped needle to a randomly generated destination point and hold it at that location for 10 seconds (the point will turn to green when the needle is position close to the point and reverts back to blue after 10 seconds as shown in Figure 6.2-9. Throughout the motion, the pressure on the needle should not exceed a customizable level. The objective of this module is to train the student in maintaining a correct grasping force and the ability to accurately maneuver the needle to a desired location.

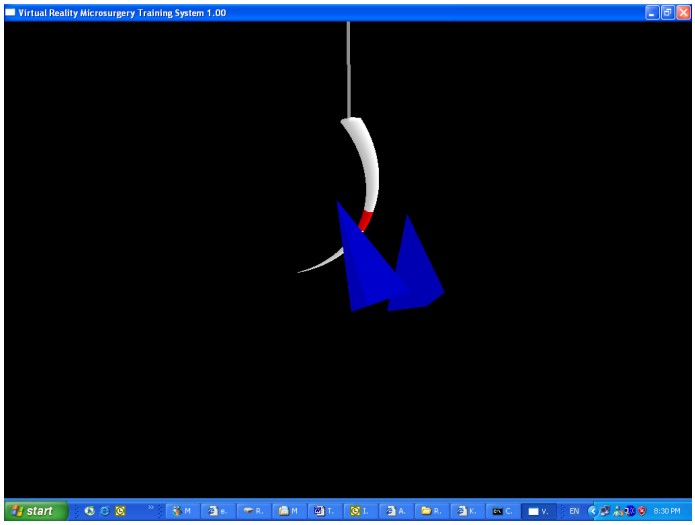


Figure 6.2-7 Advanced Module II

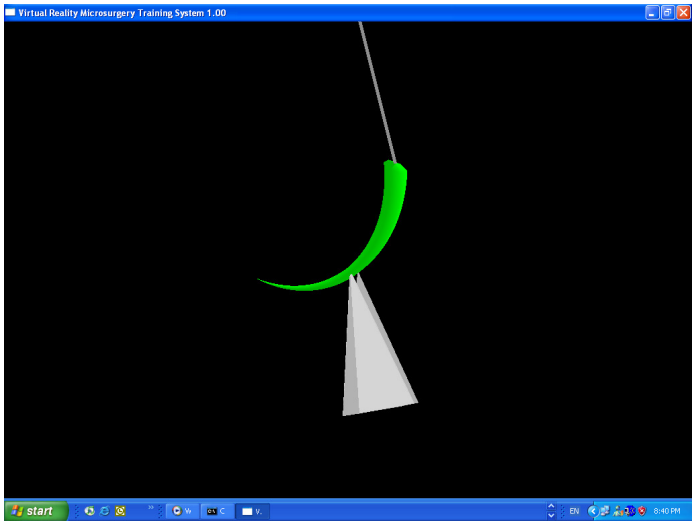


Figure 6.2-8 A good grasp but excessive force is used

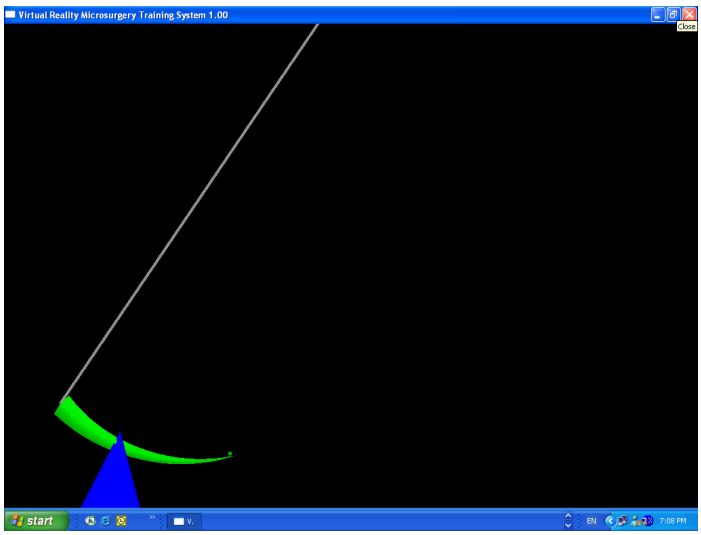


Figure 6.2-9 Moving the grasped needle to a specific point and holding it there

6.3 Grasping of a Suture

A separate module to train grasping of a micro-suture was also developed. In this module as shown in Figure 6.3-1, the student is presented with a hanging suture and he is instructed to grasp the edge of the suture and maneuver it to a randomly generated destination point. As in the advanced module II, the student should maintain a proper grasping force throughout the motion.

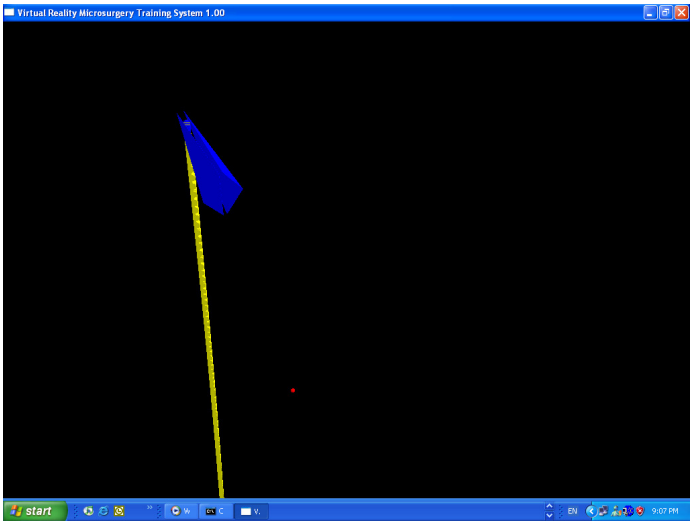


Figure 6.3-1 Grasping and moving a suture to desired point

6.4 Insertion of a Needle

The suturing series is developed to train the student in inserting a micro-needle after grasping. In these modules, it is noticeable that if the needle was grasped incorrectly, numerous difficulties will arise while inserting the needle.

The suturing module combines the subtask of grasping and inserting of the needle. The student is presented with a hanging needle and a tissue surface at different orientation (flat, right sloping, left sloping). The student is to grasp the needle successfully and insert the needle at a randomly generated insertion point on the surface of the tissue.

If the student attempts to insert the needle at an incorrect penetration angle, the needle will be deflected upward, reflecting the effect of a large reaction force twisting the alignment of the needle or even deforming the needle.

After penetration, the student is to maintain an almost perpendicular penetration angle with respect to the surface. Deviations will trigger a haptic cue resisting the continuing penetration of the needle.

If the needle is released in the midst of penetration, the needle will remain "stuck" to the surface, and the student is able to manipulate the needle in a limited manner (however, this will result in a bigger tissue tear). The student is able to grasp the needle again and continue with the insertion.

Throughout the insertion process, the "tear" on the tissue is being tracked and represented as red regions on the surface.

