A FUZZY LOGIC BASED VIRTUAL SURGERY SYSTEM

A Thesis
Presented to
The Graduate Faculty of The University of Akron

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

Shanthanand Kutuva Rabindranath
August, 2006
# A FUZZY LOGIC BASED VIRTUAL SURGERY SYSTEM

Shanthanand Kutuva Rabindranath

<table>
<thead>
<tr>
<th>Thesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved:</td>
</tr>
<tr>
<td>Advisor</td>
</tr>
<tr>
<td>Dr. Yingcai Xiao</td>
</tr>
<tr>
<td>Co-Advisor</td>
</tr>
<tr>
<td>Dr. Narender P. Reddy</td>
</tr>
<tr>
<td>Reader</td>
</tr>
<tr>
<td>Dr. Xuan-Hien T. Dang</td>
</tr>
<tr>
<td>Reader</td>
</tr>
<tr>
<td>Dr. Kathy J. Liszka</td>
</tr>
<tr>
<td>Reader</td>
</tr>
<tr>
<td>Dr. Zhong-Hui Duan</td>
</tr>
</tbody>
</table>
ABSTRACT

Virtual Reality (VR) and associated technologies are experiencing growing importance in medicine and surgery. Virtual surgery simulation can help surgeons to explore optimal surgical operations, thus reducing risks in the real surgery. It helps doctors to train medical students in a cost effective manner.

Virtual surgery simulation demands fast and real time performance. The finite element method has been used for surgery simulations; however, finite element method is computationally intensive and is unsuitable for real time simulation of cutting large amount of tissue in arbitrary paths.

We have developed a fuzzy logic based system which involves significantly reduced computational time and provides real time simulation of soft tissue cutting in any desired paths. The user-exerted forces on a simulated cutting tool held by the user are dynamically measured and fed to the fuzzy logic system. The user-exerted forces and the stiffness of the element are first fuzzified and fed to a fuzzy rule base to provide the output membership values. These output membership values are then defuzzified using the centroid defuzzification technique to provide a crisp output of the cutting depth. The computed cutting depth is then sent to the visualization module, which provides realistic rendering of surgical tools, surgery objects, and their interactions to the surgeons in a 3D virtual environment. It animates the entire surgical process in the virtual environment and
allows surgeons to view the surgical simulation in any vantage point. To realistically render the soft tissue during a virtual surgery process, his study also explores a novel method to take care of structural and textural changes in the soft tissue by splitting and cloning internal mesh nodes.

A virtual surgery system, completed with its five subsystems, has been designed and implemented. A multi threaded approach has been used to implement the system. This isolates the data acquisition system from the simulation system so that both modules can be synchronized without any speed penalty. This also makes the system more responsive to user interactions.

Experiments with the system have shown that the fuzzy logic based virtual surgery system is capable of simulating soft tissue cutting along any desired path in real time. Further work is under way to measure the accuracy of the system. It has also been proposed to port the system to a distributed environment so that virtual surgical operations can be broadcasted to the internet.
DEDICATION

I dedicate this thesis to my wonderful mother, Jayanthimala, my brother, Rajesh, and my sister, Dr. Shalini, without whose love and guidance I would never have realized this dream.
ACKNOWLEDGEMENT

I would like to thank Dr. Xiao, for his help as my professor and advisor, for motivating and encouraging me, and for genuinely caring for my personal and professional development.

I would like to thank Dr. Reddy for developing the original concept of Fuzzy Logic Based Virtual Surgery System and for giving me an opportunity to work on the project. Thanks for teaching me various concepts and helping me throughout the project.

Dr. Pelz, thanks for recognizing my potential and giving me an opportunity to pursue my education in this university. Thanks for awarding me funding, without which I think I could not have pursued my master’s program.

The conversations and discussions with Dr. Hariharan and Dr. Gao were illuminating and I would like to thank Dr. Hariharan and Dr. Gao for explaining the various nuances of Virtual Surgery System. I would like to thank my committee, Dr. Duan, Dr. Liszka, Dr. Dang, for untiringly helping and guiding me. I would like to thank Saket for helping me to set up research equipments and helping me in the research study. I would like to thank all my group members for their valuable help.
I would like to thank all my gurus for helping me throughout my academic career in this University. I thank all my friends here at The University of Akron for supporting and encouraging me to accomplish my thesis. I bow to the blessing of Almighty.
TABLE OF CONTENTS

LIST OF FIGURES…………………………………………………………………………xiv

LIST OF TABLES…………………………………………………………………………xvii

CHAPTER

I. INTRODUCTION………………………………………………………………………1

1.1 Importance of Virtual Reality in Surgery……………………………………..1

1.2 Background.............................................................................................1

1.3 Need for further study and Outline of the Thesis ............................... ..3

II. SYSTEM DESIGN……………………………………………………………………5

2.1 System Components..................................................................................5

2.1.1 Model Definition Module.................................................................6

2.1.2 Data Acquisition Module.................................................................6

2.1.3 Simulation Module...............................................................................6

2.1.4 Surgical Animation Module.............................................................7

2.2 Design....................................................................................................7

2.2.1 Model Definition Module.................................................................8

2.2.1.1 Functionality..................................................................................10

2.2.2 Data Acquisition Module.................................................................10

2.2.2.1 Functionality..................................................................................10
2.2.3 Simulation Module.........................................................11
  2.2.3.1 Functionality.......................................................13
2.2.4 Fuzzy Control System.....................................................13
  2.2.4.1 Functionality.......................................................18
2.2.5 Surgical Animation Module..............................................18
  2.2.5.1 Functionality.......................................................19

III. MODEL DEFINITION MODULE – IMPLEMENTATION......................20
  3.1 Overview.............................................................................20
  3.2 Key Definitions..............................................................20
  3.3 Data Structures and Algorithms..........................................21
    3.3.1 Boundary Node Storage Structure....................................21
      3.3.1.1 Pseudo Code......................................................23
      3.3.1.2 Complexity Analysis............................................23

IV. DATA ACQUISITION MODULE – IMPLEMENTATION..........................25
  4.1 Overview.............................................................................25
  4.2 Key Definitions..............................................................25
  4.3 Data Structures and Algorithms..........................................26
    4.3.1 Producer consumer algorithm.........................................26
      4.3.1.1 Working............................................................28
      4.3.1.2 Pseudo Code......................................................29
    4.3.2 Two Way Hand Shake Protocol....................................31
      4.3.2.1 Pseudo Code......................................................32
4.3.3 Median Filtering ................................................................. 32
    4.3.3.1 Pseudo Code ............................................................ 33
    4.3.3.2 Formula ................................................................. 35
    4.3.3.3 Complexity Analysis .................................................. 35

V. FUZZY CONTROL SYSTEM – IMPLEMENTATION .......................... 37
    5.1 Overview ........................................................................... 37
    5.2 Key Definitions .............................................................. 38
    5.3 Data Structures and Algorithms ........................................ 38
        5.3.1 Defining Membership Functions ................................... 39
            5.3.1.1 Characteristics of Membership Functions .......... 39
            5.3.1.2 Steps to Define a Membership Function ............. 39
            5.3.1.3 Formula .......................................................... 42
        5.3.2 Defining the rule base ................................................. 43
        5.3.3 Gauging the Depth of Cut .......................................... 44
            5.3.3.1 Algorithm ......................................................... 44
            5.3.3.2 Fuzzification of Input Parameters ....................... 44
                5.3.3.2.1 Pseudo Code ........................................... 46
            5.3.3.3 Calculation of Fuzzy Depth ................................. 46

VI. SIMULATION MODULE – IMPLEMENTATION ................................ 49
    6.1 Overview ........................................................................... 49
    6.2 Key Definitions .............................................................. 49
    6.3 Two Dimensional Node Splitting and Cloning Algorithm ....... 50
        6.3.1 Construction of Node Cell Matrix ............................... 51
8.2 Fuzzy Logic Testing Results
8.3 Performance Testing
8.4 Screen Shots

