View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Vanderbilt Electronic Thesis and Dissertation Archive

DISABLEMENT OF A SURGICAL DRILL VIA CT GUIDANCE TO PROTECT VITAL ANATOMY

By

Christopher C. Heath

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

In

Electrical Engineering

August, 2008

Nashville, Tennessee

Approved:

Professor J. Michael Fitzpatrick

Professor Benoit M. Dawant

To Kristin, who, for some reason, put up with me

ACKNOWLEDGMENTS

I would like to take this chance to thank those that made this work possible. I am indebted to Dr. Mike Fitzpatrick, who took a chance and helped me out by allowing me to work on this project. I have loved working on this project and hope that its development is continued, as I really feel that it could be beneficial to patients in the near future. I would also like to thank Dr. Benoit Dawant, who helped me find this project, and agreed to be on my committee.

I am extremely grateful to those who gave me assistance when I needed it: Ramya Balachandran, Omid Madjani, Jason Mitchell, and Jack Noble. I would like to especially thank Ramya as she spent many, many hours helping me out well above and beyond the call of duty.

I would be remiss to forget Dr. Robert Labadie, who continuously expressed interest in my work and also taught me what a vagus nerve reaction is.

Finally, I would like to thank my family, who have always loved and supported me, especially my beautiful wife Kristin who has carried me these past years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Pre-operative images have been employed to provide direct geometric guidance to the surgeon since the first use of stereotactic frames on humans in 1947 [1]. However, the use of a computer to display images in real-time in the operating room was not accomplished until the early 1980s, when Kelly and colleagues displayed tumor outlines in response to the movement of a microscope attached to a stereotactic frame [2,3] and Roberts and colleagues superimposed images on a view of the anatomy through a surgical microscope in response to the movement of the microscope which was tracked sonically [4]. Since that time, technology has steadily improved through (a) increased image resolution and geometric fidelity, (b) increased computer speed and memory capacity and (c) improved devices and algorithms for registering patient anatomy from intra-operative physical space to pre-operative image space. As a result, today, with the use of bone-implanted fiducial markers, a surgeon can see images on a computer display updated thirty times per second showing the position of a tracked probe to an accuracy of better than 1 mm [5], and with skin-affixed markers or laser contouring, to an accuracy of between 1.3 and 2.7 mm [6]. Peters has given recent review of the literature on image guidance for surgical procedures [7].

A recently implemented adjunct to the tracking of a surgical probe and the updating of displayed images based on probe position is the tracking of a surgical drill during bone resection and control of the motive power to the drill based on drill position. The goal of

this approach is to relieve the surgeon of the need continually to check the position of the drill as depicted on the pre-operative image volume while simultaneously operating the drill to resect bone. The need for checking the position arises from the need to avoid injuring critical anatomic structures. For example, during a mastoidectomy, the surgeon must avoid contacting the facial nerve with the drill. The location of the nerve just below the bony surface is not obvious visually, but it is clearly visible in a pre-operative CT image volume. Thus, the 3D surface of the nerve can be determined pre-operatively in the CT so that, during drilling, when the drill tip approaches that surface, power to the drill can be automatically removed under computer control, thereby disabling it and protecting the nerve. Andrew Jurik implemented a program in 2007 as part of an independent study to control a drill's power [8]. That work involved the writing of code in C# to implement both tracking and power control and testing on phantoms, but no CT images were involved. This thesis reports on a continuation and expansion of Jurik's work, which has resulted in a system that allows tracking of the drill relative to a previously acquired CT volume image of the specimen being drilled.

Mastoidectomies present an ideal application for this drill disablement application. While there are over 100,000 mastoidectomies performed per year, image-guided systems are not prevalent in the fields of otology and neuro-otology [9]. Additionally, there are many critical structures present in the surgical field, such as the facial nerve, horizontal semi-circular canal, and the sigmoid sinus. While injury to these structures is fairly rare (<1% of mastoidectomies result in facial nerve palsy [10]), the results from damaging them can be quite severe. Thus, adding safety controls can only benefit the patient. While

image-guided systems in this area are rare, Strauss and colleagues recently released a feasibility study on the possible use of a similar drill-control system [11].

Continuing development of Jurik's program necessitated porting the program to C++, and incorporating the Image Guided Surgery Toolkit (IGSTK), a free and open-source C++ library [12]. IGSTK is based both on the Insight Segmentation and Registration Toolkit (ITK) [13] and the Visualization Toolkit (VTK) [14]. ITK is a free and open-source library that implements segmentation and registration algorithms for volumes. VTK is also free and open-source, and is used for visualization, 3-d graphics, and image processing. The IGSTK library utilizes select methods from these libraries in order to create a framework that allows for rapid development and testing of image guided surgery applications. All three of these libraries are cross-platform and are widely utilized.

The move to IGSTK provided (a) platform-independence, (b) increased speed and reliability, and (c) faster development time. Platform independence was achieved through the use of both IGSTK for program development and the Fast Light Toolkit (FLTK) for GUI development [15]. FLTK is also free and open-source, and is a cross-platform C++ library. The use of IGSTK provides increased speed because it contains fast methods for image registration and probe calibration. Previously, these functions were provided by MATLAB®-based .NET components, which took significant time to utilize. IGSTK also provides increased application reliability because it was developed as an event-based library that emphasizes error recognition and prevention. Realizing that their library would be utilized in mission-critical situations, the developers crafted IGSTK to be very

robust. As part of this robustness, IGSTK is written as a state machine. Lastly, IGSTK provides an abundance of methods and is well documented. These assets facilitate the rapid development of code.

At the onset of this project, several benchmarks were decided upon. First, the general functionality of the previous iteration would be reproduced. This functionality included calibrating the drill and probe using fiducial markers, using the probe to mark the boundary of a volume, and tracking whether the drill was inside or outside the volume. Secondly, a CT image of a phantom would be used to mark the boundary. Thirdly, the phantom would be altered to represent common anatomy. Lastly, the program would be quantitatively verified in a variety of methods.

Chapter II gives an overview of the system—both hardware and software—focusing on the functionality. Chapter III describes the organization of the software. Chapter IV describes the methods employed to validate the system, while Chapter V investigates the results of these tests. Chapter VI presents conclusions drawn about the drill disablement system, and Chapter VII suggests future work to enhance the system.

CHAPTER II

SYSTEM FUNCTIONALITY

The system consists of both hardware and software. Both of these are described in this chapter. The hardware consists of a surgical drill, a power supply for the drill, a controller that sends power from the supply to the drill, a tracking system to determine the location of both the patient and the drill, and a computer that communicates with the tracker and sends signals to the controller. The software is a program that resides on and is executed by the computer. In this chapter, the functionality of the system is described. Most of the chapter deals with the software. The software will be referred to as the "application".

1. Overview

The purpose of this application is to track the location of a surgical drill using the Claron MicronTracker (Claron, Inc., Toronto, Canada) and to disable the drill so that it stops spinning when it enters a restricted region. It is also intended to be simple, robust, and as error-proof as possible.

Figure 1. The Claron MicronTracker. The two round dark spots are digital cameras.

The MicronTracker is a third-generation optical pose tracker [16], meaning that it is totally passive and uses available light to observe objects stereoscopically. The stereoscopic view is provided by a pair of cameras that are mounted rigidly on a holder, as shown in Figure 1. The size of the MicronTracker is relatively small with a length, width, and depth of approximately 172, 57, and 57 mm. In order to facilitate tracking, printed "markers" are attached to the objects that are to be tracked. Each marker consists of a set of three checkerboard patterns of high contrast. A sample checkerboard pattern is shown in Figure 2. The diameter of the circle in Figure 2 is 23 mm

Figure 2. A high-contrast checkerboard for the Claron MicronTracker

The intersection of the checkerboard is referred to as an "XPoint". The MicronTracker employs image processing to determine the two-dimensional position of an XPoint in each camera view, and tracks its location in three-dimensional space by triangulation from the two two-dimensional positions. In order to qualify as a marker, the three checkerboards must be spatially related in a specific manner, as documented in the MicronTracker's developer's manual. With the help of Claron software, a marker calibration file can be created that records, for each marker, the spatial relationship

among its three checkerboards. In this procedure, each XPoint is uniquely identified. In order for a marker to be tracked, all three of its XPoints must be visible by both cameras.

Two surgical tools have been rigidly attached with markers for use with the drill disablement system. One tool is the surgical drill itself. The other is referenced as the probe. The probe is primarily used to localize the fiducial markers in physical space, and can also be used for testing and debugging purposes while tracking. An image of the probe is shown in Figure 3.

Figure 3: The probe surgical tool. The probe is used to localize fiducial points.

The camera model being utilized in this research is the S60. This model has a measurement rate of 30 Hz, has a spatial resolution of 640x480 per image, and has a lag of about 45 ms. Its accuracy is reported as 0.25 mm RMS within the field of measurement (FOM). The FOM is defined as a rectangular section of a sphere. The radius, width and height of the FOM measure 115, 70, and 55 cm, respectively.