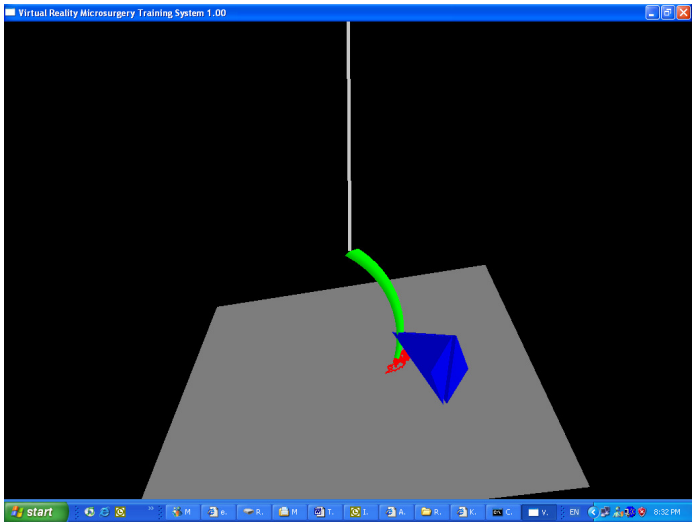


Figure 6.4-1 Insertion of the needle result in tissue tears

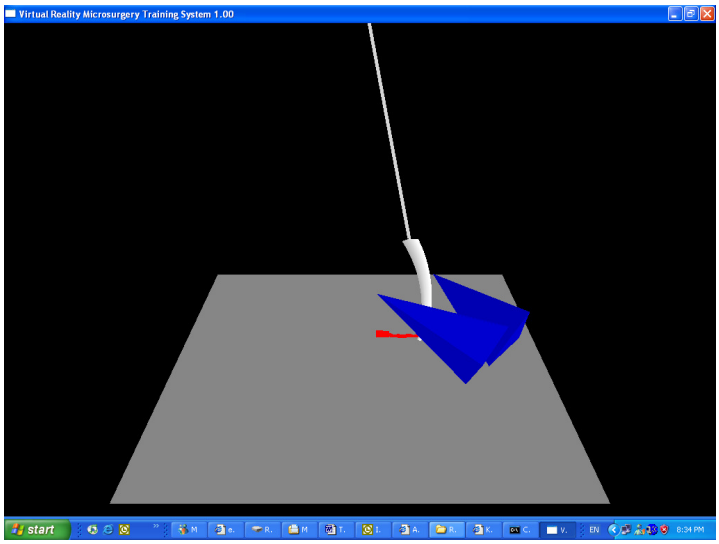


Figure 6.4-2 Limited manipulation when the needle is left on the surface

6.5 Performance Evaluation

An important part of training is performance evaluation of the student. And in order to properly evaluate the performance of any grasping or suturing practices, key traits of a good/bad practice should be identified and used as a basis for evaluation.

6.5.1 Grasping

The traits of a good grasp are that the micro-needle should be grasped vertically and longitudinally perpendicular to the needle holder without excessive force. And in addition, grasping should not occur at the micro-needle's proximal or distal extremities and the tips of the needleholder should not exceed the body of the micro-needle to avoid damaging tissues behind the micro-needle.

Hence, a highlighted region was created for each of the virtual needles used in the trainer where a function was written to check if the grasping occurs in the highlighted region. A check to evaluate the distance of the tips of the needleholder to the needle was also incorporated and the force applied to grasp the needle tracked. And with these information returned by the software, a color-coded feedback cue can be displayed to the student.

6.5.2 Wound Size

The wound size generated when the needle was inserted into the surface was chosen as another evaluation factor for insertion performance. This is because the deviations of the needle (due to tremors or poor techniques) during insertion, reflects the amount of tissue trauma inflicted during the surgery. As different needles have varying sizes, it is a more accurate gauge of performance if the wound size is represented as a percentage of the needle size. The wound size can be obtained as the software tracks the motion of the needle through the surface. It is to be noted that this is a geometric assumption for the wound size as no material properties effects are considered in the simulation.

6.5.3 Penetration Angle

The angle the needle makes with the surface as it penetrates the surface can also be used as another performance gauge. This is because as the needle passes through the tissue, it will drag/pull the surrounding tissues as it moves. And the greater the angle the needle makes with the tissue, the greater the drag, and hence the tissue trauma. Hence, the penetration angle was tracked by the software throughout the motion and can be presented as a graph.

6.6 Trial Run with VR Training Software

The training tutorials were installed onto the completed training system, with the Reachin setup. The main objective of the trial run is to ensure that the training system is functioning properly and to identify and correct any bugs in the software.

6.6.1 Methodology

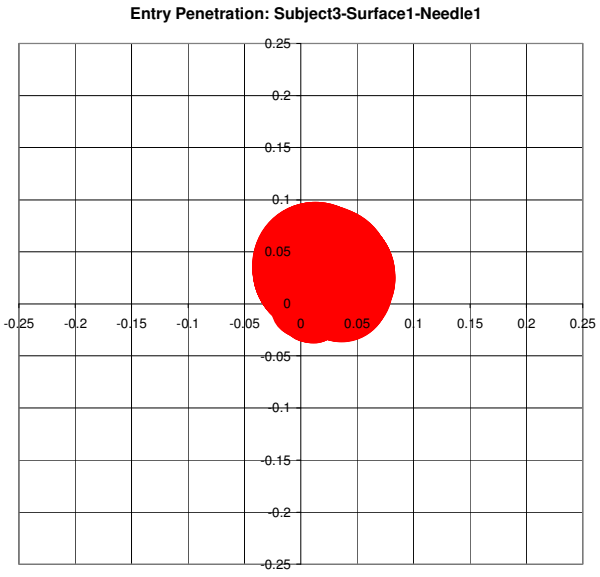
A trial run with 3 subjects was conducted with the training modules described previously. Two of the subjects were totally new to virtual reality and microsurgery, while the third had undergone a basic course in microsurgery and practiced significantly on the virtual reality trainer.

The subjects were instructed verbally on the proper techniques in grasping and inserting the micro-needle. They were then allowed to familiarize themselves with the virtual environment for at least 30 minutes; they were required to practice grasping and manipulating spheres, cylinders and needles.

A variety of needles (type 2/8, 3/8 and 4/8) and surfaces (left sloping, right sloping and flat) were used to train the subjects. Both observations and raw data were collected.

6.6.2 Data Collection

The VR-based trainer provides numerous data which can be compiled and presented so that the instructor is able to gauge the performance of the students. Wound size and Penetration angles are some of the results that can be obtained from the raw data. Some examples of typical results provided by the software are shown below. Figure 6.6-1 demonstrates the results of a well inserted needle. Figure 6.6-2 demonstrates a poorly inserted needle.