IX CONCLUSION AND FUTURE WORK

9.1 Conclusion
9.2 Future Work

BIBLIOGRAPHY
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Block Diagram of Virtual Surgery System</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Use case diagram of virtual surgery system</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Grid structure – cells and nodes</td>
<td>9</td>
</tr>
<tr>
<td>2.4 Structure of individual cell</td>
<td>9</td>
</tr>
<tr>
<td>2.5 Data Acquisition Module</td>
<td>11</td>
</tr>
<tr>
<td>2.6 Significance of recording the cut history</td>
<td>12</td>
</tr>
<tr>
<td>2.7 Node Splitting</td>
<td>12</td>
</tr>
<tr>
<td>2.8 A block diagram of Virtual Surgery Fuzzy Control System</td>
<td>14</td>
</tr>
<tr>
<td>2.9 Membership functions Force and Stiffness</td>
<td>15</td>
</tr>
<tr>
<td>2.10 Fuzzy Rule Base</td>
<td>16</td>
</tr>
<tr>
<td>2.11 Block Diagram Surgical Animation Loop</td>
<td>19</td>
</tr>
<tr>
<td>3.1 Cell structure after splitting node 2 into node 2 and node 5</td>
<td>21</td>
</tr>
<tr>
<td>3.2 Structure of boundary list</td>
<td>22</td>
</tr>
<tr>
<td>4.1 Block diagram of multi threaded application</td>
<td>27</td>
</tr>
<tr>
<td>4.2 Working of Data Acquisition Module</td>
<td>28</td>
</tr>
<tr>
<td>4.3 Truth table for critical section access</td>
<td>29</td>
</tr>
<tr>
<td>4.4 Working of two way handshake protocol</td>
<td>31</td>
</tr>
</tbody>
</table>
8.3 Surgical operations – tilted view

8.4 Surgical operations – wireframe display
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Results from fuzzy control system for a uniform grid</td>
<td>74</td>
</tr>
<tr>
<td>8.2 Results from fuzzy control system for a non-uniform grid</td>
<td>74</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

1.1 Importance of Virtual Reality in Surgery

In today’s world virtual reality is gaining importance in all major fields [1-2,5-12,15]. Virtual simulations are being conducted in both design and manufacturing phase, as they are cost effective and less time consuming. Virtual reality has spread its wings from designing of automobile parts to tracking of space shuttle’s operations. Due to the increasing computing capability, the potential of applying virtual reality in medicine and surgery is increasing day by day. Many applications have been developed for functional anatomy education in medicine. Researchers have been conducting studies in virtual surgery simulations (or simply virtual surgery), where computer technologies are used to simulate the surgical process in virtual environments. Virtual surgery simulations can be used to train medical students in a cost effective manner, and also can be used by surgeons to explore optimal surgical solutions.

1.2 Background

Many research studies have been conducted on this subject in the past. Below is a list of some well known research studies in this field. Research work has been done, in the Visualization Research Lab in Brown University and HIT (Human Interface
Technology) Lab in Washington University [1], on visualizing various anatomical structures and on virtual surgery simulation. Current research involves developing applications which enable learning of human anatomy by virtual visualization of various anatomical and functional parts of the human body, pre-operative training of surgeons, preparation for surgical intervention using virtually visualized objects, scientific visualization of medical data to some geometrical and visual form for better representation and inferences. The simulations for virtual bone cutting have also been implemented.

INRIA- a leading French national research institute has simulated the action of virtual laparoscopic surgical instruments for deforming and cutting three-dimensional anatomical models. Ultrasound imaging simulators are used for surgical simulations.

The simulation of deformable tissue has been implemented using the spring-surface mesh model. But the model is intended to be physically based and less interactive to support dynamic alteration such as incisions, under several constraints: mechanical realism, robustness and real time.

Song and Reddy [12] from Human Interface Lab at The University of Akron proposed to use the Finite Element Method (FEM) to model soft tissue cutting in 1996. Since then much research work has been conducted using the method. UC Berkeley’s Medical Robotic Lab developed a real-time finite element model for soft tissue behavior. They use non-linear FEM with an approach called Dynamic Progressive Meshes (DPM) which refines the mesh locally, where necessary such as where the instrument is in contact with the tissue, at runtime. The HIT Lab in Washington University [1] developed
an algorithm to solve tissue cutting finite element models. They have achieved real time simulation along pre-specified cutting paths.

1.3 Need for Further Study and Outline of the Thesis

Though lots of studies have been done, there are many unresolved problems still prevailing in this subject. Some of the unresolved issues are

- modeling of the deformation of soft tissue accurately and in real time,
- real time simulations of virtual surgery without pre-specifying surgical paths,
- And development of a highly interactive virtual surgery simulation system.

The purpose of this research is to develop a virtual space which can be used to perform surgical operations in real time. This study explores the various modules that integrate to form a virtual surgery system.

This study explores the novel approach of incorporating fuzzy logic control system [3] into virtual surgery. By incorporating fuzzy logic the processing time needed to simulate tissue cutting can be greatly reduced. This allows virtual surgical operations to be performed in real time without the need to pre-specify the cutting paths.

Visualization is an indispensable part of virtual surgery. The objective of visualization is to realistically represent the surgical tools, surgery objects, and their interactions to the surgeons in a virtual environment. For virtual surgery, structural and textural data visualization is most important for generating realistic rendering of the surgery process. This study also explores a novel method to render structural and textural
changes in soft tissue during a virtual surgery process. A node splitting and cloning algorithm has been developed for this purpose.

A multi threaded approach has been developed to speed up the simulation process. This isolates the data acquisition system from the simulation system so that both modules can be synchronized without any speed penalty. This also produces a more interactive system from the users’ perspective.

The following chapters will walk through the overall study process and the results that have been produced from the study. The next chapter describes the overall system design of the virtual surgery system. Chapters III – VII discuss the implementation details of the modules and subsystems. Chapter VIII shows some testing results from the system. The last chapter concludes the project and lists future work can be done to the system.
2.1 System Components

The virtual surgery system is designed in a modular fashion. Modules have been divided based on their functionality. Each module performs a particular core functionality. The virtual surgery system is composed of five core modules.

- Model Definition Module
- Data Acquisition Module
- Simulation Module
- Fuzzy Control System
- Surgical Animation Module

![Figure 2.1 Block Diagram of Virtual Surgery System](image-url)
2.1.1 Model Definition Module

The model definition module defines the geometry and topology of surgical objects. It interacts with the surgical animation module to initialize the data structures that hold the grid information. It defines the composition of each node and its geometry. The model definition module starts the virtual surgery process by initializing all the required data structures.

2.1.2 Data Acquisition Module

The core functionality of this module is to acquire data from the cutting tool. This module acts as an interface between the user and the virtual surgery system. The module acquires the force exerted by the user and the angle of tool. This module works in a multi threaded fashion. This helps in synchronizing both data acquisition and virtual surgery simulation process. The data acquisition module employs a median filter to blur and normalize the inputs acquired from the user.

2.1.3 Simulation Module

The simulation module acts as the controller of the whole surgical process. It interacts with the data acquisition module for data acquisition. It then feeds the data to the fuzzy control system to get the depth of cut. It then controls the surgical animation module to give the visual feed back to the user. The simulation module controls the node splitting process and also holds the cut history. It incorporates special data structures and algorithms for this purpose.
2.1.4 Surgical Animation Module

The surgical animation module provides the visual feedback for the surgical process. The surgical animation module communicates with the simulation module to get the current changes in the node data and updates the visual display. The core functionality of this module is to provide realistic rendering of surgical tools and to provide visual feedback of the surgical process. The module manages the whole grid data.

2.2 Design

The design of the virtual surgery system deals with the working of core modules and their co-operative operations. The synchronization of the core modules forms the key to the real-time surgical simulation process. Various synchronization mechanisms have been adopted to make this system work in real time. The Unified Modeling Language shows the system design from the user’s perspective with a use case diagram.
2.2.1 Model Definition Module

The model definition module helps to initialize the surgical object. It defines grid cells and the nodes associated with each cell. Each cell has three nodes associated with it. The initialization process helps the surgical animation module in constructing the node and cell information data structures to start the surgical animation process.
The model definition module also defines the structure for surface generation. The surface generation is used for generating a 3D display. Each edge of the surface has an id associated with it. The module keeps track of the boundary nodes and the edges with which they are associated. It manages the data structure used to store the boundary nodes and provides methods to add or remove a boundary node from the data structure.

The surface generation process in the surgical animation module interacts with the model definition module to retrieve the current boundary node and performs the surface
generation process. This module allows the user the flexibility to construct a custom model definition by specifying the number of rows and columns in the structure.