In designing the application, the goals of simplicity and error-resistance were paramount. One method of achieving these goals is to use a state-machine model for the application. The state machine diagram for the application is shown below on page 13 in Figure 5. States are denoted by labeled ovals. Transitions are denoted by arrows. This approach reduces the complexity of the application and makes the order of the application clear. In order to enforce the state machine, the respective buttons are activated and deactivated, both providing visual feedback to the user and disallowing illegal transitions. The state machine diagram is simplified by leaving out states for clarity. For example, the state "Localize_Physical_Fiducials" is actually a combination of several states. Since a minimum of three fiducial points are required to complete a registration, the state of the application is different when zero, one, two, or more than two fiducial points are localized. Additionally, the state of the application changes when attempting to localize the next fiducial, and the attempt can either succeed or fail. The full state diagram that is represented by "Localize_Physical_Fiducials" can be seen in Figure 4.

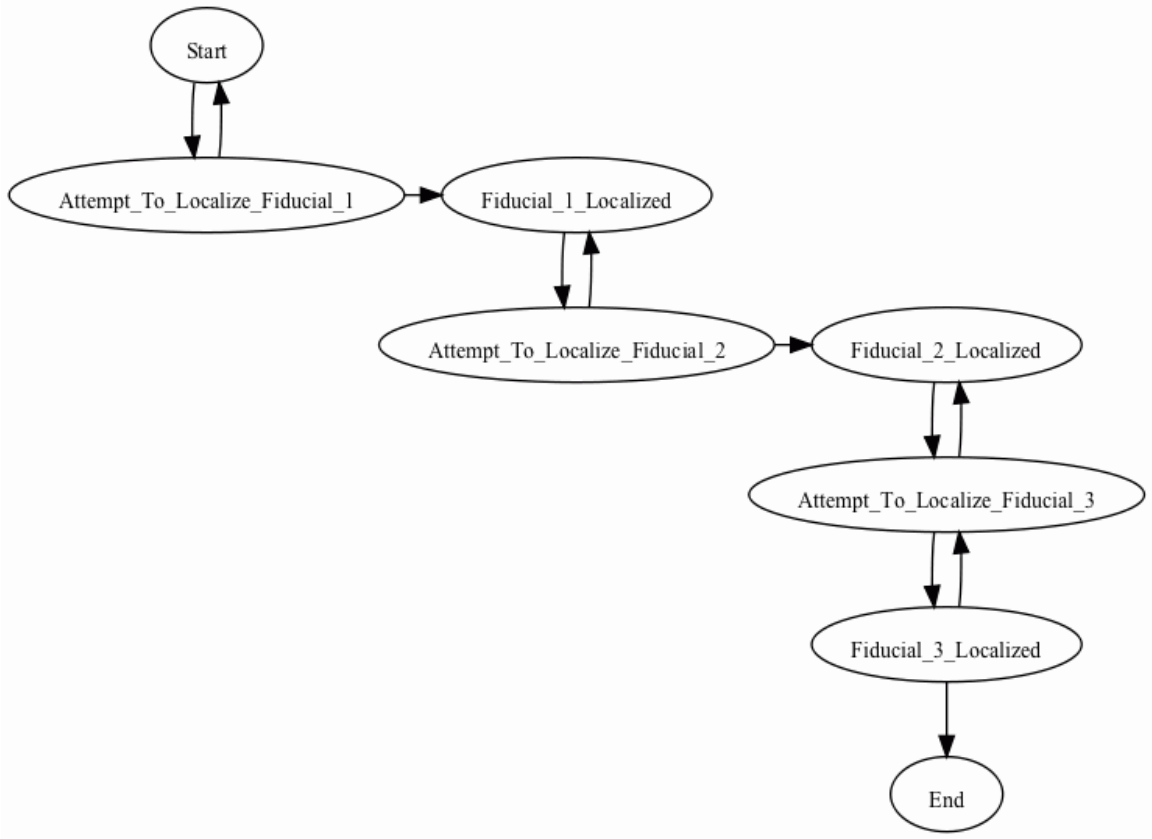

Figure 4: The state machine diagram for Localize_Physical_Fiducials

The first state transition in the application occurs when the CT image is loaded into the program (i.e., from the state labeled “Start” to the state labeled “Load_CT”). This image must be in the Digital Imaging and Communications in Medicine (DICOM) [17] format. The methods used to load and display the CT image were provided by IGSTK, and the user interface used to inspect the image is shown in Section 2.

Within the Segment_CT state, the CT segmentation file is loaded into memory. The CT segmentation file is actually a combination of a binary array of short integers and a binary header file. The header file maps the array elements to the voxels in the CT image,

and the binary array is a distance map, where each array element is the distance in micrometers from the corresponding voxel to the nearest point on a restricted region. Typically, the CT image slices are about 400 microns thick and have a slightly smaller width and height. However, the MicronTracker has an accuracy of about 250 micrometers [18]. To maximize the accuracy of the system, linear interpolation is employed to determine the distance from the drill to the restricted region. This process is described in more detail in Section 5.

The next step, which occurs during the state “Initialize-Tracker”, is to initialize the MicronTracker, which consists of loading a directory that contains (i) the calibration file for the MicronTracker, (ii) the marker template directory for the MicronTracker, and (iii) an initialization file. The calibration file is unique to each individual tracker, is generated by Claron at production, and comes with the camera at purchase. The user can also generate the calibration using software provided by Claron. The marker template files specify the spatial relationship among a set of three XPoints and are used by the MicronTracker to differentiate the XPoints. Finally, the initialization file contains the configuration settings to be used by the camera, such as the light coolness, gain, and shutter limiter. At this point, the MicronTracker is connected to the application and is ready to use.

The next step is to calibrate the surgical tools. This calibration can be loaded from a text file (“Load_Calibration”) or done manually (“Calibrate_Manually”), and consists of the spatial coordinates of the XPoints and the tool tip for the surgical instrument at some

reference time. The calibration files are simply text files that identify the coordinates and the marker name that they come from. Manual calibration is discussed in Section 3 of this chapter.

At this point, the camera and CT image are set up and the application is ready to begin registering physical space to CT space. This machine state is labeled “Idle”. At any subsequent point, the user can choose to return to this state in order to redo the registration process.

The first step in registering the two spaces is to choose the CT fiducial points. These points are features that can be seen in both the CT image and in physical space, and are used find the transformation between the two. The application has three methods for inputting CT fiducial points. All methods involve inputting their spatial coordinates as seen in CT space. First, the points can be manually typed in via the keyboard if they are known beforehand. Secondly, the user can click on the CT image, recording the position of where the click occurred. Both of these methods use the same functions, and are labeled in the state machine diagram as “Localize_CT_Fiducials”. Finally, a file containing the CT coordinates can be loaded into memory (“Load_CT_Fiducials”).

Next, the fiducial points are localized in physical space (“Localize_Physical_Fiducials”). The physical points must be input in the same order as the CT points. If the CT fiducial points were loaded from a file, the order of the fiducial points must be known beforehand.

To locate the fiducial points, the “probe” surgical tool tip is placed on the fiducial marker, and the location of the tool tip is recorded.

During the state “Calibrate_Reference”, the coordinate reference frame (CRF) is calibrated. The CRF is a set of markers that is rigidly attached to the target. The frame can then be used to take into account movement of either the target or the camera during the tracking process. To calibrate, the location and names of the markers on the CRF are obtained. This process is discussed further in Section 5 of this chapter.

After the physical fiducials are localized, physical space is registered to CT space (“Register_CT_Space”). This registration allows the tracker to track the surgical tool in CT space. Again, this process is discussed further in Section 3 of this chapter.

Finally, the surgical tools are tracked in CT space. This process occurs in the machine states of “Track_Probe” or “Track_Drill” depending on the surgical tool the user wishes to track. The probe is used for debugging and testing purposes, while the drill is obviously used for surgical purposes. The tracking process is discussed further in Section 4 of this chapter. As the drill is tracked, the application determines whether or not the tip is inside the restricted volume. Power to the drill is shut off if the tip enters the restricted volume. This process is described in detail in Section 5 of this chapter.