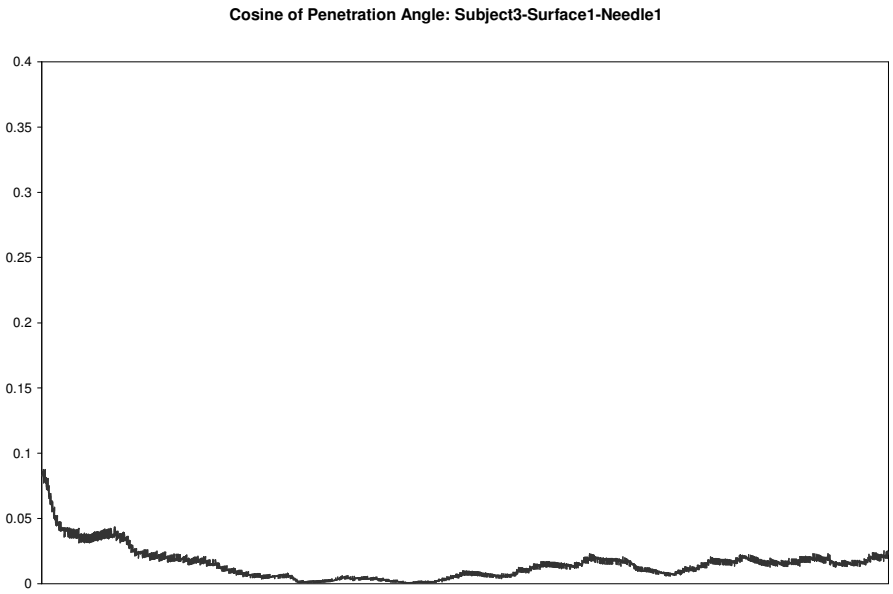


Figure 6.6-1 Subject3: Insertion#3 of 2/8 Needle on Flat Surface

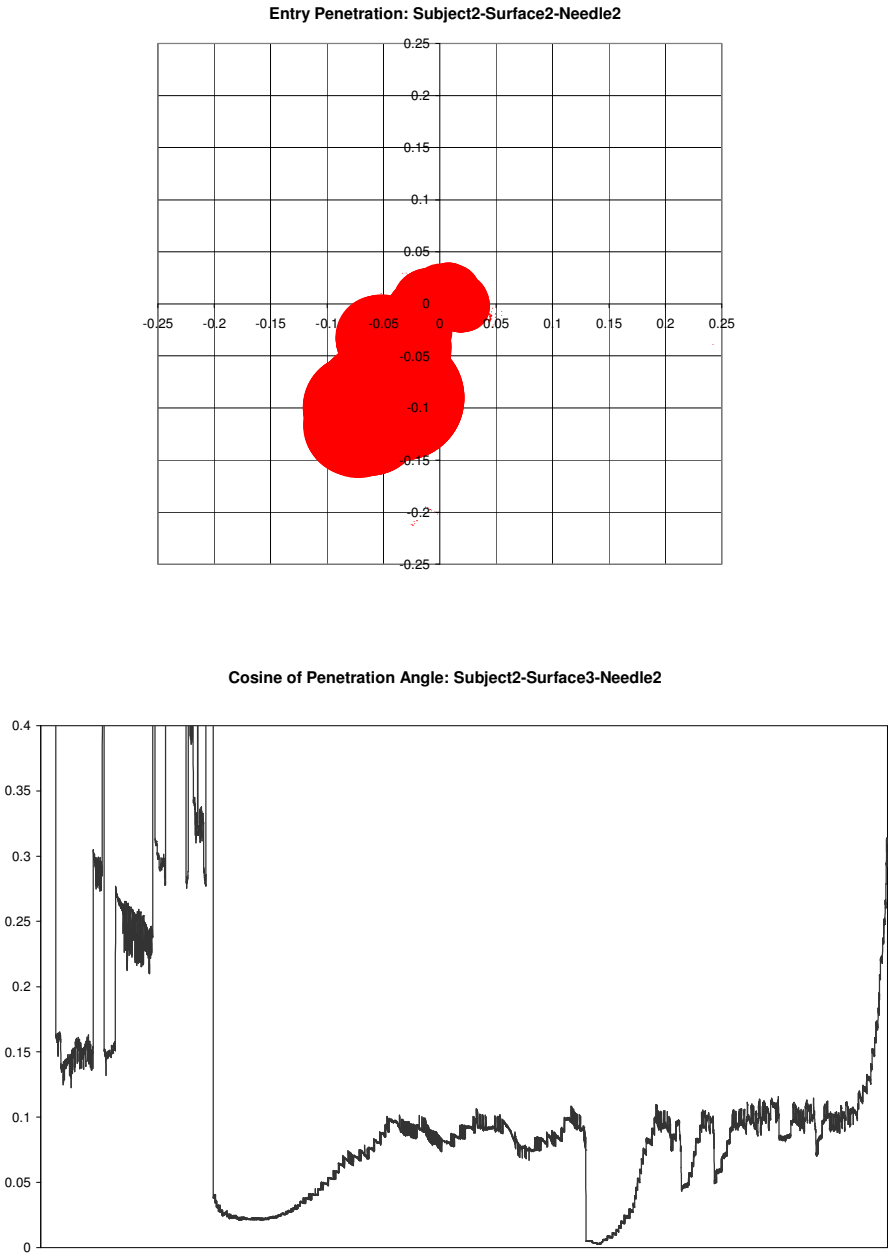


Figure 6.6-2 Subject2: Insertion#2 of 3/8 Needle on Right Sloping Surface

Observations and Results

The less experienced subjects had most difficulties with extremely curved needles, and required numerous attempts before successfully grasping the needles. They also have great difficulty in perceiving the angle of the needle with the surface or the needleholder. On the average they took more attempts and a relatively long time (1~5 minutes, not inclusive of unsuccessful attempts) to complete a single suture successfully.

The more experienced subject only exhibit slight difficulties with extremely curved needles and in some cases performed better with the extremely curved needle. Hence, it can be hypothesized that with sufficient training in the system, the difficulties arising from the curvature of the micro-needle can be overcome. The experienced subject also required less attempts before successfully grasping the needles and was able to accurately judge the angle the needle is making with the surface or the needleholder. He was also able to perform a successful suture with a single attempt and with significantly less time (< 1 minute).

6.6.3 Discussion

As this is only a trial run to test the operation of the system, there were insufficient results available for analysis. However, it can be observed that most of the subjects had problem perceiving and judging the angle the needle makes with the surface or the needleholder, it is recommended that an additional module is to be developed to train the students in estimating angles.

7 Conclusion

7.1 Virtual Reality based Trainer

A virtual reality based training system to train grasping and inserting of the micro-needle was developed in this project. A Reachin Display was used as the setup for the training system, while a Phantom^{tm} Desktop was used as the spatial/orientation input and haptic feedback for the trainer. In addition, a modified needleholder with a displacement sensor was integrated with the Phantom^{tm} Desktop.