2.2.1.1 Functionality

- Generating node data
- Generating cell data
- Initializing boundary node data

2.2.2 Data Acquisition Module

The data acquisition module interacts with the surgical tool to acquire data. The surgical tool acts as a user interface to acquire the force applied and the angle of the tool at any given time. The data acquisition module performs multi threaded processing to enable data acquisition and surgical simulation simultaneously. It adopts the thread safe paradigm in order to facilitate multi threaded processing. The data acquisition module performs thread synchronization with the help of a two process producer/consumer algorithm [18]. The sampling rate specifies the rate at which the data is being acquired from the instrumented cutting tool or scalpel. The data acquisition module employs a median filter to blur the force acquired from the tool. This helps in producing a uniform force parameter.

2.2.2.1 Functionality

- Interfacing cutting tool and surgical system.
- Acquiring data from the tool.
- Filtering input data.
- Synchronizing data acquisition and surgical simulation.
2.2.3 Simulation Module

The simulation module acts as a core controller of the entire surgical simulation process. It initializes the surgical simulation process by invoking the other modules. The simulation module forms the heart of surgical simulation process. It creates the objects required for the simulation process such as grid, data storage objects, and cutting tool. It provides methods to initialize and animate the display, animate the cutting tool, enabling the inter-actor etc. The simulation module records the cut history by storing the pointers to the grid nodes storage structure. The location of the nodes is adjusted for each step.
The simulation module encapsulates the algorithms to perform the node splitting. It determines the next destination node and splits the node by adding a new node. The process of generating a new node is called cloning. The module then re-arranges the node-cell associations. This is then repeated for each cutting step. A node splitting and cloning algorithm has been developed for this purpose. The node re-arrangement is achieved by using the node-cell relational matrix.
The Simulation module initializes the fuzzy control system by defining the membership functions and the fuzzy rule base. The simulation module then interacts with the fuzzy control system for each cutting step to determine the depth of cut. The final phase is to update the node and cell information data structure and feed it into the visual animation module to produce a realistic rendering of the surgical process. As said before the simulation module forms the heart of surgical system.

2.2.3.1 Functionality

- Recording the cut history
- Locating current cutting node
- Tracking boundary nodes
- Node splitting and cloning algorithm
- Initializing fuzzy control system
- Initializing surgical animation

2.2.4 Fuzzy Control System

The fuzzy control system controls the cutting process in surgical simulation. It takes as input the force and stiffness to calculate the depth of cut at each time step. The control system uses fuzzy logic to regulate cutting process. It defines the membership functions
Figure 2.8 A block diagram of Virtual Surgery Fuzzy Control System.

- **Membership Value**

  The membership value determines the degree of membership of a number in a fuzzy set. The apex number always has a membership value of 1 and the base numbers always have a membership value of 0.

- **Identifying the Inputs and Outputs**

  The aim of fuzzy control system is to regulate the depth of cut at any given instant of time. The depth of cut depends on the force applied by the user on the cutting tool and on the material property of the object. So inputs to the fuzzy control system would be force applied by the user and stiffness of the material. The output of the fuzzy control system would be the depth of cut.

- **Defining Membership Functions**

  Fuzzy control systems are modeled based on expert experience of real people. The membership functions for inputs and outputs have to be defined by expert experience. The membership function defines the fuzzy sets in each input and output
variables. In a virtual surgery system, membership function has been defined for input variables (force and stiffness) and output variable (depth). Defining membership function is not a one step process and needs expert experience to evaluate.

Figure 2.9 Membership functions for force (a) and stiffness (b).
• Constructing the Fuzzy Rule Base

The fuzzy rule base transforms the input functions into outputs. In order to construct the rule base a lookup table is defined. The lookup table defines the corresponding action to be taken for each combination of input fuzzy sets. In order to achieve this, a two dimensional table is constructed. The input functions are represented in x and y axes. The table lists the output fuzzy set for each combination of input fuzzy sets. In virtual surgery systems the input function force is represented in x axis and stiffness in y axis. The lookup table for the virtual surgery fuzzy control system is given below.

<table>
<thead>
<tr>
<th>Force</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
<th>Very Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>No Cut</td>
<td>Small</td>
<td>Medium</td>
<td>Large</td>
</tr>
<tr>
<td>Medium</td>
<td>No Cut</td>
<td>No Cut</td>
<td>Small</td>
<td>Medium</td>
</tr>
<tr>
<td>Large</td>
<td>No Cut</td>
<td>No Cut</td>
<td>No Cut</td>
<td>Small</td>
</tr>
<tr>
<td>Very Large</td>
<td>No Cut</td>
<td>No Cut</td>
<td>No Cut</td>
<td>No Cut</td>
</tr>
</tbody>
</table>

Figure 2.10 Fuzzy Rule Base
The rules are written in “AS THEN” format. The rule base is then used to get the outputs. “AS THEN” rule format makes the control system work fuzzy. Some of the rules of virtual surgery system are given below:

As the membership function for medium force is x
And the membership function for small stiffness is x,
Then the output function for small depth gets fired partially.

As the membership function for large force is x
And the membership function for small stiffness is x,
Then the output function for medium depth gets fired partially.

- Applying Fuzzy Math

The membership value of each fuzzy set in the input functions is determined. This is calculated from the membership function definition. For each combination of input membership values, the conjunction fuzzy set is determined based on fuzzy math. This is because the rules are written in AND format. For example

Membership value of medium force = 0.7
Membership value of small stiffness = 0.5

This combination of input fuzzy set fires the rule
As the membership function for medium force is \( x \)
And the membership function for small stiffness is \( x \),
Then the output function for small depth gets fired partially.

The membership value of the output function for this combination of input fuzzy set is determined by

\[ 0.7^0.5 = 0.5 \]

2.2.4.1 Functionality

- Provides API’s for incorporating fuzzy control system in other applications.
- Provides ability to define membership functions for fuzzy control system based on the application needs.
- Allows applications to define a custom fuzzy rule base.
- Calculates depth of cut based on force and stiffness inputs.

2.2.5 Surgical Animation Module

The surgical animation module is designed to provide visual feedback to the surgical simulation process. The surgical animation module acts as an interface between the virtual surgery system and the visualization tool kit system [19]. It provides API’s to the virtual surgery system and then communicates with visualization tool kit to get the visual feedback. It implements data structures for fast storage and retrieval of simulation data. It acts as a wrapper around the visualization tool kit by providing virtual surgery system with API’s to create visual feedback. The module incorporates a class library.
Each class in the library is designed to provide certain functionality to the virtual surgery system.

2.2.5.1 Functionality

- Algorithms for fast retrieval and storage of data.
- Texture mapping functionalities
- Real time animation of surgical objects
- Construction of dimensional surface
- Construction of surgical cutting tool.

The implementation details of the virtual surgery simulation system explains in detail the data structures and algorithms used, does a complexity analysis and also reasons out the necessity for using these data structures. The implementation details of each module are discussed in the following chapters.
3.1 Overview

The model definition module uses various data structures to define and keep track of grid geometry and topology. It keeps track of the boundary nodes which are helpful in 3D animation. The boundary nodes are stored in a linked list. Each boundary node has a node id, and an edge id.

3.2 Key Definitions

- **Boundary list** – Represents the data structure used to store nodes on each edge of the boundary. A doubly linked list is used to define the boundary list.

- **Edge ID** - It identifies the edge to which the node belongs. The edge id becomes significant in node splitting. The splitting algorithm uses the edge id to re-generate three dimensional surfaces.

- **Node ID** – The node id identifies individual nodes in the grid. This acts as a key to the node information data structure in the surgical animation module. This helps in accessing the location of each node, the cell to which it belongs and its scalar properties.
- Fast traversal pointer – The fast traversal pointer points to the updated node at which the next node splitting would occur. This reduces the cost of traversing the linked list at each node splitting time step.

3.3 Data Structures and Algorithms

This section illustrates the data structures and algorithms used in the model definition module.

3.3.1 Boundary Node Storage Structure

The model definition module uses a doubly linked list representation to keep track of boundary nodes. The linked list stores a pointer to structure VSBoundPoints and a pointer to VSPoinData.

- VSBoundPoints – This structure contains the node id, edge id, pointer to next boundary node, and a pointer to the previous boundary node. The node id acts as a key to access VSPoinData.

- VSPoinData – This data structure stores the information about each node.

Figure 3.1 Cell structure after splitting node 2 into node 2 and node 5
Figure 3.1 shows the state of grid cells after cutting of node 2. The node 2 gets split into node 2 and node 5. The boundary node storage structure has to be updated to reflect the changes in grid structure.