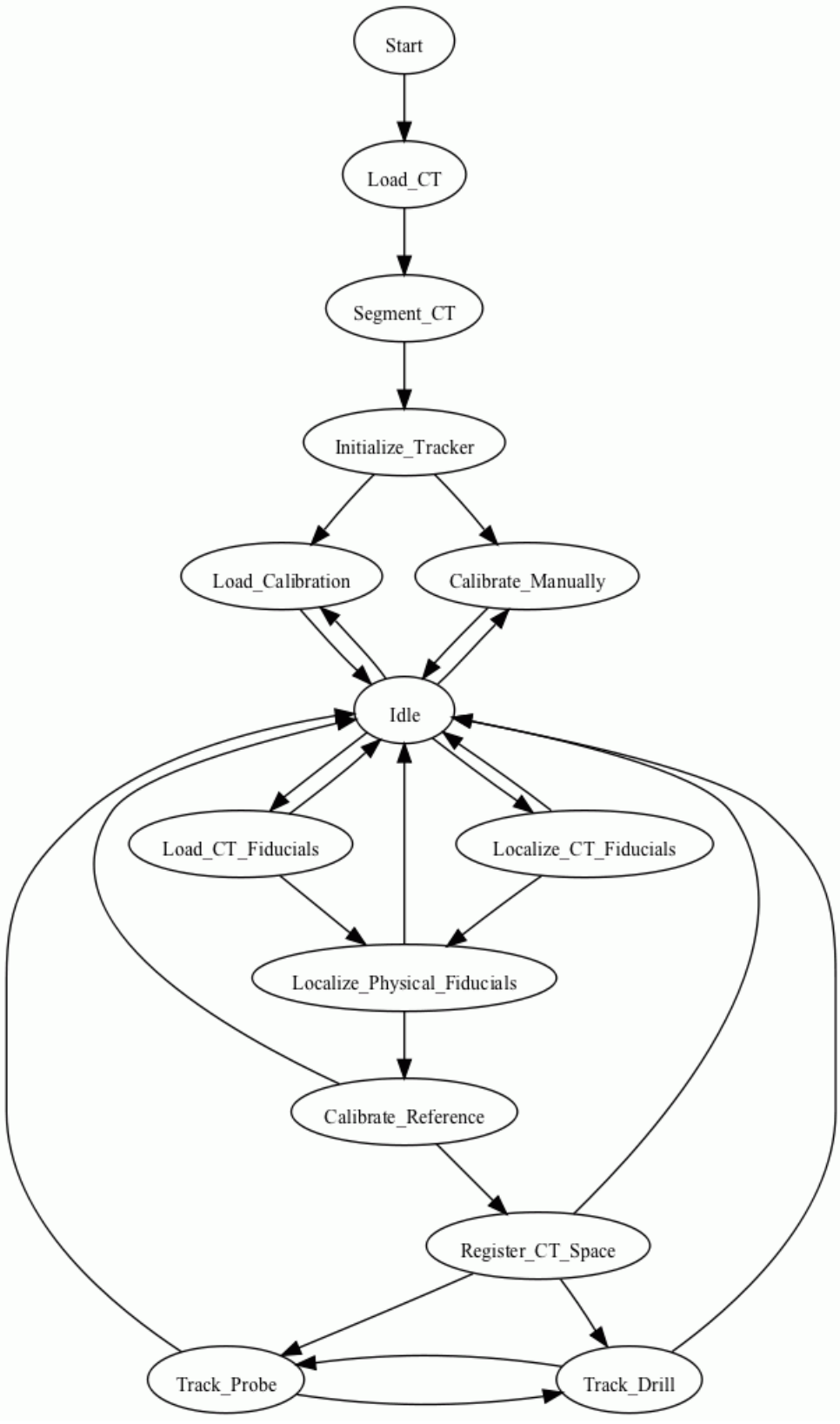

Figure 5. State machine diagram for the drill disablement application

2. Display

A graphical user interface (GUI) is extremely important for any application, but for mission-critical applications they are even more so. In an effort to reduce complexity and maximize information, the GUI for this application was set up to be simple, but provide the relevant feedback at any particular state. Figure 6 is a screenshot of the application's main GUI. It was generated using FLTK.

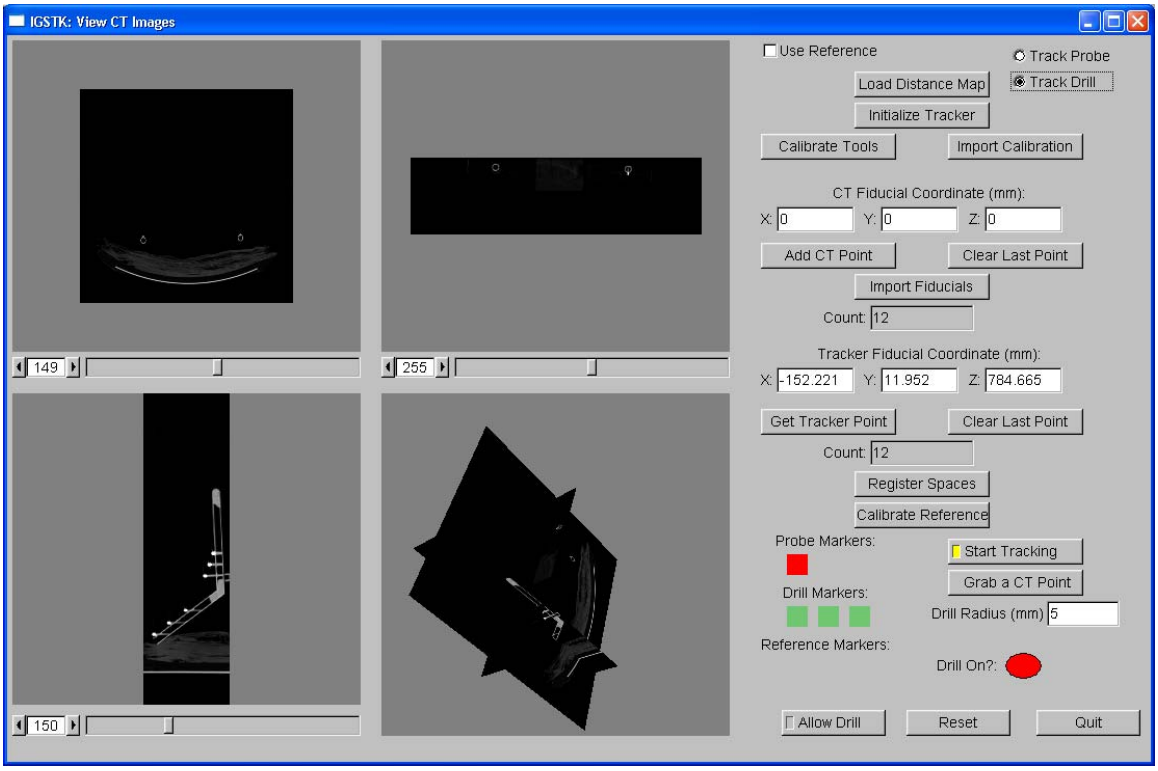

Figure 6: The main application GUI.

The majority of screen space is taken up by the CT image. Because this is a volume image, it is necessary to show multiple views. The upper left view shows one slice through the volume in the xy plane. The upper right view shows one slice through the xz

plane, and the lower left view shows one slice through the yz plane. Sliders beneath each view can be used to scroll through the slices. Additionally, a 3D image is shown at the lower right that consists of a composite of all 3 views. Clicking and dragging on this view will rotate the image for a more thorough inspection. Additionally, the images automatically re-slice during the tracking process to display on-screen the location of the tool tip.

The right side of the screen consists of various buttons that call different methods. In order to enforce the state machine aspect of the programming, the buttons are activated and deactivated at the relevant portions of the application. Also, the flow of buttons from top to bottom follows the flow of the application, re-enforcing the state machine behavior.

Several indicators are located at the bottom-right of the screen. The red oval indicator at the bottom right gives visual feedback on whether the application is stopping the drill. It changes color from red to green to indicate the status of the drill. Red indicates that the drill is off; green indicates that it is on. Square indicators change from red to green when a particular marker is visible to the camera. The number of square indicators is set during the calibration of the tool tips, where each square indicator represents the visibility of a particular marker. This feedback can be useful to determine how to properly hold the drill, and to troubleshoot the application if the drill is stopping in non-restricted regions.

Because of the complicated nature of calibrating the surgical tools, a separate GUI exists to facilitate the calibration process. This GUI is shown in Figure 7.

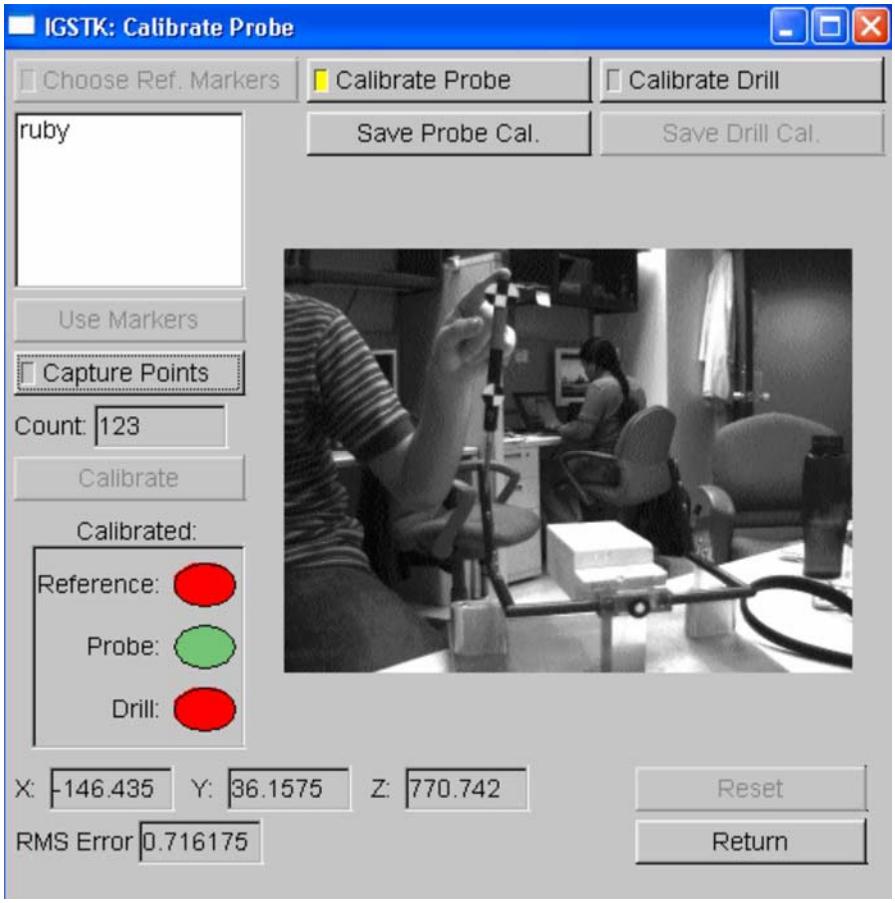

Figure 7: The surgical tool calibration GUI

This interface also provides important information to the user. The largest amount of screen space is taken up by live feed from the MicronTracker. This feed can be used to ensure that the markers on the tool that is to be calibrated are the only ones visible to the tracker. The text output in the top left displays the names of these markers. While the user may not know the specific names of the markers, the number of markers that should

be visible is known. Additionally, the list can be inspected to ensure that no duplicate markers are used, as this can cause problems during the registration process.