7.2 Physics Engine

A library of APIs was developed for the system such that rapid generation of customized training modules is possible. The library of APIs can generate and co-ordinate interactions of customized virtual objects (found in microsurgery) with the needleholder (and the Phantom^{tm} desktop). The various models of the virtual objects found in the library are, the virtual jaws (needleholder), needle, suture, sphere, cylinder and surface. In each of the models except for the virtual suture, rigid body dynamics were implemented. For each of the virtual objects, effective collision detection and response were studied and implemented. One of the algorithms used to optimized computational time is the Boundary Sphere Hierarchy (BSH). A wrapper class was also developed to supervise the general interactions between the virtual objects, and each loop (iteration) of the simulation was able to complete within 1 ms which is the requirement for haptic interface.

7.3 Training Modules

A series of tutorial software to train grasping and inserting of micro-needle was also developed using the library mentioned previously. It was able to display various cues to inform the user of his performance e.g. did he grasp at the correct point. It was also able to track and store a variety of information which can be analyzed and used to determine the performance of the student by the instructor.

7.4 Trial Run

Using the setup and software mentioned previously, a trial run with 3 subjects was conducted and initial results are encouraging. Detailed information on the trial run can be found in Chapter 6 and Appendix D and E.

7.5 Future Works

Although the API library has most of the objects that can be found in microsurgery, it is still by no means complete. The virtual surface is relatively basic, and should be implemented with a multi-scale mesh [4] to better simulate the soft tissue commonly found in microsurgery. In addition, visco-elastic behaviors of the blood vessel tissue can be integrated into the multi-scale mesh. In addition, the virtual surface should be readily described into any form, such as a smooth flat flap, or a cylindrical flap, etc. This would allow the development of more advanced training modules.

Currently, the virtual needle is modeled as a rigid body with the assumption that it will not experience deformations of any form. However in reality, the micro-needle deforms when it is grasped or inserted improperly. A better and more realistic approach would be to implement the deformation effects for the virtual needle. For example, a virtual coarse grained finite elements (FE) mesh for the virtual needle can be implemented in parallel with the current needle model. This is to allow a basic analysis on the stress-strain experienced by the virtual needle during the virtual surgery. And thus, a scoring function can be developed based on the stress-strain field of the needle, together with existing criteria (e.g. grasp location, pressure, etc). Alternatively, the force and impulse experienced by the virtual needle can also be recorded and analyzed offline. In additionl, environmental effects like electrostatic effects, and viscous effects due to environmental fluids can also be implemented to improve the realism of the simulation.

Lastly, as one of the observations during the trial run was that the subjects had difficulties in judging the angle between the needle and the surface/needleholder. It is recommended that an additional module, to train estimating of angles, to be incorporated into the training system.

Bibliography

1. The American Heritage® Dictionary of the English Language http://www.thefreedictionary.com/microsurgery
2. MicroSurgeon.org, http://www.microsurgeon.org
3. Indiana University Medical Center, Microsurgery Training Program, http://www.iupui.edu/~micrsurg
4. F. Wang. Development of Virtual Reality based Microsurgery Trainer. M.Eng dissertation thesis, National University of Singapore, 2004.
5. 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.
6. E. Erel, B. Aiyenibe, and P.E.M. Butler. Microsurgery simulators in virtual reality: Review. Journal of Microsurgery, 23(2):147-152, 2000.
7. 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.
8. M. Haruno, D.M. Wolpert, and M. Kawato. Mosaic model for sensorimotor learning and control. Neural Computation, 13:2201-2220, 2001.
9. 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.

10. 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.
11. Cao, C.G.L., Webster, J.L., Perreault, J.O., Schwaitzberg, S., Rogers, G.. Visually Perceived Force Feedback in Simulated Robotic Surgery.
12. Caroline G. L. Cao, Christine L. MacKenzie, Shahram Payandeh. Task and Motion Analyses in Endoscopic Surgery. 1996 ASME IMECE Conference Proceedings: 5th Annual Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Atlanta, Georgia. pg. 583-590.
13. Julian Mackay-Wiggan and Desiree Ratner. Suturing Techniques. http://www.emedicine.com/derm/topic828.htm
14. Stephen Y Lai and Daniel G Becker. Sutures and Needles. http://www.emedicine.com/ent/topic38.htm
15. David Ota, Bowen Loftin, Tim Saito, Robert Lea, James Keller. Virtual Reality in Surgical Education.
16. S. Pednekar and I. A. Kakadiaris. Applications of Virtual Reality In Surgery. Proceedings of the Indian Conference on Computer Vision, Graphics, and Image Processing(ICVGIP 2000), pp. 215-223, Banglore, India, December 20-22, 2000.
17. M. K. Holden and T. Dyar, Virtual Environment Training: a new tool for Neurorehabilitation, vol. 26 No. 2, 2002.
18. Duane Boman, Virtual Environment Applications, SRI International - Business Intelligence Program, May 1995.

19. Bradshaw, G. Bounding Volume Hierarchies for Level-of-Detail Collision Handling. PhD Thesis - Dept. of Computer Science, Trinity College Dublin, May 2002.
20. Chris Hecker. Rigid Body Dynamics. Game Developer Magazine, 1996 ~ 97. http://www.d6.com/users/checker/dynamics.htm
21. Bob Akka. Writing Stereoscopic Software for StereoGraphics® Systems Using Microsoft Windows® OpenGL. StereoGraphics Corporation August 13, 1998.
22. Dan Sunday. Geometry Algorithms. http://softsurfer.com
23. Randy Haluck, Roger Webster, Dean Zimmerman, Betty Mohler, Alan Synder, Mike Melkonian. A Prototype Haptic Suturing Simulator.
24. Randy Haluck, Roger Webster, Betty Mohler, Mike Melkonian. A Haptic Lumbar Puncture Simulator. In Proceedings of Medicine Meets Virtual Reality (MMVR '2000) Conference.
25. Jonathan Blow. Practical Collision Detection. In the Proceedings of the Computer Game Developer's Conference, 1997.
26. M. Lin, D. Manocha, J. Cohen and S. Gottschalk. Jean-Paul Laumond and M. Overmars, A.K. Peters. Collision Detection: Algorithms and Applications. In Proceedings of Algorithms for Robotics Motion and Manipulation, pp. 129-142 eds.
27. Andrew Witkin and David Baraff. Physically Based Modeling: Principles and Practice. Online Siggraph '97 Course notes. http://www-2.cs.cmu.edu/~baraff/sigcourse