Figure 3.2 (a) Structure of boundary list before splitting the node #2

Figure 3.2 (b) Structure of boundary lists after splitting the node #2

The figure above shows the state of boundary node storage structure before and after splitting node 2. Boundary point storage structure is modified by adding the current destination node and the new node that has been added to the grid structure.
3.3.1.1 Pseudo Code

Initialize boundary point storage structure with initial parent nodes – from cut history.

Initialize fast traversal pointer to point to the current cutting node.

For each cutting time step
{
    Insert destination node to boundary storage structure – after the node that fast traversal pointer points to.
    Insert the new node added to the grid structure – after the destination node
    Update links between inserted nodes.
    Update the fast traversal pointer to point to current cutting node.
}

3.3.1.2 Complexity Analysis

- Traversal Complexity

In order to traverse the doubly linked list the complexity is $O(n)$. This traversal has to be done at each time step during the cutting process. With the help of a fast traversal pointer the complexity to traverse the list is reduced significantly. Fast traversal pointer always points to the current cutting boundary node. During the cutting process it updates itself by adding the associated boundary nodes and re-locating to the updated cutting node.

- Memory Requirements

This structure helps to store the node information at a centralized location without replicating the information. This reduces the memory requirement of the whole system.
Further a separate pointer to the actual location of node information avoids repeated traversal of the centralized data structure.
CHAPTER IV
DATA ACQUISITION MODULE – IMPLEMENTATION

4.1 Overview

The data acquisition module uses thread safe classes to acquire data from the user. The data acquisition module incorporates a producer consumer algorithm [18] to solve the critical section problem. Semaphores are used in the producer consumer algorithm to allow access to the critical section. The data acquisition module filters the acquired data before feeding it into surgical animation system. A median filter is designed for this purpose. The module uses a two way handshake protocol to stop the surgical animation process. The two way handshake protocol allows the synchronization of the multi-threaded application.

4.2 Key definitions

- Thread – Thread is a control of execution. It is also called as light weight process since it shares all resources. It uses a concept called Copy On Write [18].

- Multi threading – In multi threading more than one thread shares the processor in a time sharing fashion. Multi threading makes an illusion to the end user. The end user feels like all the threads are executing simultaneously. Two types of multi threading are preemptive and non preemptive [18].
• Critical section – A section of code that has the data shared [18]. Since multiple threads have access to the shared data it necessary to secure the data from read write conflicts. The concept of avoiding read write conflicts by allowing one process to write at any given time is called mutual exclusion principle.

• Semaphores – Semaphores are counters that hold the number of resources currently available in the system [18]. When a thread enters the critical section it decrements the semaphore. When the thread exits the critical section it increments the semaphore.

• Two way handshake protocol – A procedure used to synchronize threads so that operations can be performed in an orderly fashion.

• Median filter – Filters are used to smooth inputs feed into the system by removing aberrations. Filters become significant when input quantities acquired differ significantly. Filters use different kinds of masks to define the method of filtering. A median filter uses a median neighborhood mask.

4.3 Data structures and algorithms

This session illustrates the use of various data structures and algorithms that have been used in the data acquisition module.

4.3.1 Producer consumer algorithm

A producer consumer algorithm is used to synchronize data access between two threads [18]. One thread produces the data while the other consumes data. The virtual
surgery system has two threads to perform surgical animation. The data acquisition thread acquires data from the user. The data processing thread uses the data acquired by the data acquisition thread to perform surgical animation. A producer consumer algorithm is used to synchronize the data access between these two threads.

Figure 4.1 Block diagram of multi threaded application

The block diagram illustrates the working of multi threaded virtual surgery system. The data processing thread accesses the shared data critical section to update input values. The data processing thread accesses the shared data critical section to access the updated input value.
The block diagram describes the working of the data acquisition thread. The data acquisition thread waits for the timer control to fire. Once the timer control fires the data acquisition thread acquires data from the instrumented surgical tool (scalpel). The data acquisition thread then waits for critical section access. Once the data acquisition thread gets the critical section access it updates the input data. The cycle again repeats until the user tries to stop the simulation process.

4.3.1.1 Working

Each thread assumes that the other thread is in the critical section, processing the data. This is achieved by making the value of Turn to be that of the other process. Each thread which wants to enter the critical section sets its Flag to be true. When both data acquisition thread and data processing thread try to enter the critical section, each thread
sets its Flag to be true. Then the value of Turn determines the thread that enters critical section.

- Truth Table

<table>
<thead>
<tr>
<th>Turn</th>
<th>Flag[DAT]</th>
<th>Flag[DPT]</th>
<th>Critical Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAT</td>
<td>T</td>
<td>T</td>
<td>DAT</td>
</tr>
<tr>
<td>DPT</td>
<td>T</td>
<td>T</td>
<td>DPT</td>
</tr>
</tbody>
</table>

Figure 4.3 Truth table for critical section access

The truth table illustrates how the value of turn determines the access to critical section when both data acquisition thread and data processing thread try to access the critical section at the same time. When data acquisition thread tries to access the critical section it makes its flag value to true. It assumes the turn to be data processing thread and assigns DPT to turn. When data processing thread tries to access the critical section, at the same time, it sets its flag value to be true and assumes the turn to data acquisition thread and assigns DAT. Now the turn value becomes DAT and data acquisition thread gets access to the critical section.

4.3.1.2 Pseudo Code [18]

**Data Acquisition Thread**

Loop
   Acquire data from surgical tool
Flag [Data Acquisition Thread] = true
Turn = Data Processing Thread

While (Flag [Data Processing Thread] and Turn = = 1)
{
    Data Processing Thread is reading data
    Do No Operation
}

Enter Critical Section
    Update Data
Exit Critical Section
Flag [Data Acquisition Thread] = false
Until False

**Data Processing Thread**

Loop
    Flag [Data Processing Thread] = true
    Turn = 0

    While (Flag[Data Acquisition Thread] and turn = 0)
        Data Acquisition Thread is updating data
        Do No Operation
    
Enter Critical Section
    Read data from critical section
Exit Critical Section
Flag [Data Processing Thread] = false
Do the surgical simulation
Until False
4.3.2 Two Way Hand Shake Protocol

The two way hand shake protocol synchronizes data acquisition thread and data processing thread. This is used to make sure that data processing thread processes all of its data before shutting down the system. A two way hand shake protocol also makes sure that data processing thread is terminated before terminating the data acquisition thread.

When there is no more data available and the user is ready to shut down the surgical system, the data acquisition thread initializes two way hand shake protocol by placing a “Terminate” message in the critical section. When data processing thread sees this message it completes all jobs, places a “Terminated” message in the critical section, and terminates its thread. Data processing thread, on receiving this message, terminates the thread and stops the simulation process.

![Diagram of two way handshake protocol]

Figure 4.4 Working of two way handshake protocol
4.3.2.1 Pseudo Code

**Data Acquisition Thread**

Write Terminate message in critical section.

Repeat

    Check the critical section for Terminated message

Until (TRUE)

Stop surgical simulation and exit

**Data Processing Thread**

Repeat

    If (Terminate Message)

        Do the clean up work

        Write Terminated message in critical section

        Break the loop

    Else

        Do the simulation process

Until (FALSE)

Terminate the thread.

4.3.3 Median Filtering

Median filtering is used to blur inputs fed to surgical system by defining a mask based on the neighborhood. The median filter takes the median of all inputs in the neighborhood, as input to the system. In order to get the median of given input values, the values in the input list have to be sorted. A linked list is used to hold the inputs. The
linked list acts as the mask. Each input obtained from the surgical tool is time stamped. When a new input is obtained from the surgical tool, the node which has the earliest time stamp is deleted and the new input is compared with every other input value in the list to determine a suitable location for the new input.

![Figure 4.5 Median filter](image)

The figure above shows the structure of the median filter mask. Each node represents an input value which is time stamped at the time of acquisition. The next value to be replaced represents the node with earliest time stamp value. This implies that the node represents the input value that arrived before all other input values currently in the mask.