The oval indicators change colors to indicate whether or not a specific tool has been calibrated. Green ovals represent calibrated tools while red ovals represent non-calibrated tools. The X, Y, and Z value outputs represent the calibrated position of the tool in tracker space. While there is no way to verify their accuracy absolutely, these values are useful as a "sanity check". Finally, the RMS error reported is the target registration error of the calibration.

3. Calibration

Tracking the location of surgical tools requires calibration of those tools. That is, it is necessary to accurately find the geometric relationship between the fiducial markers that are rigidly attached to the tool and the tip of the tool. From this relationship, the position of the tool tip can be accurately tracked through the movement of the rigid fiducial markers. The application provides a GUI in order to facilitate the calibration. The state machine diagram for the calibration of a surgical tool is shown in Figure 8.

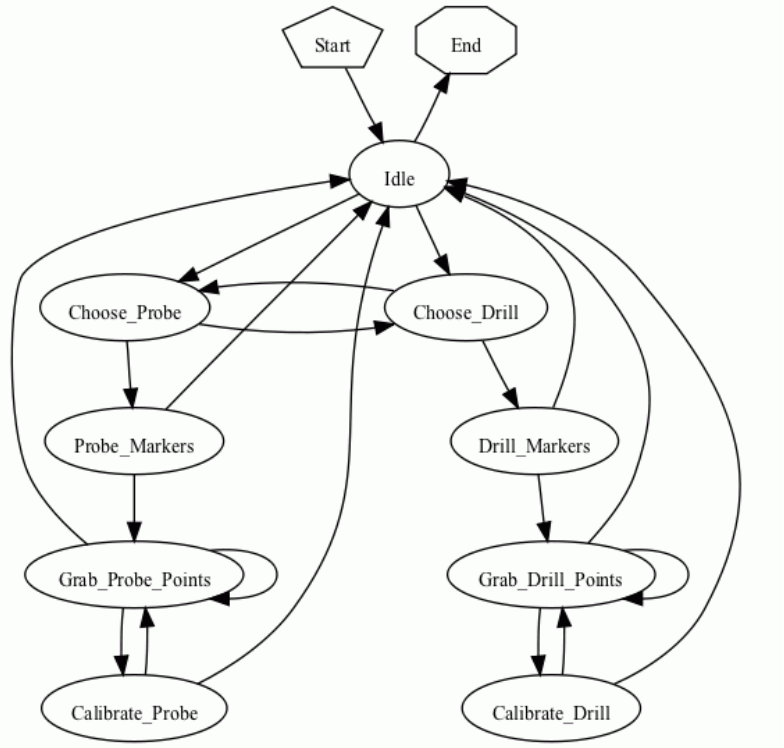

Figure 8: State machine diagram for surgical tool calibration

The first step in calibrating a surgical tool is to designate the tool to be calibrated as either a drill or a probe. The probe is used during the process of registering physical space to CT space for the purpose of choosing fiducial marker locations, and can also be tracked for testing purposes. In order to proceed in the application, the probe must be calibrated. The drill is the surgical tool that is to be tracked and stopped, and cannot be used for registration purposes.

Secondly, the markers that are rigidly attached to the tool are confirmed. The GUI automatically displays the names of all visible markers during this process for the user to verify.

Thirdly, the XPoints on the tool are acquired. To do so, the tip of the tool is held stationary while the tip is pivoted back and forth in a variety of angles. During this time, the application continuously records the position of the XPoints of the probe. The MicronTracker provides the x, y, and z coordinate for each of these XPoints, along with a unique identifier, so that points from different times can be matched up. The first set of points acquired is saved as a reference set of points.

The transform from the reference point to each subsequent set of points is then calculated. The transform is a rigid, point-based transform consisting of an orthonormal rotational matrix *R* and a translational vector *t*. The method used to find this transform is described by Horn [19] and is implemented by methods provided by IGSTK. At least three points must be used in order to avoid having an under-determined system. The method is explained in detail below.

First, the centroid of the fiducials in both the reference and the current set of point is calculated, followed by the displacement from the centroid to each fiducial point.

$$\begin{aligned} \overline{x} &= \frac{1}{N}\sum_{i=1}^{n} x_i \\ \overline{y} &= \frac{1}{N}\sum_{i=1}^{n} y_i \end{aligned} \tag{2.3.1}$$

$$\begin{aligned} \tilde{x}_i &= x_i - \overline{x} \\ \tilde{y}_i &= y_i - \overline{y} \end{aligned} \tag{2.3.2}$$

These values are useful for solving the transform. Then the rotational matrix R between the reference points and the new set of points at time i is calculated. Horn shows that the least-square solution for the rotation matrix can be found by maximizing the term

$$\sum_{i=1}^{n} \tilde{y}_i R \tilde{x}_i \tag{2.3.3}$$

Using quaternions, term (2.3.3) can be re-written as

$$\sum_{i=1}^{n} \left(\dot{q} \dot{\tilde{x}}_i \right) \left(\dot{\tilde{y}}_i \dot{q} \right) \tag{2.3.4}$$

where the dot represents a quaternion, and q is the quaternion representation of the rotational matrix R. To multiply quaternions, either the first or second quaternion is expanded into a 4*4 orthogonal matrix, $\mathbb{R}$ or $\bar{\mathbb{R}}$, respectively. These matrices are defined by Horn. Using this fact, term (2.3.4) can be re-written as

$$\dot{q}^T \sum_{i=1}^{n} \left(\bar{\mathbb{R}}_{x,i} \mathbb{R}_{y,i} \right) \dot{q} \tag{2.3.5}$$

Or,

$$\dot{q}^T N \dot{q} \tag{2.3.6}$$

In order to solve for N, the matrix M is defined. Horn demonstrates that this matrix contains all the information required to solve the least-squares problem for the rotation.

$$M = \sum_{i}^{n} \bar{x}_i \bar{y}_i^T \tag{2.3.7}$$

N can be formed from the elements of M by

$$N = \begin{bmatrix} \left(M_{1,1} + M_{2,2} + M_{3,3}\right) & M_{2,3} - M_{3.2} & M_{3,1} - M_{1,3} & M_{1,2} - M_{2,1} \\ M_{2,3} - M_{3.2} & \left(M_{1,1} - M_{2,2} - M_{3,3}\right) & M_{1,2} + M_{2,1} & M_{3,1} + M_{1,3} \\ M_{3,1} - M_{1,3} & M_{1,2} + M_{2,1} & \left(-M_{1,1} + M_{2,2} - M_{3,3}\right) & M_{2,3} + M_{3.2} \\ M_{1,2} - M_{2,1} & M_{3,1} + M_{1,3} & M_{2,3} + M_{3.2} & \left(-M_{1,1} - M_{2,2} + M_{3,3}\right) \end{bmatrix} \tag{2.3.8}$$

It is shown that the eigenvector that corresponds to the most positive eigenvalue of N is the quaternion that is the least-squares solution for the rotation. From the quaternion, the orthonormal rotation matrix R can be found as follows:

$$R = \begin{bmatrix} \left(q_0{}^2 + q_1{}^2 + q_2{}^2 + q_3{}^2\right) & 2\left(q_1q_2 - q_0q_3\right) & 2\left(q_1q_3 + q_0q_2\right) \\ 2\left(q_1q_2 + q_0q_3\right) & \left(q_0{}^2 - q_1{}^2 + q_2{}^2 - q_3{}^2\right) & 2\left(q_2q_3 - q_0q_1\right) \\ 2\left(q_1q_3 - q_0q_2\right) & 2\left(q_2q_3 + q_0q_1\right) & \left(q_0{}^2 - q_1{}^2 - q_2{}^2 + q_3{}^2\right) \end{bmatrix} \tag{2.3.9}$$

From R, the translational vector t can be calculated.