28. Tim Poston, Ankur Dhanik, Etienne Burdet, Teo Chee Leong: Haptics of Buckling. WHC 2005: 299-307.

29. Sensable Technologies. http://www.sensable.com

30. Force Dimension. http://www.forcedimension.com

31. Reachin Technologies. http://www.reachin.se

Appendices

A. Phantom™ Desktop Technical Specifications

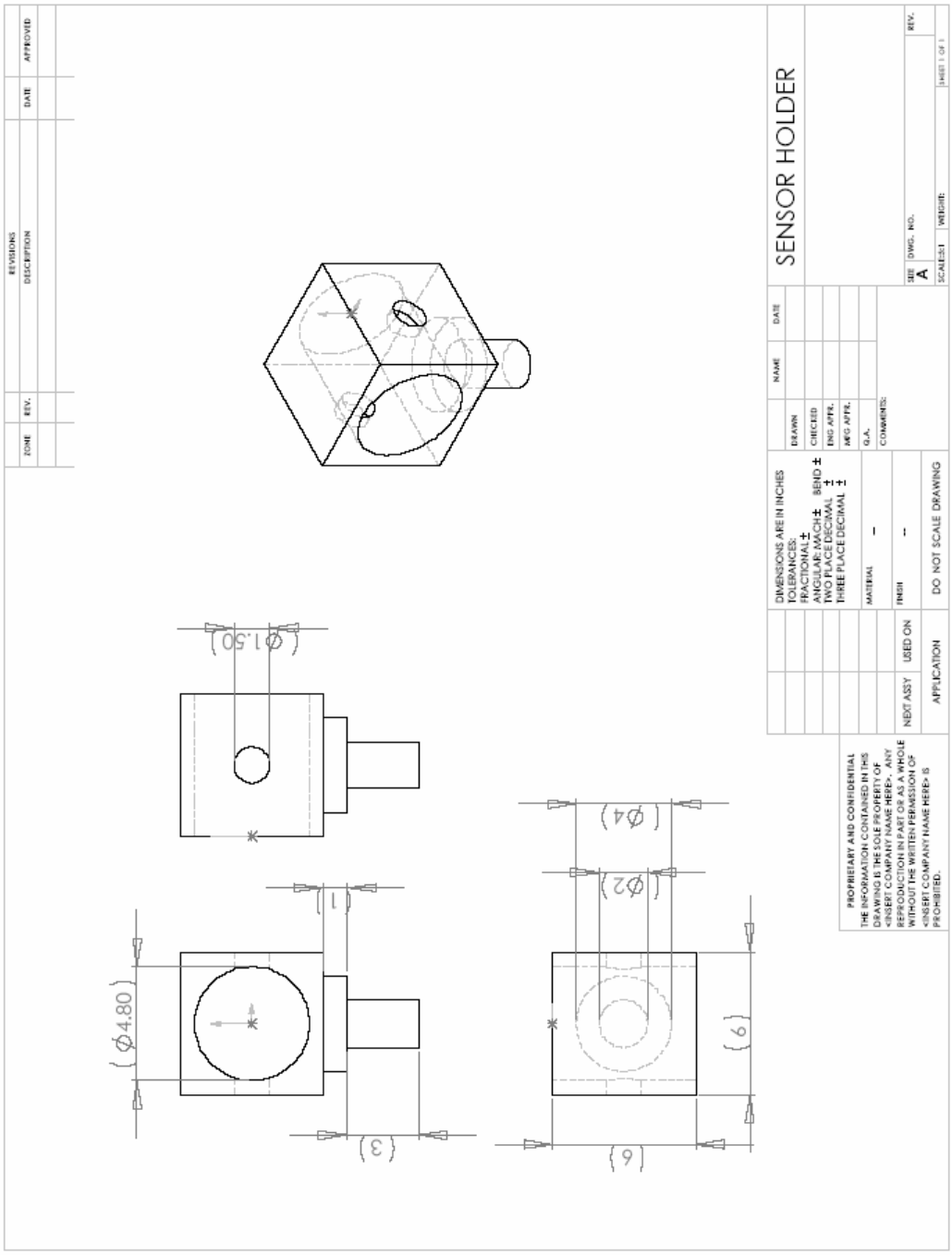
Force feedback workspace	~6.4 W x 4.8 H x 4.8 D in. > 160 W x 120 H x 120 D mm.
Footprint (Physical area device base occupies on desk)	5 5/8 W x 7 1/4 D in. ~143 W x 184 D mm.
Weight (device only)	6 lbs. 5oz.
Range of motion	Hand movement pivoting at wrist
Nominal position resolution	> 1100 dpi. ~ 0.023 mm.
Backdrive friction	< 0.23 oz. (0.06 N)
Maximum exertable force at nominal (orthogonal arms) position	1.8 lbf. (7.9 N)
Continuous exertable force (24 hrs.)	0.4 lbf. (1.75 N)
Stiffness	X axis > 10.8 lbs. / in. (1.86 N / mm.) Y axis > 13.6 lbs. / in. (2.35 N / mm.) Z axis >8.6 lbs. / in. (1.48 N / mm.)
Inertia (apparent mass at tip)	~0.101 lbm. (45 g)
Force feedback	x, y, z
Position sensing [Stylus gimbal]	x, y, z (digital encoders) [Pitch, roll, yaw (± 3% linearity potentiometers)
Interface	Parallel port
Supported platforms	Intel-based PCs
GHOST® SDK compatibility	Yes
3D Touch™ SDK compatibility	Yes
Applications	Selected Types of Haptic Research, the FreeForm® Modeling™, and the FreeForm® Modeling Plus™ systems

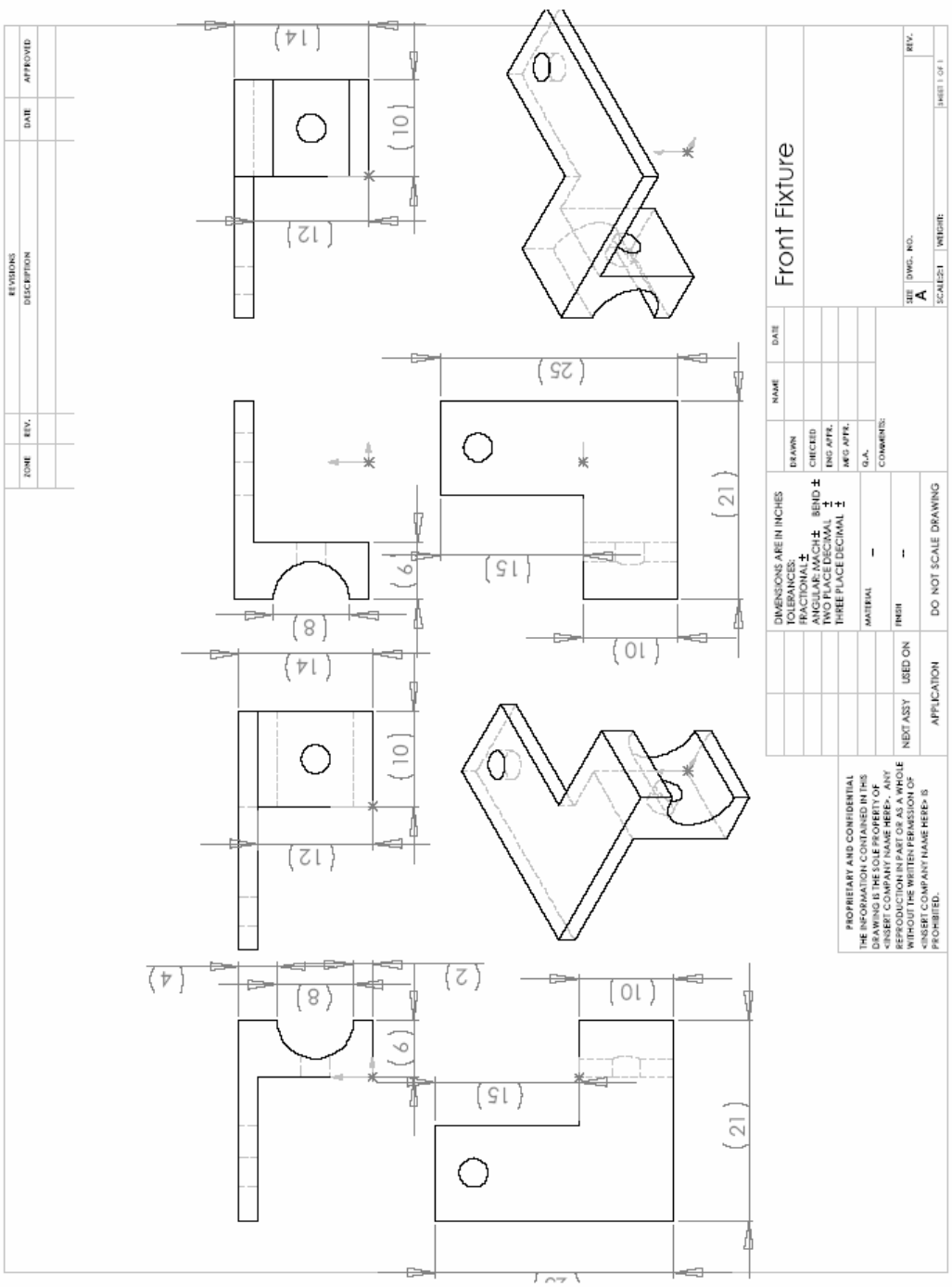
B. Subminiature DVRT Technical Specifications