4.3.3.1 Pseudo Code

1. Time stamp the new input value
2. Initialize TraversePointer := First node of the input list
3. Initialize MinTimeStamp := First node’s time stamp
4. DeletePointer := TraversePointer
Repeat

    If( TraversePointer->TimeStamp < MinTimeStamp)
    {
        Assign MinTimeStamp := TraversePointer->TimeStamp
        DeletePointer := TraversePointer;
    }

    Assign TraversePointer := TraversePointer->Next

Until (TraversePointer = = NULL)

Make the previous node of DeletePointer point to the next node of DeletePointer

Delete the node that is pointed to by DeletePointer

Assign TraversePointer := First node of input list

Repeat

    If(TraversalPointer->InputValue < InputValue)
        TraversePointer := TraversePointer->Next
    Else
        Break the loop

Until (TraversalPointer = = NULL)

Create a new node and assign new input value to it

Make TraversePointer’s previous node point to new node

Make new node point to TraversePointer

Find the median of the sorted list of input values
4.3.3.2 Formula

\[
\text{Median} = \begin{cases} 
\text{Value of } (n+1/2)\text{th term} - n \text{ is odd} \\
\frac{\text{Value of } (n/2)\text{th term} + \text{Value of } ((n/2)+1)\text{th term}}{2} - n \text{ is even}
\end{cases}
\]

4.3.3.3 Complexity Analysis

The complexity of median filtering can be divided into three parts:

- Complexity to fill the node list

  Initially when the simulation process starts, the node list for the median filter is considered to be empty. If \( n \) is the length of the node list for the median filter then, in order to fill the node list \( n \) inputs have to be obtained from the user. Complexity to fill the node list initially becomes \( O(n) \).

- Complexity to find the node to be replaced

  Each input obtained from the user is time stamped. This time stamp is used to find the node that has to be replaced at a particular time step. In order to find the node with the earliest time stamp, the entire node list has to be traversed. Time complexity to traverse the entire list of length \( n \) is \( O(n) \).

- Complexity to sort the list

  In order to find the median the list has to be sorted. The sorting of the list, whenever a new input arrives, is done by comparing the new input with every other input
value in the list and then finding a right location to insert the new input. The time complexity to traverse the entire list of length n is O(n). The total complexity for sorting the list O(n^2).
CHAPTER V
FUZZY CONTROL SYSTEM – IMPLEMENTATION

5.1 Overview

The fuzzy control system helps to control the surgical animation process. It calculates the depth of cut based on the force applied at a particular location and the stiffness of the node at that particular location. This section explains the working of fuzzy control system and its functionalities.

The working fuzzy control system can be divided into two categories.

• Membership definition
  The membership definition section deals with defining the membership functions for each input parameter in the fuzzy control system, and defining a lookup table to determine the rules that need to be fired.

• Gauging the depth of cut
  Gauging the depth of cut section deals with the runtime measurement of depth with actual inputs. This section uses the membership functions and lookup table defined in the membership definition section to calculate the actual depth of cut.
5.2 Key Definitions

- Fuzzy logic control system – A fuzzy logic based control system used to control the surgical cutting process based on the force exerted by the user and the stiffness of the tissue at that location.

- Membership function – A membership function is defined for each input parameter. It quantifies the partial memberships of the input value in each fuzzy region [20].

- Normalization – Normalization of input value converts the actual input into some normalized form. In this implementation the input value is converted to a normal form that ranges from 0 to 1.

- Fuzzification – Fuzzification is a process of converting crisp input values to fuzzy values [3-4, 13, 20].

- Defuzzification – Defuzzification is a process of converting fuzzy values into crisp values. This implementation uses centroid defuzzification mechanism to convert fuzzy output values to crisp output values [13, 20].

- Membership lookup table – Membership lookup table defines output rules that get fired with each combination of input parameters. This lookup table is defined using AS and THEN rules.

5.3 Data Structures and Algorithms

This section explains the data structures and algorithms used in the fuzzy logic control system. The fuzzy logic control system makes use of data structures to define
membership functions for input parameters and for defining the lookup table. The fuzzy logic control system uses a fuzzification algorithm to convert crisp value into fuzzy value and a centroid defuzzification algorithm [14] to convert a fuzzy value into a crisp value.

5.3.1 Defining Membership Functions

In the fuzzy logic control system membership functions have to be defined in order to make use of the rule base. A membership function quantifies the partial membership values of each input value.

5.3.1.1 Characteristics of Membership Functions

- Each membership function has a name
- Each membership function is divided into several fuzzy regions. Each fuzzy region is given a name.
- A fuzzy region can be of any type. This implementation uses triangular fuzzy regions.

5.3.1.2 Steps to Define a Membership Function

- Assign a name to the membership function
- Divide the membership function into several fuzzy regions.
- Assign a name to each fuzzy region.
- Define each fuzzy region by specifying the start point, end point and divider point. This is used to specify a triangular fuzzy region. The start and end points specify the beginning and ending locations of a particular fuzzy region. The divider point
specifies the peak point of the triangular region. The membership value at start and end points are 0. The membership value at the divider point is 1. The figure below illustrates the triangular function for the fuzzy region medium.

![Figure 5.1 Structure of fuzzy regions](image)

VSMembershipFunction makes up the structure that holds the definition of each membership function. The definitions of all membership functions are stored in a linked list. Each node of the list has a pointer to VSMembershipFunction. Each membership function is identified by its name.
The block diagram above explains the data structure which holds the definitions of all membership functions. The top level is a linked list. Each node of the linked list points to a structure that holds the membership function definition. Each membership function definition is identified by its name. The structure also holds the number of regions in that membership function. It allocates a dynamic array based on the number of regions in that membership function. Each array node has a pointer to a structure holding the values of each region definition. Each region is identified by the region name. Each region has a

**Figure 5.2 Data structure storing membership functions**
start point, end point and a divider point. Each region has two edges. The slopes and constant values of the two edges are calculated based on the values of start, divider and end points.

5.3.1.3 Formula

Slope of first edge \( m_1 = \frac{Y(\text{Start}) - Y(\text{Divider})}{X(\text{Start}) - X(\text{Divider})} \)

\( Y(\text{Divider}) - Y(\text{End}) \)

Slope of second edge \( m_2 = \frac{Y(\text{Divider}) - Y(\text{End})}{X(\text{Divider}) - X(\text{End})} \)

Constant of first edge \( c_1 = Y(\text{Start}) - (m_1 \times X(\text{Start})) \)

Constant of second edge \( c_2 = Y(\text{End}) - (m_2 \times X(\text{End})) \)

\( Y(\text{Start}) \) – Y co-ordinate of start point

\( Y(\text{End}) \) – Y co-ordinate of end point

\( Y(\text{Divider}) \) – Y co-ordinate of divider point

\( X(\text{Start}) \) – X co-ordinate of start point

\( X(\text{End}) \) – X co-ordinate of end point

\( X(\text{Divider}) \) – X co-ordinate of divider point
5.3.2 Defining the Rule Base

The rule base is written using AS and THEN rules. For each combination of input function’s fuzzy regions, the rule base holds the fuzzy region of the output function that gets fired. For example let force and stiffness be the input functions. Let the fuzzy region for force be medium and the fuzzy region for stiffness be small. The AS THEN rule for this combination of input function is

As the membership function for medium force is x

And the membership function for small stiffness is x,

Then the output function for small depth gets fired partially.

The lookup table for the rule base holds these values. The structure of the lookup table is shown in the diagram below.

<table>
<thead>
<tr>
<th>Region for Force</th>
<th>Region for Stiffness</th>
<th>Region for Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>Small</td>
<td>No Cut</td>
</tr>
<tr>
<td>Medium</td>
<td>Small</td>
<td>Small</td>
</tr>
<tr>
<td>Large</td>
<td>Small</td>
<td>Medium</td>
</tr>
<tr>
<td>Very Large</td>
<td>Small</td>
<td>Large</td>
</tr>
</tbody>
</table>

Figure 5.3 Data structure for fuzzy control system
5.3.3 Gauging the Depth of Cut

After all the membership functions have been defined and the lookup table initialized the fuzzy control system is all set to start its operation. The operation of the fuzzy control system is divided into three steps (steps 2-4 below).

5.3.3.1 Algorithm

**step #1** Get crisp input values

**step #2** Convert input parameters into fuzzy values

**step #3** Calculate the fuzzy depth

**step #4** Use centroid defuzzification to convert the fuzzy depth into a crisp value

**step #5** Return the value of depth

5.3.3.2 Fuzzification of Input Parameters

The fuzzification of input parameters deals with converting the crisp input parameter value into a fuzzy value. For each value of the input parameter the membership value for each fuzzy region is determined [3-4, 13]. The diagram given below illustrates the procedure to determine the membership values.
Let the value of normalized force be 0.7. This force is represented in x-axis. In order to quantify the membership values in each fuzzy region, the region for which the force parameter belongs is determined. This is represented in the diagram by drawing a vertical line and finding the intersection of fuzzy region edges. In the diagram the vertical line meets edge #2 of medium fuzzy region and edge #1 of large fuzzy region.