$$t = \overline{y} - R\overline{x} \tag{2.3.10}$$

After a suitable number of time points have been calculated (300 time points was used as a rule of thumb), a set of transform rotations $\{R_1, R_2, \ldots, R_n\}$ and translations $\{t_1, t_2, \ldots, t_n\}$ has been calculated and saved into memory. The tool can now be calibrated. Since the tool tip remains stationary throughout the calibration process, if we define the location of the tool tip as $p' = (x', y', z')$, then for any time period i,

$$R_i p' + t_i = p' \tag{2.3.11}$$

Or,

$$\left(R_i - I\right) p' = -t_i \tag{2.3.12}$$

From the n transforms that have been collected, the matrix S and vector u can be constructed such that

$$S = \begin{pmatrix} R_1 - I \\ R_2 - I \\ \vdots \\ R_n - I \end{pmatrix} \text{ and } u = \begin{pmatrix} -t_1 \\ -t_2 \\ \vdots \\ -t_n \end{pmatrix} \tag{2.3.13}$$

where M is of dimension ($3n$ by 3) and t is of dimension ($3n$ by 1). Since p' remains stationary throughout the entire calibration process, it follows that

$$Sp' = u \tag{2.3.14}$$

Equation (2.3.14) is an over-determined system. Due to the inherent errors in finding each transform and measuring each point, no exact solution exists. Thus, the minimum norm solution must be acquired.

$$p' = (S^T S)^{-1} S^T u \tag{2.3.15}$$

In order to avoid numerical instability, the system was solved through the use of singular value decomposition. First, the singular value decomposition of the matrix S is performed.

$$S = UDV^T \tag{2.3.16}$$

Then, the minimum norm solution is obtained as follows

$$p' = UD^{-1}V^T u \tag{2.3.17}$$

At this point, the calibration is complete. The calibration can then be saved to a file, which consists of the nine XPoints and the location of the probe at the first (reference) acquisition time. From this information, the tool tip can be located at any time by finding

the transform from the reference points to the XPoints at that particular time using the same method as above, and then applying the transform to the tool tip point saved in the calibration.

One method to determine the efficacy of the calibration is to determine the target registration error (TRE). The TRE is defined as the registration error at some point of interest. Because the tool tip does not change position, the TRE for each transform T_i can be defined as

$$TRE = \left\| T_i\left(p'\right) - p' \right\| \tag{2.3.18}$$

Because there are many TRE values, the mean and standard deviation of the TRE are used to evaluate the success of the calibration.

4. Tracking

The process of tracking the surgical tool is the primary task of the application. During tracking, a steady stream of coordinates for the XPoints is obtained from the MicronTracker through APIs provided by Claron. In order to use this information to track the tool tip in CT space, three steps are undertaken. Firstly, movement of both the tracker and the target since registration are accounted for using the CRF. During the Reference_Calibrated state, the spatial coordinates of the CRF markers are recorded and saved. The transformation to take the location of the CRF markers from their current spatial location in tracker space to this original location is calculated. This transformation is then applied to the surgical tool XPoints. The CRF used is seen in Figure 9.

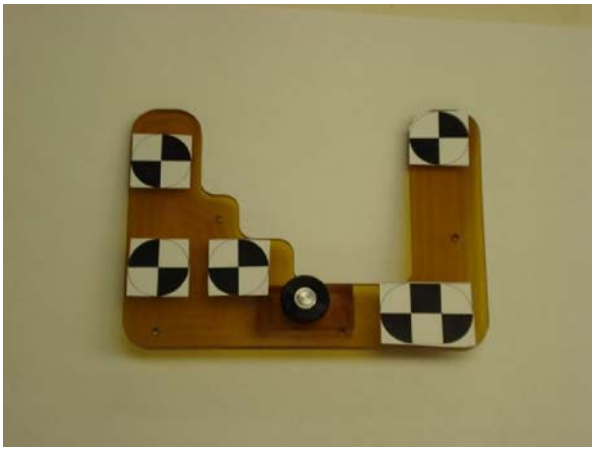

Figure 9: The coordinate reference frame. The CRF is used to account for movement of the target and the tracker during the tracking process.

Secondly, the location of the tool tip in physical space is calculated. Lastly, the transformation from physical space to CT space is applied. These steps are discussed below.

As stated above, the CRF markers are used to account for the movement of the patient during tracking. Using the method described in Section 3, the program calculates the rigid transformation that takes the reference markers from their current location in tracker space to their location in tracker space during calibration. Applying this transformation to the XPoints has the effect of transforming their positions from tracker space to patient space. As a result, the patient can move relative to the MicronTracker, or vice versa, without affecting the XPoints calculated positions. The calculation is as follows:

$$\begin{bmatrix} x_{1,0} & x_{2,0} & \cdots & x_{n,0} \\ y_{1,0} & y_{2,0} & \cdots & y_{1,0} \\ z_{1,0} & z_{2,0} & \cdots & z_{3,0} \end{bmatrix} = R \begin{bmatrix} x_{1,i} & x_{2,i} & \cdots & x_{n,i} \\ y_{1,i} & y_{2,i} & \cdots & y_{1,i} \\ z_{1,i} & z_{2,i} & \cdots & z_{3,i} \end{bmatrix} + t \qquad (2.4.1)$$

The location of the tool tip is calculated using the calibration determined earlier. Recall that the calibration of the tool tip consists of the location of the XPoints and the location of the tool tip at a reference time. Using the method described in section III(c), the rigid transformation between the XPoint at this reference time and the XPoints in the calibration coordinate system is calculated, and then applied to the tool tip location at the reference time.

$$p'_i = Rp'_0 + t \qquad (2.4.2)$$

The transform from physical space to CT space is then applied, resulting in the location of the tool tip in CT space.

$$p'_{i,CT} = Rp'_i + t \qquad (2.4.3)$$

Calculating the transform from physical space to CT space requires the use of fiducial markers. These markers must be visible both in physical and CT space. For this project titanium spheres of 5 mm diameter were used. Titanium is used because it shows up well in CT but is not so dense as to cause image artifacts. The spherical shape is used to facilitate localization in physical space. To complete the calculation, the location of these fiducial markers in both CT space and physical space must be acquired. First the CT points are acquired either manually or from an input file. To acquire the CT points manually, the fiducial markers may be found in the CT image by manipulating the sliders provided, and then clicking on them. A preferred solution, however, is to find the CT

points through an external application and then import the points as a text file. This latter approach will generally lead to more accurate CT points.

Next, the fiducial points in physical space must be acquired. A calibrated tool, called a probe, is used for this process. The tool is shown in Figure 10 in contact with a fiducial marker. As can be seen in the figure, the probe has trackable checkerboards attached to one end. At the other end is a hollow cylinder. The tip of the cylinder is designed so that it fits on a spherical fiducial marker. The probe is held so that the tip is placed on the fiducial marker while the checkerboards are visible by both cameras of the MicronTracker. The MicronTracker then records the location of the marker, and the process is repeated for each marker. In order for the transform to be calculated correctly, the fiducials must be acquired in the same order as they were in CT space. The transform is then calculated using the same method as described in Section 3 of this chapter.

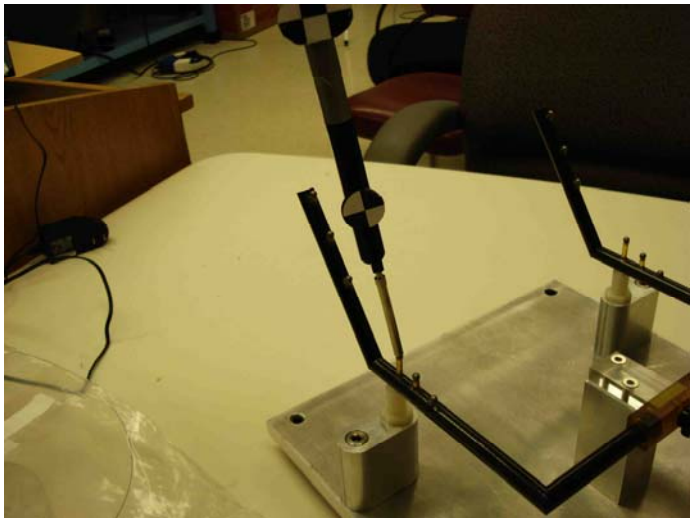

Figure 10: The probe in contact with a fiducial marker. The probe is designed such that the probe tip will always be located in the center of the ball bearing.

5. Disablement

While tracking the location of the surgical drill encompasses most of the computational time of the application, the effort is useless if there is no methodology to disable the surgical drill when its tip enters the designated restricted volume. In order to provide high accuracy, the restricted volume is determined using a distance map through linear interpolation. This approach allows for a sub-voxel approximation of the location of the surface of the restricted volume.

The distance map is provided as an input file to the application. It is obtained from a separate application that segments a CT image into the different anatomical structures that are visible in the CT. From this program, several distance map files are obtained, where each file represent a different anatomical structure in the CT image. Each of these files is an array of binary floating point numbers (floats), and each float corresponds to the value at a certain voxel. This value represents the smallest distance, in millimeters, from the center of that voxel to the anatomical structure.