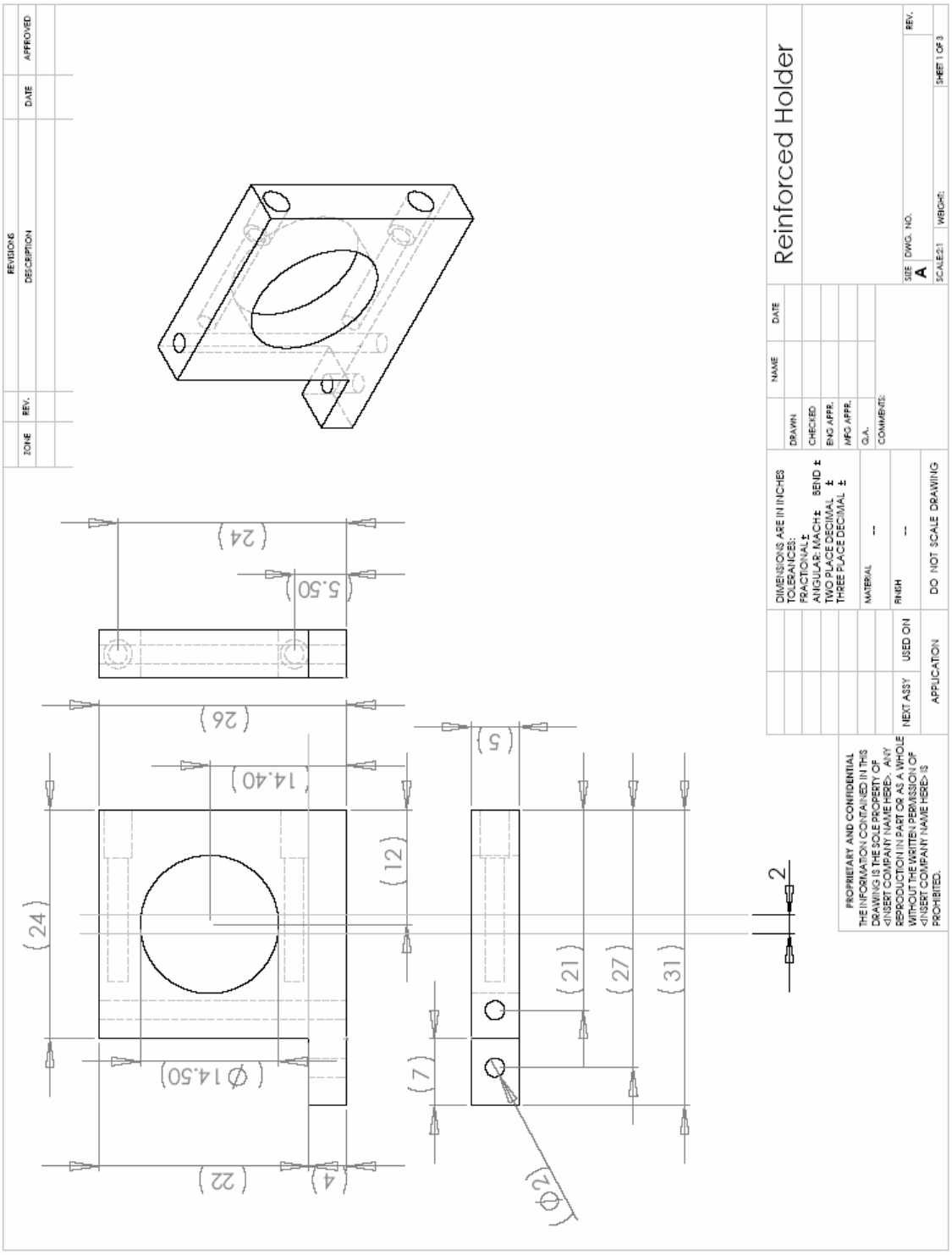
Linear Stroke Lengths	8, 24, and 38 mm (standard resolution)
	6 mm (high resolution)
	500 microns or less depending on your required resolution spec. (nano resolution)
Nonlinearity	+/- .4% over 6 mm (max. 8mm stroke)
	+/- .7% over 8 mm (max. 8mm stroke)
	+/- .6% over 24 mm (max. 24mm stroke)
	+/- .17% over 8 mm (max. 24mm stroke)
	+/- .3% over 24 mm (max. 38mm stroke)
	+/- .7% over 38 mm (max. 38mm stroke)
	+/- 1.2% over 6 mm (high res. 6mm stroke)
	+/- .4% over 3 mm (high res. 6mm stroke)
Sensitivity	1 volt/mm typ. (for 8 mm stroke)
Signal to noise	4200 to 1 (with filter 3 dB down at 900 Hz, standard resolution; 466 to 1 (unfiltered)
Resolution	1.9 microns (standard version 8mm stroke)
	5.7 microns (standard version 24mm stroke)
	0.6 microns (high resolution 6mm stroke)
	10 nanometers (nano resolution optimized for 200 microns of stroke with filter 3 dB down at 80 Hz)
Frequency response	7 Khz (unfiltered) -contact us for faster response reqirements.
Temp. coeff. offset	.002% / degree C (typical)
Temp. coeff. span	.030% / degree C (typical)
Hysteresis	+/- 1 micron
Repeatability	+/- 1 micron
Synch. demod. input	+/- 7 volts min. @ 10 milliamps/rail
Synch. demod. output	+/- 4 volts typ., buffered line out
Overall body length	approx. 3.5 times linear stroke
	~29 mm for 8 mm stroke
	~84mm for 24 mm stroke
	~118 mm for 38 mm stroke
Outside Diameter	4.76 mm (3/16 inch)
Housing material	300 series stainless steel (SS); 400 SS optional, 5/16-24 threaded
Leadouts	45 cm (18"), multistranded, shielded, SS reinforced, teflon insulated
Connectors	sensor: keyed Lemo 4-pin
Strain relief	Stainless steel center conductor for tensile strain relief
Operating temperature	- 55 to 105 degrees C (standard); -55 to 175 degrees C (high temp. option)