Programmatically this is computed with the formula

\[
\text{Membership Value} = (m \times \text{Input Parameter Value}) + c
\]

\(m\) – Slope of the edge in that particular fuzzy region

\(c\) – Constant for the edge in that particular fuzzy region

The values of \(m\) and \(c\) are computed when defining the membership functions. The value is computed for each edge in a particular fuzzy region. The procedure is then repeated for each fuzzy region. The results are tabulated.
### 5.3.3.2.1 Pseudo Code

For each fuzzy region

- Calculate the membership value of first edge
- Calculate the membership value of second edge
- Find the membership value of that fuzzy region
- Tabulate the results

### 5.3.3.3 Calculation of Fuzzy Depth

The fuzzy depth is calculated with the tabulated results obtained from fuzzification process. Each fuzzy region, in input membership function one is combined with every fuzzy region in input membership function two. The fuzzy region for the output membership function is determined from the lookup table for that particular combination. The extent up to which the fuzzy region is fired is determined from the membership values of the input parameters.
For each combination of input fuzzy regions the fuzzy logic control traverses the lookup table to find the output fuzzy region. For example the combination of medium force and small stiffness results in the small output fuzzy region from the lookup table. The extent up to which small output fuzzy region is fired is determined as follows:

Membership value for fuzzy region medium (Force) – 0.2
Membership value for fuzzy region small (Stiffness) – 0.3
Output membership value for fuzzy region small (Depth) – 0.2 $\times$ 0.3 = 0.2

The cumulative membership value for each region is found by

**Cumulative membership value of small fuzzy region (output membership function)**

$= \text{membership value for small in combination #1} \times \text{membership value for small in combination #2} \times \ldots$

Centroid Defuzzification

The centroid defuzzification process determines the center of gravity of the resultant structure which forms the crisp output value [14]. For each fuzzy set the
membership value of the fuzzy set is used to chop off the remaining part in the triangular structure. The center of gravity of the resultant structure is found out. The output value is given by the expression

\[
\text{Membership [1]} \times \text{Center [1]} + \text{Membership [2]} \times \text{Center [2]} \ldots
\]

Crisp output value = 

\[
\text{Membership [1]} + \text{Membership [2]} \ldots
\]

For each membership function the fuzzification process determines the membership values of each fuzzy region in that membership function. If there are \(n\) membership functions and \(n\) fuzzy regions in each membership function, the time complexity to find the membership values is \(O(n^2)\).

To calculate the fuzzy depth all the combinations of fuzzy regions from different membership functions need to be considered. If there are ‘\(x\)’ number of membership functions and each membership function has ‘\(y\)’ number of fuzzy regions, then \(x^y\) combinations are possible. For each combination in the lookup table the partial membership value of output fuzzy region is calculated.

If,

\[
\text{Complexity to traverse through each combination in the lookup table} = O(x^y)
\]

Cumulative complexity = \(O(n^2) + O(x^y) = O(x^y)\)
6.1 Overview

The simulation module uses the node splitting algorithm to perform the cutting process. This section illustrates the cutting process using the two dimensional node splitting algorithm. This algorithm uses the cut history information to get the reference points for node ordering.

6.2 Key Definitions

- Node Splitting – Node splitting refers to the procedure followed to perform the cutting process in virtual surgical simulation. The algorithm used for this purpose is called node splitting algorithm.

- Node Cell Matrix – Node cell matrix is a matrix used by the node splitting algorithm to perform the node ordering process.

- Node Ordering – Node ordering is a procedure followed to get the order of neighboring nodes at the current cutting location.

- Current Node – The current node refers to the node at which the knife is currently located.
• Destination Node – The destination node refers to the node where the knife is going to be placed after the cutting process.

• Cloning – Cloning is a process of adding a new node with the same properties as the existing node for performing the node splitting process.

• Reference Nodes – Reference nodes refers to the nodes used as reference in node ordering process.

• Knife angle – The knife angle refers to the angle of knife relative to the X – axis.

• Reference Angle – The reference angle refers to the angle of the node relative to the current cutting node.

• Cutting Angle – The cutting angle refers to direction in which the knife has to travel to reach the next destination node.

• OrderedNodeArray – This array holds the neighboring nodes in an ordered fashion from start reference node to the end reference node.

• OrderedCellArray – This array holds the cells associated with the current cutting node in an order based on the OrderedNode array.

6.3 Two Dimensional Node Splitting and Cloning Algorithm

The 2D node splitting and cloning algorithm is used in the virtual surgery simulation process to perform node cutting. The 2D node splitting and cloning algorithm consists of the following steps

• Construction of Node Cell Matrix

• Determining the destination node

• Cloning Process
6.3.1 Construction of Node Cell Matrix

This process starts by determining the neighboring nodes for the current cutting node. The neighboring nodes of a node are obtained from the node information data structure in the visual animation module. The node information data structure stores each node’s information, identified by its node id, and the cells to which the node is associated. This gives the neighboring cells for the current cutting node. These cell ids are listed in the Y axis of the node cell matrix. For each of the neighboring cells find the nodes associated with that cell from the cell information data structure. The cell information data structure stores each cell’s information, identified by its cell id, and the nodes by which the cell is constructed. Get the neighboring nodes of the current cutting node and list it the X – axis of node cell matrix. Find the relationship between each node and cell in node cell matrix. Assign 1 if there is a relationship between a node and a cell. Else assign infinity.

![Figure 6.1 Structure of grid before node splitting](image-url)
Figure 6.2 Structure of node cell matrix

6.3.1.1 Pseudo Code

Get the cells associated with current node

Update the node cell matrix with the neighboring cells

NumberOfRows := Number of neighboring cells

NumberOfColumns := 0

NodeCell[0][0] := 0

For each cell

{ AssociatedNodes[] := nodes associated with the cell
For each node in AssociatedNodes
{
    For(i:=0;i<=NumberOfColumns;i++)
    {
        If (NodeCell[Cell row][i] == node)
        {
            NodeCell[Cell row][Node column] := 1
            Break the loop
        }
        Else
        {
            Add a column to NodeCell matrix
            NodeCell[Cell row][LastColumn] := 1
        }
    }
}

6.3.1.2 Complexity Analysis

Let there be n neighboring cells. The complexity to process each neighboring cell is O(n). For each neighboring cell the nodes associated with that cell are processed and added to NodeCell Matrix. Each cell has three nodes associated with it as we are using triangular elements. So the complexity to process each node of a neighboring cell is
constant time. If there are n nodes already added to NodeCell matrix then the searching process to find out if the node is already added to the NodeCell matrix has a complexity of O(n).

Total complexity to construct the NodeCell matrix is given by

\[ \text{Complexity to process neighboring cells} \times \text{Complexity to process nodes in each cell} \times \text{Searching Process} \]

\[ = n \times 3 \times n \]

\[ = 3n^2 \]

\[ = O(n^2) \]

6.3.2 Determining the Destination Node

This process determines the next node where the knife is to be placed. This process uses knife angle, cutting angle and reference angle to determine the destination node. Knife angle is the angle of knife at the current cutting node relative to the X – axis. Cutting angle is the angle of the node edge, relative to the knife, that the knife should follow to reach the next destination node. Cutting angle is given by knife angle + 90 degrees. Reference angle is the angle of each neighboring node, relative to the current cutting node.
The reference angle is calculated by determining $dx$ and $dy$ between current cutting node and the reference neighboring node.

$$dx = X\text{ (Current cutting node)} - X\text{ (Reference neighboring node)}$$

$$dy = Y\text{(Current cutting node)} - Y\text{(Reference neighboring node)}$$
Reference angle (relative to x – axis) = \[
\begin{align*}
\text{if } dx &= 0 & \text{Angle} &= 0 \\
\text{if } dy &= 0 & \text{Angle} &= 90 \\
\text{Angle} &= \tan (dy/dx)
\end{align*}
\]

The reference angle relative to the X – axis is then converted to reference angle relative to knife by adding the knife angle to the reference angle relative to the X – axis. This conversion helps in comparing cutting angle, which represents the trajectory to be followed by the knife, and the reference angle.

To determine the destination node the cutting angle is compared with each neighboring nodes reference angle. The reference angle which has the least difference with the cutting angle is taken as the next destination node.