These files are then fed into a custom MATLAB script, which takes the minimum value of the distance maps per voxel, multiplies the value by 1000, and changes it into an integer value (short). The resulting map then represents the minimum distance, in microns, from the voxel to any of the separate anatomical structures. This is done because all anatomical structures are deemed restricted volumes and should not be contacted by the drill to avoid injuring the patient. The value is changed from a float to a short strictly to reduce the memory requirement of the application.

At this point, each voxel has an integer value that correlates to the minimum distance from the center of the voxel to a restricted volume. However, the location of the tool tip will fall somewhere between the voxel centers. In order to use the distance map, the location of the tool tip in CT space is converted into a continuous voxel index. From this continuous index, the distance from the tool tip to the volume is approximated through linear interpolation. The figure below illustrates a two-dimensional example of linear interpolation.

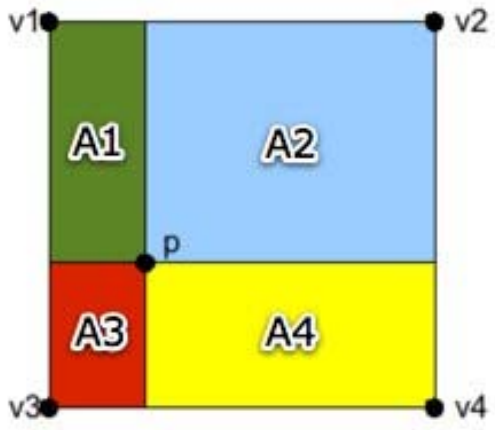

Figure 11: 2-D example of linear interpolation

Each voxel location v_i has a location (x_i, y_i, z_i) and a distance value g_i. The distance value g_p is weighted proportionally by each voxel value according to the opposite area.

$$g_p = \frac{A_1 g_4 + A_2 g_3 + A_3 g_2 + A_4 g_1}{\left(A_1 + A_2 + A_3 + A_4\right)} \tag{2.5.1}$$

Or,

$$g_p = \frac{(x_p - x_1)(y_1 - y_p)g_4 + (x_2 - x_p)(y_2 - y_p)g_3 + (x_p - x_3)(y_p - y_3)g_2 + (x_4 - x_p)(y_p - y_4)g_1}{(x_2 - x_1)(y_1 - y_3)} \tag{2.5.2}$$

Extending Equation (2.5.1) to a 3-dimensional space results in

$$g_p = \frac{V_1 g_8 + V_2 g_7 + V_3 g_6 + V_4 g_5 + V_5 g_4 + V_6 g_3 + V_7 g_2 + V_8 g_1}{(V_1 + V_2 + V_3 + V_4 + V_5 + V_6 + V_7 + V_8)} \tag{2.5.3}$$

If the resulting value is greater than the radius of the drill burr (which is input by the user through the GUI), the point is determined to be outside of the restricted region, while a negative or zero value is determined to be inside of the restricted region. Due to the estimation involved in the linear interpolation and the precision of the camera, an extra 0.25 mm buffer is added to the radius of the burr. If g_p is below this value, the drill is disabled.

To disable the drill, circuitry was designed by Thomas Edwards in July of 2006 to allow a parallel port to interrupt the drill's power [20]. When the application determines that the drill has entered a restricted region, it sends 0x00 through the 16 data pins. To allow the drill to function normally, 0xFF is sent.

CHAPTER III

SOFTWARE ORGANIZATION

The application was written in C++ and relied on the IGSTK library. The organization of this code is discussed in this chapter.

The code is broken up into several classes, with each getting their own header and source code files. Other files are included with these C++ files, and are all held in a project directory. The files in this directory include

- main.cxx
- CT_GUI.fl
- CTDisplay_Implementation.cxx
- CTDisplay_Implementation.h
- CTRegisterProbe.cxx
- CTRegisterProbe.h
- CTPortAccess.cxx
- CTPortAccess.h
- igstkLandmark3DRegisterChris.cxx
- igsktLandmark3DRegisterChris.h
- igstkMicronTrackerChris.cxx
- igstkMicronTrackerChris.h
- CT_Callbacks.h
- bwBox.cxx
- bwBox.h
- CMakeLists.txt
- INSTALLME.doc
- README.txt

The application uses CMake as the build system [21]. CMake is a cross-platform utility that can generate the correct build files on any major platform. For example, it will generate a Visual Studio environment on Windows machines, and on Linux it will

generate a “make” file. CMake uses the file CMakeLists.txt to locate and set the correct include files and linkers. This system removes the need to maintain separate build environments for multiple platforms. Instructions to install and build all dependencies and the application, including the use of CMake, can be found in the document INSTALLME.doc.

As might be expected, main.cxx contains the main program loop. Its main purpose is to instantiate CTDisplay_Implementation and to check for timeouts. During tracking, it also calls methods used to track the surgical tools.

CT_GUI.fl is a fluid file, a file type generated by the FLTK GUI builder. It is a concise way of coding a FLTK GUI, and strongly resembles C++. When building the project, CT_GUI.fl will generate both CT_GUI.cxx and CT_GUI.h for building, and creating the GUI. In addition, it contains empty functions that are used as callbacks. CTDisplay_Implementation inherits from CT_GUI and implements these callbacks.

CTDisplay_Implementation contains the bulk of the code and most of the methods used by the application. Most of the methods contain comments in their source code to explain their functionality; however, some of these methods are explained in further depth below.

- void GetVisibleXPoints()
    - This method obtains the current xpoints by passing two std::maps by reference to the tracker object.
    - One map, CurrentXPointMap, contains a mapping of std::strings to std::vector<double>s. The strings contain the markers name, while the vector contains the XPoint in the form $<x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3>$. The other, xpVisibleMap, maps std::strings to bools, where the strings are again the marker names and the bools represent whether the marker is currently visible.
    - The visible markers are then loaded into a std::vector for use in loops.
- void SetMarkersList()
    - This method sets the list of markers that is associated with a particular surgical tool and is a callback during calibration.
    - If the reference is being calibrated, GetVisibleXPoints is called and these points are saved as the reference location.
    - For any calibration, boxes are added as children to the appropriate indicator group, and then are colored red.
- void FindProbeTip()
    - This method is probably the most important used by the application as it is the way the surgical tool is tracked.
    - The method first calls GetVisibleXPoints(), then uses xpVisibleMap to set the indicators to either red or green depending on whether the tracker can see that tool.

- The position of the reference markers is then obtained from the CurrentXPointMap. These values, along with its original position, are fed into the igstkLandmark3DRegister object.
- The method RequestFindTheTransform() is called, which obtains the transform from the location of the reference now to their position during calibration from the igstkLandmark3DRegister object.
- This transform is then applied to the probe or drill markers that are visible.
- The same process is repeated to find the transform from where the surgical tool markers were during calibration to where they are now.
- This transform is applied to the position of the surgical tool tip found during calibration to find the location of the tool tip now.

The previous listing of methods is limited. Figure 12 displays the methods called from CTDisplay_Implementation during each state of the application, in order to more accurately describe the flow of the application.