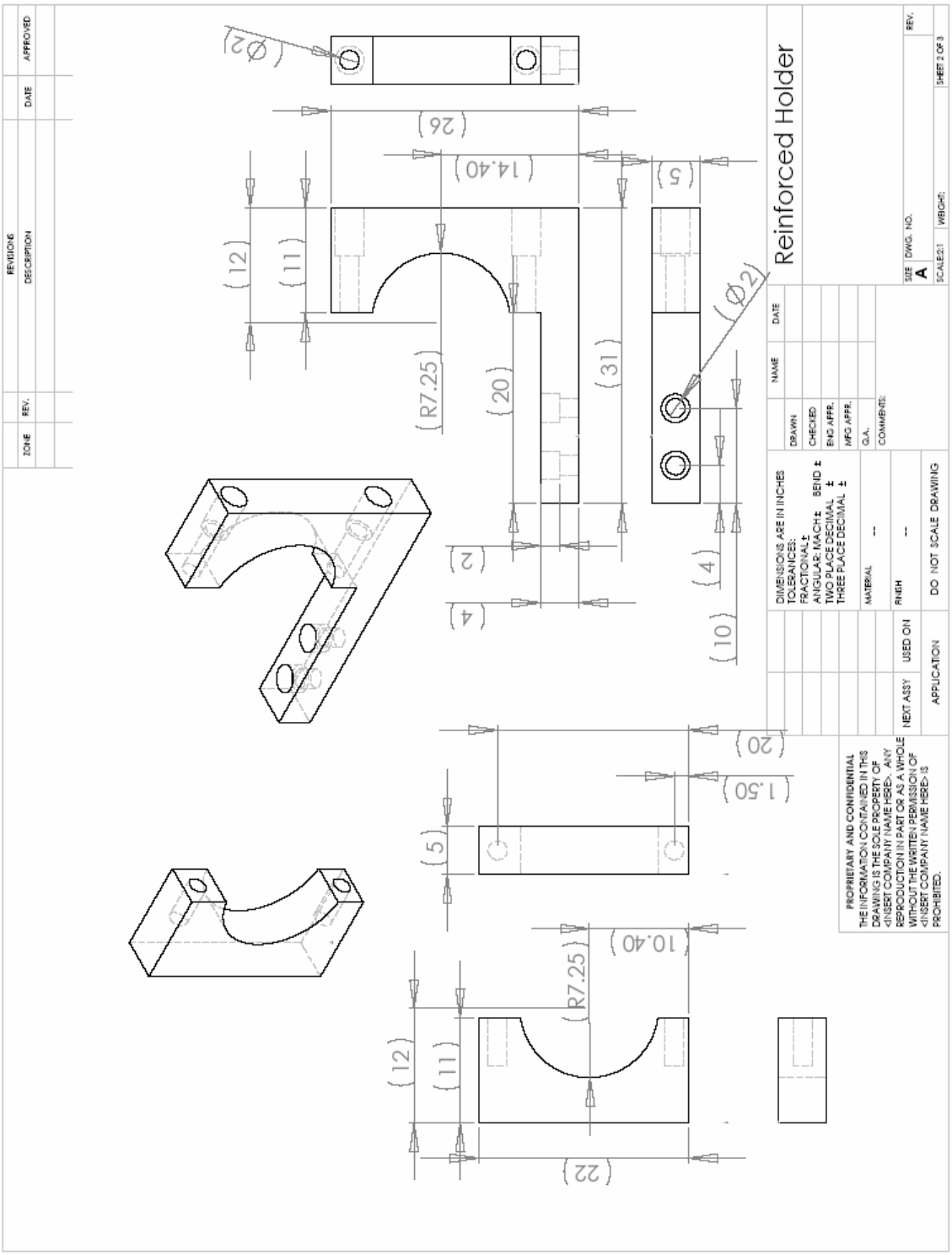
Core weight	350 milligrams (for 8 mm stroke)

C. Modified Needleholder Fixtures Technical Drawings




Front Fixture
REVISIONS
ZONE
REV.
DESCRIPTION
DATE
APPROVED
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ± ±
THREE PLACE DECIMAL ± ±
MATERIAL –
FINISH –
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
SIZE
A
DWG. NO.
REV.
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.


REVISIONS
ZONE
REV.
DESCRIPTION
DATE
APPROVED
Reinforced Holder
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
SIZE
A
DWG. NO.
REV.
SCALE:2:1
WEIGHT:
SHEET 1 OF 3
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL --
FINISH --
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
24
5.50
26
14.40
5
12
24
Ø 14.50
22
4
7
21
27
31
2
Ø2


Reinforced Holder
REVISIONS
ZONE
REV.
DESCRIPTION
DATE
APPROVED
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
NAME
DATE
SIZE
DWG. NO.
REV.
A
SCALE:2:1
WEIGHT:
SHEET 2 OF 3
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
MATERIAL
--
FINISH
--
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
(Ø2)
(26)
(14.40)
(5)
(12)
(11)
(R7.25)
(20)
(31)
(Ø2)
(2)
(4)
(4)
(10)
(20)
(1.50)
(5)
(10.40)
(R7.25)
(12)
(11)
(22)

D. Trial Run Details

D1. Test Subjects

Three subjects were involved in this preliminary study, of which, one of them had practiced more intensively with the simulator than the other two. It is regrettable that there is no medical student or surgeon among the test subjects. The two less experienced subjects are notated as Subject 1 and Subject 2, while the more experienced subject is notated as Subject 3.

It can be noted that Subject 2 had great difficulties in performing the insertion with the more curved needle. The results from Subject 2 are generally inconsistent despite numerous attempts.

D2. Needle Types

Three types of needle were used in this experiment, namely a 2/8, 3/8 and 4/8 needles, where the fraction before the needle represents the fraction encompassed by the arc of the needle. The needles are notated as Needle 1, Needle 2, Needle 3 respectively.

It can be noted that only Subject 3 had done any experiment with Needle 3. This was because despite much practices, both Subject 1 and 2 were unable to perform any successful insertion with Needle 3.