6.3.2.1 Pseudo Code

\begin{verbatim}
KnifeAngle := Angle of knife obtained as input (relative to X – axis)
CuttingAngle = KnifeAngle + 90
NeighboringNodes[] := nodes in the neighborhood of the current cutting node
MinDifference := 360 degrees
Destination node := NeighboringNodes[0]
For each node in NeighboringNodes
{
    Determine the reference angle (relative to X – axis)
    Reference angle (relative to knife) := Reference angle (relative to X – axis) + KnifeAngle
    If ( (CuttingAngle - Reference angle (relative to knife)) < MinDifference
\end{verbatim}
\[
\text{MinDifference} = \text{CuttingAngle} - \text{Reference angle (relative to knife)}
\]

\[
\text{Destination node} := \text{node}
\]

6.3.2.2 Complexity Analysis

Let the number of neighboring nodes to the current cutting node be n. Then to determine the destination node each node’s reference angle has to be compared with the cutting angle. The complexity to compare each node’s reference angle with the cutting angle is O(n).

6.3.3 Cloning Process

After the destination node has been determined the current node has to be cloned and the cell node association has to be modified. The nodecell matrix is used for this purpose. The term reference nodes refer to those nodes from which the knife had traveled to the current node. That is, it represents the current node and the cloned node in the previous cutting step. The reference node of each cutting step is obtained from the cut history. In order to update the cell node association the node ordering scheme has to be performed.
6.3.3.1 Node Ordering Scheme

The Nodecell matrix is used to perform the node ordering scheme. One of the reference nodes is taken as the starting point and the other is taken as the ending point. The node ordering scheme is performed by traversing through the Nodecell matrix and updating the OrderedNode and OrderedCell arrays accordingly.

![Grid structure before cutting node #16](image)

The column of start reference node in Nodecell matrix is searched to find the value 1. This gives the cell associated with the node. The value of current cell is updated to the row header value. The value 1 is then changed to value 999. The value of the current cell is appended to OrderedCell array. This means that the node has already been traversed. The next node number associated with this cell is found by searching the row for the value 1. This gives the next node associated with the cell. The value of the current node is updated to the column header value. The current node value is appended to the OrderedNode array. Every node other than the reference nodes, in Nodecell matrix, will
be shared by two cells. So to determine the next cell that shares the node, the
corresponding column is searched for the value of 1, then the value of current cell is
updated to the value of the row header and the value is then appended in the OrderedCell
array. Repeat the process until the current node becomes equal to the end reference node.

Figure 6.6 Node cell matrix

In the figure shown above current cutting node is #16. The two reference nodes
are nodes 6 and 102. Node #102 was cloned in the previous cutting step. The start
reference node is assigned as 6 and the end reference node is assigned as 102. The
Nodecell matrix is traversed by starting from the node #6. The next cell node is cell #45.
The value is then updated to 999 to denote that the node has already been traversed. The process is repeated until node #102 is reached.

![Figure 6.7 Ordered node array](image1)

![Figure 6.8 Ordered cell array](image2)

### 6.3.3.2 Pseudo Code

```
StartReferenceNode := ReferenceNode[0]
EndReferenceNode := ReferenceNode[1]
CurrentNode := StartReferenceNode
CurrentCell := NULL
OrderedNodeArray[Index] := CurrentNode
While (CurrentNode != EndReferenceNode)
{
    Check for value 1 in NodeCell matrix – NodeCell[i][CurrentNode]
    NodeCell[i][CurrentNode] := 999
    Update CurrentCell := NodeCell[i][0]
    Append CurrentCell in OrderedCell array
}  
```
Check for value 1 in NodeCell matrix – NodeCell[CurrentCell][i]

NodeCell[CurrentCell][i] := 999

Update CurrentNode := NodeCell[0][i]

Append CurrentNode in OrderedNode array

After the current cutting node has been cloned and the node ordering scheme is done, the node cell association has to be updated. This is performed using the OrderedNode array.

![Diagram](image1)

**Figure 6.9 Ordered node array used to update cell node association**

![Diagram](image2)

**Figure 6.10 Ordered cell array used to update cell node association.**
Figure 6.11 Grid structure after cutting node #16

The OrderedNode array is traversed to find the destination node in that array. Note the cells that share the destination node. All the cells in the OrderedCell array, until one of the shared cells are encountered, are associated with the current cutting node. The cloned node is associated with every other cell in the OrderedCell array.

6.3.3.3 Complexity Analysis

Let the number of nodes in the matrix be \( n \). Then the number of cells in the matrix will be \( n-1 \). This is because each cell has three nodes. But one of the nodes is common to all the cells. This is omitted and so each cell has two nodes. All the nodes except the reference nodes are shared by two cells. These shared nodes are represented only once in the matrix. So each pair of nodes is associated with one cell. The total number of nodes that can be traversed is given by
Total number of nodes = number of rows * number of columns

\[ = n \times (n-1) \]

\[ = n^2 - n \]

This is for the traversal along the rows. The traversal along the columns again takes \(n^2-n\).

Total Complexity = \((n^2-n) + (n^2-n) = 2(n^2-n) = O(n^2)\).
CHAPTER VII
SURGICAL ANIMATION MODULE – IMPLEMENTATION

7.1 Overview

The surgical animation module acts as an interface between the virtual surgery system and the visualization tool kit. The visualization tool kit is used for displaying the animation process. The surgical animation module implements a class library which acts as a wrapper around the visualization tool kit. The developers of the virtual surgery system can use the class library to create virtual objects, such as surgical knife, to model objects of the virtual surgery system.

The surgical animation module implements the key data structures to store node and cell information. As the cell and node information are accessed by all the other modules several times at each time step, the access time is significant. The data structures used in this module have been designed to reduce this access time. The module implements the same data structure to store node and cell information.

7.2 Key Definitions

- Hash function – Used to determine the address of the location where the data is to be stored.
• Collision – When to data values get the same address from the hash function collision occurs.

7.3 Virtual Surgery Animation Class Library

This section describes the virtual surgery animation class library. It describes the general purpose of each class in the library. Each class in this library starts with the letters “VS” denoting virtual surgery system.

VSData – Provides data structure and functionality for storage and retrieval of grid node and the data associated with them.

VSGrid – Provides data structure and functionality for storage and retrieval of grid cells and the nodes with it.

VSTexture – Acts as an interface to provide texture mapping to the virtual surgery display system.

VSLookupTable – Constructs a lookup table with Hue, Saturation and Intensity parameters.

VSText – Acts as an interface to provide textual display to the surgical system. It provides functionality to map font properties to the textual display in the surgical display system.

VSSurface – Used to provide three dimensional appearances to the surgical model generated by the Model Definition module.

VSTool – Creates and displays the cutting tool. It incorporates methods to animate the position and angle of the cutting tool in real time.
VSResolveColor - Has functionality to resolve color mapping. It provides a named constant for each color mapping.

VSRenderer – Incorporates methods to add various objects to the display system.

7.4 Data Structures and Algorithms

The data structure used to store the node and cell information uses a hybrid data structure generated from a hash table and red black trees [21]. Each node or cell is identified by a numerical id. The key criteria of interest are reducing the access time and expandability. Since the virtual surgery system adds nodes at each time step the data structure used to store the information should be expandable.

7.4.1 Open Hashing

Hashing is a process of storing and retrieving data directly [21]. It is used as a direct access storage method. In hashing each data value is identified by an identifier. The storage location of each data value is computed by applying the hash function on the unique identifier of data value. A desired property of a hash function should be such that it should return a unique address for each data value. When more than one data value gets the same address, a collision occurs. While retrieving data the same hash function is again applied on the unique identifier to compute the address of location where the data value was stored. There are different ways to solve the collision problem. In open hashing, a separate storage location is used to store the values that form collisions. One of the implementations of open hashing is bucket hashing [21] which uses a separate list or bucket to store the collided values.
Figure 7.1 Block diagram showing bucket hashing – a type of open hashing

The disadvantage of this approach is that when there are lot of collided values the bucket becomes very large and the access time to retrieve a value again becomes $O(n)$, where $n$ is the number of values in the bucket.

7.4.2 Red Black Trees

Red black trees [21] are a form of binary search trees. In a binary search tree the tree gets unbalanced easily. When the tree gets unbalanced the right sub tree or the left sub tree of the binary search tree becomes long. In this case the access time again
becomes similar to accessing a value in the sequential list. The advantage of a red black tree is that it keeps the tree balanced. This is performed by reordering the nodes in the tree. The reordering of nodes is done in constant time and it does not induce any further complexity.