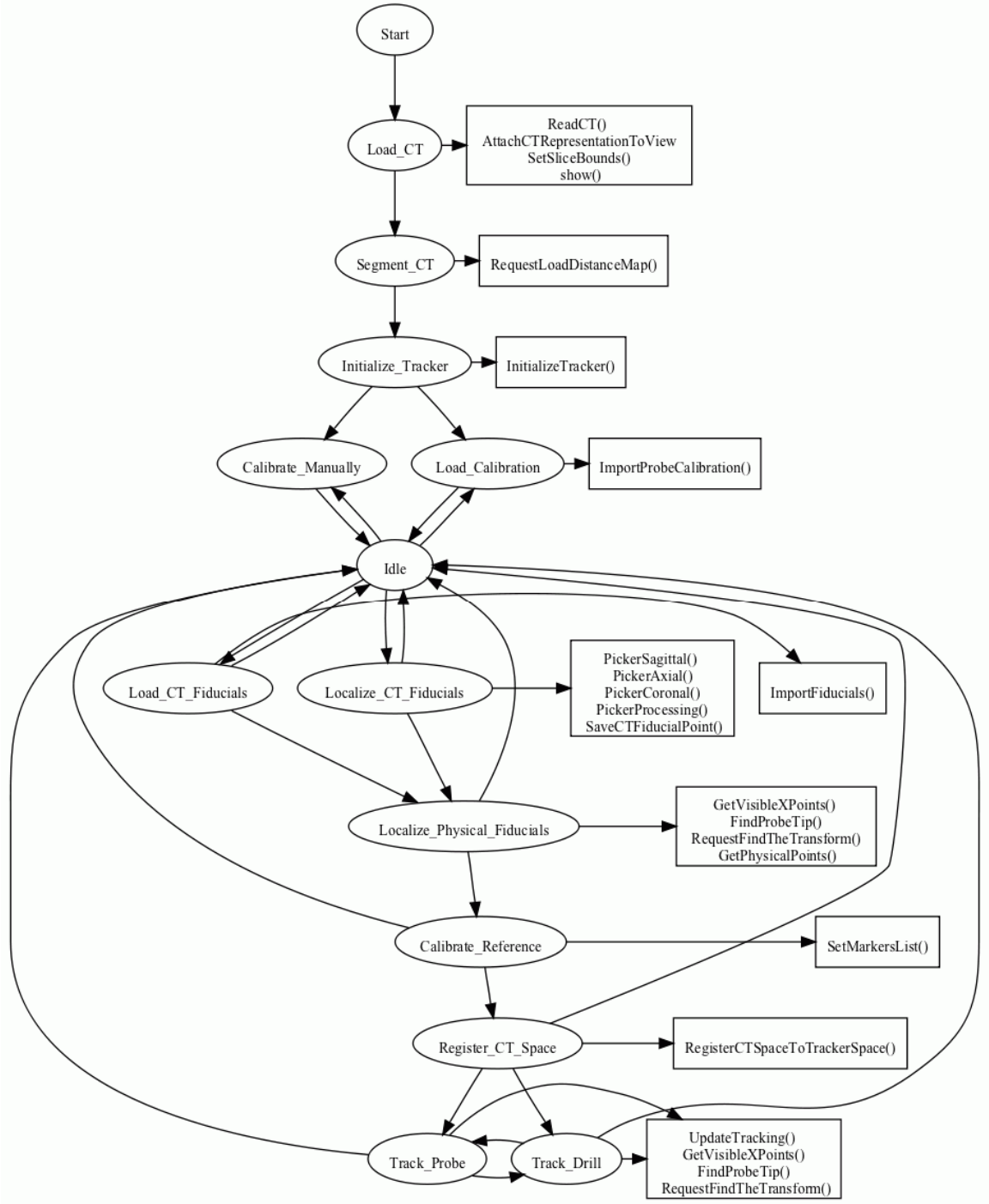

Figure 12: The methods called from CTDisplay_Implementation for each state in the application. The method names are listed in the square boxes.

Additionally, the methods used by during a surgical tool calibration are found in Figure 13.

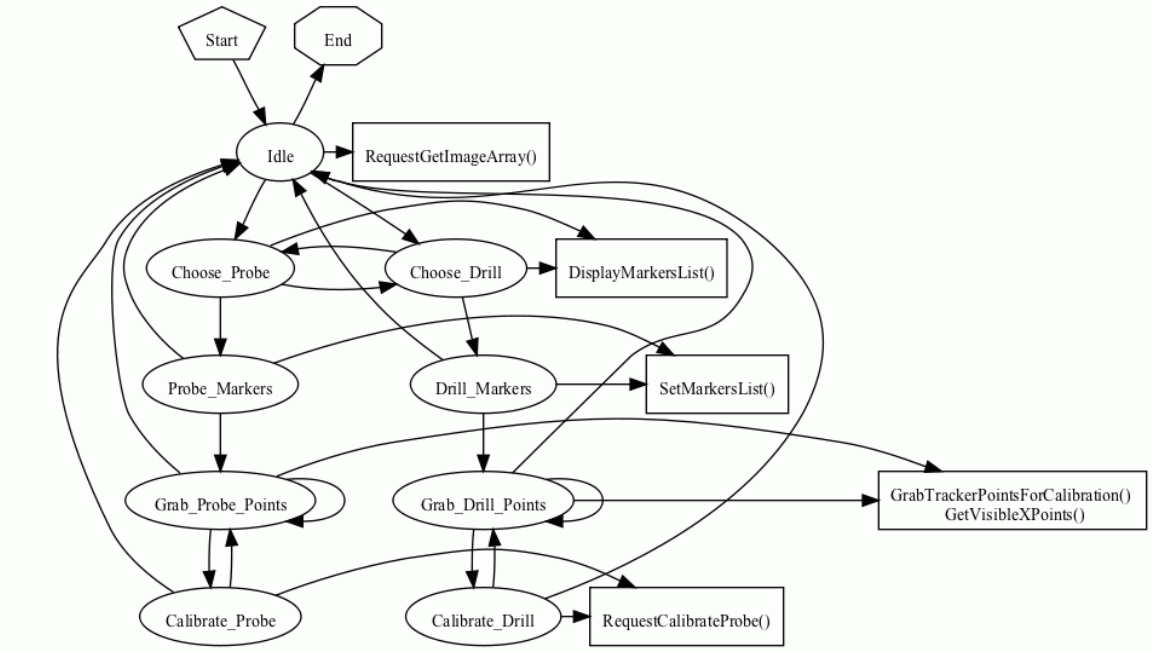

Figure 13: The methods called from CTDisplay_Implementation for each state in the calibration. The method names are listed in the square boxes.

CTRegisterProbe contains methods that are used to calibrate the probe. CTDisplay_Implemenation::GrabTrackerPointsForCalibration() passes the marker points from a given time period to this method. The transform from the *n*th time period to the first time period is calculated, and the resultant transform is added onto a vector. After all samples have been taken, the position of the surgical tool tip is found using the method described in Chapter II, Section 3.

CTPortAccess contains the methods that are used to access the distance map, determine if the drill should remain on or turn off, and then send the correct signal through the parallel port to the drill. It passes a Boolean value back to CTDisplay_Implementation indicating whether the drill has been shut off, which is used to change the color of the drill status indicator.

The classes igstkLandmark3DRegisterChris and igstkMicronTrackerChris are hacked versions of the IGSTK libraries bearing the same name. A method was added to Landmark3DRegister to allow for pass by reference retrieval of transforms instead of using callbacks. This was implemented to prevent an issue that occurs with the callback method utilized by IGSTK when finding the transform many times. MicronTracker had a number of methods added and altered, allowing for the storing and retrieval of the XPoint locations and the visibility status of all markers.

CTCallbacks contains several classes of callbacks. Three of these are used in conjunction with igstkLandmark3DRegister, while another is used to verify that the tracker is calibrated correctly.

Finally, bwBox is a custom FLTK box that is used to display the video output from the tracker. It is used in CT_GUI.fl, defining a box in the calibration window.

CHAPTER IV

VALIDATION METHODS

Validation of the system requires obtaining data about its accuracy, speed, and effectiveness. The methods employed to acquire these results are discussed in this chapter. In order to begin experimentation, a phantom had to be designed and created. Balsa wood was chosen as the material for the phantom. Balsa is soft enough that drilling it away is quite easy for the surgical drill, while it is rigid enough to allow attachment of markers. It is also dense enough to be visible in a CT image.

1. Phantom Description

In our validation approach, the phantoms used are solid blocks of balsa wood, which play the role of human bone that is subject to surgical ablation. The blocks were imaged in CT, and restricted regions, which play the role of vital anatomical structures, were artificially defined in CT space inside the wood. An operator then drilled the blocks while the drill's power was being controlled by our system. The regions lie completely below the wood surface, so no visual feedback is available to the operator of the drill, and hence the only way to avoid striking the restricted regions is through use of the drill disablement system. Before acquisition of the CT, the blocks of balsa were rigidly attached to a fiducial frame (Figure 14), which has been previously described [22,23]. Fiducial markers in the form of spherical titanium spheres of 5 mm diameter are permanently mounted on the frame on plastic posts and serve as fiducial markers. These markers can be accurately localized in

CT using an algorithm that finds the intensity-weighted centroid, and they can also be accurately localized in physical space using the probe described in Chapter II.

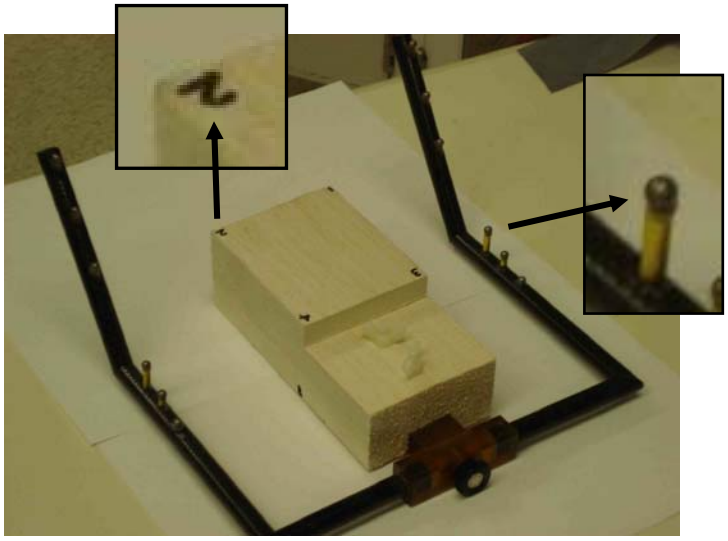

Figure 14: The balsa block attached to the frame. A CT image of this arrangement was taken. The inset at right shows a close-up of one of the titanium spheres used as fiducial markers. Each marker sits on a plastic post that is rigidly attached to the frame. The upper inset shows a close-up of a number at the corner of the block.

a. Experiment One

For the first experiment, a restricted region was defined as a simple rectangular prism that was similar to the block, but smaller. The restricted region was defined using a program written in MATLAB (Mathworks, Natick, MA). In order to define the region, the spatial coordinates in CT space of the corners of the block were determined using third party software (VU Planner, Vanderbilt University, Nashville, TN 2007). These corners were then numbered, as can be seen in Figure 14. The corners were then shifted somewhat toward the center of the block to define the corners of the restricted volume. First, the centroid of the block is determined from all corner points c_i.