D3. Surfaces

Three types of surfaces were used in this experiment, namely a flat, left sloping and right surfaces. These surfaces are simplistic planar models of the soft tissues. The surfaces are notated as Surface 1, Surface 2, Surface 3 respectively.

D4. Wound Size and Penetration Angle

The results that are used in the experiment to gauge the performance of the subjects are the wound size and penetration angle. The wound size is essentially the area on the surface that had been disturbed by the needle. A large wound size would represent a poor suture as significant tissue damage had been done to the blood vessel. The wound size is represent as a percent of between the largest distance of the edge of the wound to its center and the radius of the needle body.

The angle of penetration throughout the insertion is also an important factor. If angle deviates significantly from 90°, there will be significant pull of the needle on the tissue which will result in additional tissue damage. And in this experiment, instead of the angle of penetration, the cosine of the angle of penetration was used, such that, a zero would represent a perfect insertion, while a one is the worse possible insertion.

D5. Observations

Throughout the experiment, it can be generally observed that a more curved needle posed a greater difficulty to the inexperienced subjects. The left sloping surface also posed a significant problem, particularly when using Needle 2.

The inexperienced subjects generally take around 5~15 minutes to perform a single suture, while the more experienced one take around 1 minute.

Inexperienced subjects seemed to find great difficulties in perceiving the orientation of the needle with respect to the needleholder and surface, despite the stereoscopic vision. Most of the unsuccessful insertions are due to their inability to determine the orientation of the needle with respect to the surface.

This problem is further aggravated by the fact that their inability to perceive the needle orientation resulted in an improper grasp (as in the needle is not grasped in a correct orientation) which make it even more difficult to determine the orientation of the needle point with respect to the surface.

D6. Wound Size versa Needle Type and Surface Type

As shown in Figure D-1, it can be observed that for Subject 2, significantly bigger wounds were created when a Needle 2 was used. However for Subject 1 and 3, the increases in wound size were not significant except for the case a flat surface for Subject 1. In fact, for

Subject 3, there was a slight decrease in wound size for a right sloping surface with Needle 3.

Ignoring the inconsistent results from Subject 2 for Needle 2, it can also be observed that Needle 1 generally performed better on a flat surface, while Needle 2 and 3 performed better on a sloping surface.

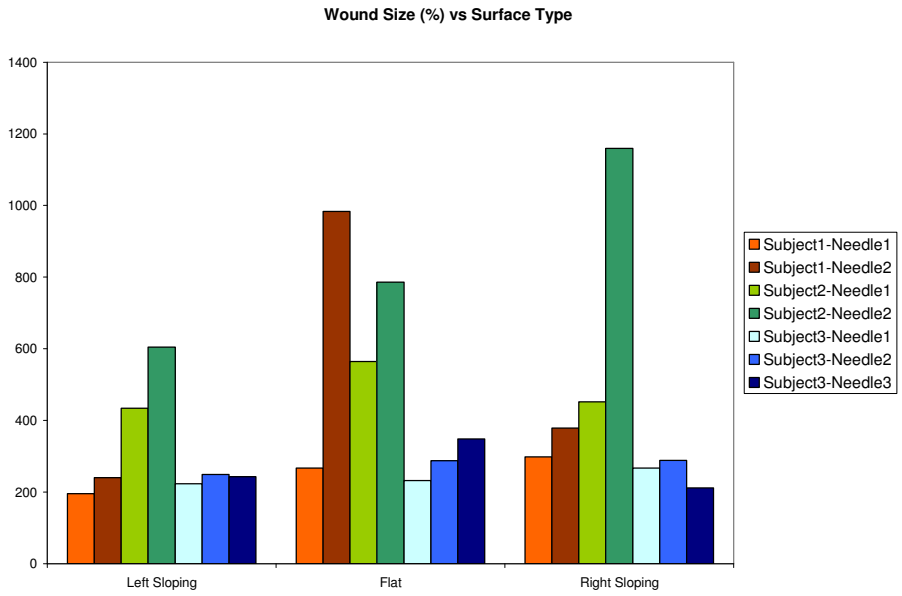


Figure D-1. Wound Size resulted wrt Surface Slope and Needle Type

D7. Penetration Angle versa Needle Type and Surface Type

As shown in Figure D-2, it can be observed that generally maintain a correct penetration angle was harder with a more curved needle. It can also be observed that the Left Sloping surface was significantly more difficult for all three subjects, although less so for the more experienced Subject 3.

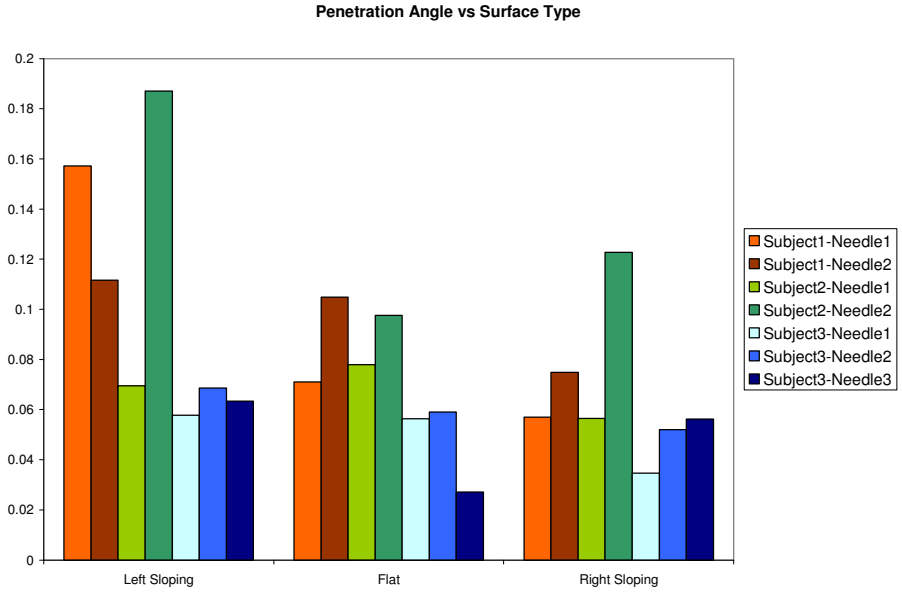


Figure D-2 Mean Penetration Angle wrt Surface Slope and Needle Type

D8. Discussion

From the results of this experiment, it is plausible to hypothesize the following points:

- Inexperienced students have difficulty in perceiving orientations (angle) in virtual environment.
- Inexperienced students have more difficulties with an extremely curved needle, which will decrease with practice.
- Left sloping surface is more difficult to suture, this is probably because the suturing motion is awkward for right handed students.

Having observed the difficulty in perceiving orientations in virtual environment, it is desirable to develop an additional module to train the students in identifying orientations. As noted previously, with training an extremely curved needle will not pose great difficulty to the surgeons. A more curved needle is generally useful for suturing where the direction of approach is restricted by the workspace, a more curved needle is better at piercing a sloped surface, however, it is harder to maintain the correct angle once the tissue is pierced.