7.4.3 Node and Cell Info Data Structures

The data structure used to store the node and cell information uses open hashing but solves the problem of lengthy buckets by taking advantage of red black trees. It also replaces the common bucket with separate red black tree structures for every hash table location. This modification reduces the access time taken to get the information from the node and cell info data structures.

7.4.3.1 Insertion Process in Node and Cell Info Data Structure

When values are inserted into the node and cell info data structure, each data value is assigned a unique numeric identifier. The hash function is applied against this numeric identifier to get the address of the location in the hash table. A modulo hash function is used to compute the address of the location in the hash table. If the hash table already has a value at that location, a collision occurs. In this case the data value is stored in the red black tree if it already exists. Otherwise a red black tree is created for that hash table location and the data value is inserted in the tree.
<table>
<thead>
<tr>
<th>Location address</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data identifier value</td>
<td>30</td>
<td>11</td>
<td>72</td>
<td>33</td>
</tr>
</tbody>
</table>

**Memory allocated for node info or cell info**

**Pointers to storage locations**

- Red black tree nodes containing the numeric identifier and pointer to node or cell information
- Hash table containing one data value and pointer to red black tree structure to store collided values

**Node Information Structure**

- Node’s numeric identifier
- Pointer to array of cell identifiers associated with this node
- Number of cells associated with the node
- X location of the node
- Y location of the node
- Z location of the node
- Scalar value associated with the node - stiffness

**Array containing cells associated with the node**
7.4.3.2 Retrieval Process in Node and Cell Info Data Structure

When the values are to be retrieved the identifier of the node or cell for which the value is to be retrieved is used to find the address of the location. The identifier is applied to the hashing function, which returns the address of the location. The hash table is then checked to see if the identifier exists. If not the red black tree corresponding to that node is traversed to get the cell or node information.

7.4.3.3 Working of Red Black Tree

A red black tree [21] can be considered as a balanced binary search tree. In order to have the tree balanced, the red black tree performs reordering processes on each insertion and deletion. Each node in the red black tree is colored either red or black. Each red black tree satisfies four properties which makes sure that the tree is balanced:

- Red node’s children are always black.
- All the leaf nodes are black.
- Black color is assigned to the root node of the tree.
- All the leaf nodes have same number of black nodes in a path from the root node.
Two reordering operations are performed to have the tree balanced. They are restructuring and re-coloring. The reordering process is done when double red occurs in child and parent nodes. The reordering process that is to be performed is chosen based on the color of parent’s sibling. Restructuring is performed when parent’s sibling is black and re-coloring is performed when parent’s sibling is red.

![Restructuring](image1)

**Figure 7.3 (a) Restructuring**

Restructuring is performed by rearranging the pointers and takes constant time. Re-coloring changes the colors of parent, parent’s sibling and grand parent. Re-ordering is also performed in constant time.

**7.4.4 Complexity Analysis**

The worst case complexity for insertion into node and cell info data structures can be found by considering that the address computed by the hash function produces a collision. In this case the red black tree has to be traversed to find out a suitable place for the new node. If there are n nodes in the tree then the complexity to traverse the tree is
O(log n), since the tree is always balanced. The re-ordering process takes constant time O(1). The complexity to insert n nodes is given by O(n log n).

The worst case complexity to retrieve from the node and cell info data structure can be found by considering that the information is not available in the hash table. In this case the tree has to be traversed to get the information. The complexity to traverse the tree is O( log n). In order to retrieve n nodes it takes O(n log n).
8.1 Overview

The virtual surgery system employs a novel approach of controlling the surgical operation using a fuzzy control system. The system was build with efficiency and speed as the foremost criteria. These two criteria made the system work in real time with great responsiveness. The data structures and algorithms used in the system have been analyzed based on the worst case complexity. The multi threaded approach allowed the system to synchronize without any speed penalty. The next section walks through the testing processes that have been conducted to evaluate the operational ability of the system.

8.2 Fuzzy Logic Testing Results

The output of fuzzy control system has been tested for random input forces by using both uniform and non-uniform soft tissues. For each input force the depth of cut was calculated by the fuzzy control system based on the stiffness of the soft tissue at the current location. The value of the input force, the stiffness at that current location and the depth of cut has been tabulated. To make it easier for future calibration, the force and stiffness are normalized to the range of [0,1]. The depth of cut is measured in number of
grid nodes.

Table 8.1 Results from fuzzy control system for a uniform grid.

<table>
<thead>
<tr>
<th>Force</th>
<th>Stiffness</th>
<th>Depth of Cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.280713</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.2</td>
<td>4</td>
</tr>
<tr>
<td>0.345254</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.664831</td>
<td>0.2</td>
<td>2</td>
</tr>
<tr>
<td>0.316228</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.55211</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.2</td>
<td>4</td>
</tr>
<tr>
<td>0.505594</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.26613</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.495202</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.265754</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.347131</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.241299</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.336786</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.300666</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.428019</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.464785</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.447772</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.241299</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.41</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.340331</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>0.628013</td>
<td>0.2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 8.2 Results from fuzzy control system for a non–uniform grid.

<table>
<thead>
<tr>
<th>Force</th>
<th>Stiffness</th>
<th>Depth of Cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.26</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>0.412311</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>0.412311</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>0.655134</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.9</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
<td>5</td>
</tr>
<tr>
<td>0.320351</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>0.320039</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>0.300167</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>0.317844</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>0.330189</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>Force</td>
<td>Stiffness</td>
<td>Depth of Cut</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td>0.504777</td>
<td>0.7</td>
<td>2</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>0.308707</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>0.528394</td>
<td>0.7</td>
<td>2</td>
</tr>
<tr>
<td>0.278568</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.264008</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.340588</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.327567</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.363352</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.413793</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.381608</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.415963</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.433244</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.4272</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.458803</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.496412</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.475079</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.46457</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.479401</td>
<td>0.6</td>
<td>0</td>
</tr>
<tr>
<td>0.547814</td>
<td>0.6</td>
<td>1</td>
</tr>
<tr>
<td>0.907083</td>
<td>0.6</td>
<td>3</td>
</tr>
<tr>
<td>0.340588</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>0.891067</td>
<td>0.5</td>
<td>5</td>
</tr>
<tr>
<td>0.465188</td>
<td>0.4</td>
<td>1</td>
</tr>
<tr>
<td>0.280045</td>
<td>0.4</td>
<td>1</td>
</tr>
<tr>
<td>0.448219</td>
<td>0.3</td>
<td>2</td>
</tr>
<tr>
<td>0.333204</td>
<td>0.3</td>
<td>2</td>
</tr>
<tr>
<td>0.822192</td>
<td>0.2</td>
<td>3</td>
</tr>
<tr>
<td>0.220907</td>
<td>0.2</td>
<td>0</td>
</tr>
</tbody>
</table>

8.3 Performance Testing

The performance testing of the system was divided into two phases. The training phase and the testing phase. In the training phase the users were allowed to get used to the system by performing virtual surgical operations with arbitrary paths. This was repeated until the users were comfortable with the system. In the next phase each user was given a specific path and was asked to follow the prescribed path. Three out of five users who were involved in the performance testing were able to follow the prescribed
path. The other two had slight deviations but after some more training were able to perform better.

8.4 Screen shots

Figure 8.1 Surgical operations – straight view

Figure 8.2 Surgical operations – angular view
Figure 8.3 Surgical operations – tilted view

Figure 8.4 Surgical operations – wire frame display
CHAPTER IX
CONCLUSION AND FUTURE WORK

9.1 Conclusion

The fuzzy logic based virtual surgery system was built to simulate surgery in real time. Careful design, implementation and assembling of the various modules have allowed the system to perform with great efficiency. The system is very responsive to user actions at runtime. It has been demonstrated that the system can be used to conduct real time soft tissue cutting along non pre-specified paths.

9.2 Future Work

The accuracy of the fuzzy control system can be tested by comparing the cutting of virtual soft tissues and the cutting of real soft tissue (animal meat). It can also be compared with finite element based simulation systems [22].

Studies can be conducted to introduce parallel processing in virtual surgery systems. This might result in even more responsive systems. Parallel processing will also allow processing of high resolution models in less time.

This research study provides a good base for further study. A research study on building a distributed virtual surgery system has already been proposed [16]. This system would broadcast surgical operations to others who are simultaneously connected to the network.