$$\bar{c} = \frac{1}{8}\sum_{i=1}^{8} c_i \tag{2.5.4}$$

Then, new corner points p_i are defined to be closer to the centroid by 25 percent.

$$p_i = .25\left(\bar{c} - c_i\right) + c_i \tag{2.5.5}$$

Using these new corner points, the restricted region is defined. Since the regions are denoted to our program by means a distance map, the distance from each voxel to the prism must be calculated. To facilitate these calculations, the planes that contain the six sides of the prism are determined. First, the four corners that are contained in each plane are found. Since the corners were numbered, this process is simple. The equation that defines a plane in a Cartesian coordinate system is

$$Ax + By + Cz + D = 0 \tag{2.5.6}$$

Since there are four points in each plane, the least-squares solution for the plane must be found. To find this we use the distance from a given point to a plane and try and minimize this value. The distance from a point *i* to a plane is given by

$$d_i = \frac{Ax_i + By_i + Cz_i - D}{\sqrt{\left(A^2 + B^2 + C^2\right)}} \tag{2.5.7}$$

[24]. The expression that needs to be minimized is then

$$f\left(A, B, C, D\right) = \frac{\sum_{i=1}^{4}\left(\begin{bmatrix} A & B & C \end{bmatrix} p_i + D\right)^2}{\sqrt{\left(A^2 + B^2 + C^2\right)}}, \tag{2.5.8}$$

where p_i is a column vector [25]. The four points p_i are the four corners of the block that are used to define the plane. To optimize *f*, first the partial derivative of *D* is taken and set equal to zero.

$$\frac{df}{dD} = 2\sum_{i=1}^{4}\left(\begin{bmatrix} A & B & C \end{bmatrix} p_i + D\right) = 0 \tag{2.5.9}$$

Let

$$p_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{2.5.10}$$

Then

$$D = -\sum_{i=1}^{4}\left(Ax_i + By_i + Cz_i\right), \tag{2.5.11}$$

or

$$D = -\left(A\sum_{i=1}^{4} x_i + B\sum_{i=1}^{4} y_i + C\sum_{i=1}^{4} z_i\right) \tag{2.5.12}$$

Let

$$\begin{aligned} x_c &= \sum_{i=1}^{4} x_i \\ y_c &= \sum_{i=1}^{4} y_i \\ z_c &= \sum_{i=1}^{4} z_i \end{aligned} \tag{2.5.13}$$

Substituting *D* back into equation (2.5.8) yields

$$f = \frac{\sum_{i=1}^{4}\left(\begin{bmatrix} A & B & C \end{bmatrix} \left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} - \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \right) \right)^2}{\sqrt{\left(A^2 + B^2 + C^2\right)}} \tag{2.5.14}$$

The partial derivative of f with respect to A, B, and C are then taken and set equal to zero.

$$\begin{aligned} \frac{\delta f}{\delta A} &= \frac{\sum_{i=1}^{4}\left(A\left(x_i - x_c\right)^2 + B\left(x_i - x_c\right)\left(y_i - y_c\right) + C\left(x_i - x_c\right)\left(z_i - z_c\right)\right)}{\sqrt{A^2 + B^2 + C^2}} = 0 \\ \frac{\delta f}{\delta B} &= \frac{\sum_{i=1}^{4}\left(A\left(x_i - x_c\right)\left(y_i - y_c\right) + B\left(y_i - y_c\right)^2 + C\left(y_i - y_c\right)\left(z_i - z_c\right)\right)}{\sqrt{A^2 + B^2 + C^2}} = 0 \\ \frac{\delta f}{\delta C} &= \frac{\sum_{i=1}^{4}\left(A\left(x_i - x_c\right)\left(z_i - z_c\right) + B\left(y_i - y_c\right)\left(z_i - z_c\right) + C\left(z_i - z_c\right)^2\right)}{\sqrt{A^2 + B^2 + C^2}} = 0 \end{aligned} \tag{2.5.15}$$

The denominators are then multiplied out of the equation. In matrix form, the resultant equation can be written as

$$\begin{bmatrix} \sum_{i=1}^{4}\left(x_i - x_c\right)^2 & \sum_{i=1}^{4}\left(x_i - x_c\right)\left(y_i - y_c\right) & \sum_{i=1}^{4}\left(x_i - x_c\right)\left(z_i - z_c\right) \\ \sum_{i=1}^{4}\left(x_i - x_c\right)\left(y_i - y_c\right) & \sum_{i=1}^{4}\left(y_i - y_c\right)^2 & \sum_{i=1}^{4}\left(y_i - y_c\right)\left(z_i - z_c\right) \\ \sum_{i=1}^{4}\left(x_i - x_c\right)\left(z_i - z_c\right) & \sum_{i=1}^{4}\left(y_i - y_c\right)\left(z_i - z_c\right) & \sum_{i=1}^{4}\left(z_i - z_c\right)^2 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = M \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.5.16}$$

The eigenvector that corresponds to the smallest eigenvalue of M contains the optimized values of A, B, and C. To finish defining the planes, D is calculated from Equation (2.5.12).

$$D = -\left(Ax_c + By_c + Cz_c\right) \tag{2.5.17}$$

Now that the planes are defined, the distance from each voxel to the restricted region can be found. A MATLAB script was written to iterate through each voxel point and find the distance to the restricted region. For simplicity, the planes were defined so that the normal vectors from the plane are directed out from the restricted region. As a result, the distance values for all voxels inside the restricted region are negative while the values outside the restricted region are positive. For each voxel, the distance from the point to each of the six planes is calculated using Equation (2.5.7)

From the six distance values calculated, the voxel is then classified into one of four different categories. This step is necessary because simply using the minimum calculated distance would result in inaccuracies as the planes defined in (2.5.6) are not bounded. The first category occurs if the voxel is inside or on the edge of the restricted region. Categories two through four are illustrated in Figure 15.

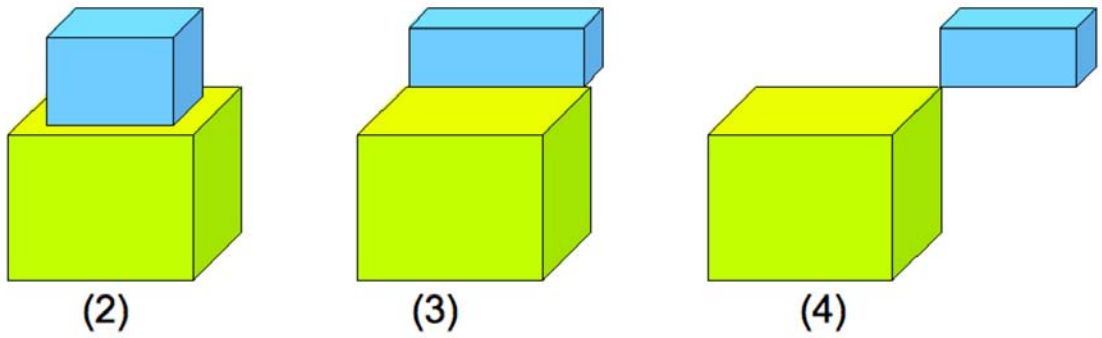

Figure 15: An illustration of the regions represented by categories two through four. In each case, the yellow, larger prism represents the restricted volume, while the smaller prism represents the region in space occupied by the voxel that is being categorized.

A voxel can be segregated into one of the categories by examining the set of six distances calculated and counting the number of positive values in the set. If no positive values are present, the voxel must lie within the restricted volume, and therefore falls into category one. This fact occurs because the planes were explicitly defined so that their normal vectors point away from the volume. In this case, the maximum (least negative) value in the set is chosen as the distance to save to the distance map.

If exactly one positive value is present in the set of distances, the voxel must reside in category two. This categorization can be visualized by drawing a vector from a point in the region of category one to each of the six planes and noticing that only one of these vectors will cross the plane from the outside of the volume. The positive value is then chosen as the distance to write to the distance map.

Using the same visualization method, it can be seen that a voxel whose set of distances contains two positive values falls into category three. In this case, the vectors cross two of the planes from the outside of the volume. However, none of the calculated distances are the correct distance to the volume, as they are found to points on the planes that lie outside of the volume bounds. In this case, the distance to the volume is found by calculating the distance from the voxel to the line defined by the intersection of the two planes. Since the corner points of the volume are already known, the line is defined in terms of the corner points common the two planes. If p_1 and p_2 are the corner points and v is the voxel point, then

$$d = \frac{\left\| (p_2 - p_1) \times (p_1 - v) \right\|}{\left\| p_2 - p_1 \right\|} \tag{2.5.18}$$

Finally, if three positive values are found in the set of distances, the voxel falls into category four. In this case, the intersection of the three planes that have positive distance values is a point. The distance to record into the distance map is simply the distance from the voxel to the intersection point.

$$d = \|p - v\| \tag{2.5.19}$$

These four categories encompass all points in the CT image, so this process creates the complete distance map, which defines the restricted region for the application. An image of a slice through the distance map is displayed in Figure 16. In the middle of the distance map is a volume that is manually placed inside the restricted region. This block is placed strictly to reduce computation time and should have no effect on the performance of the drill, as the drill should never reach that far into the restricted region.

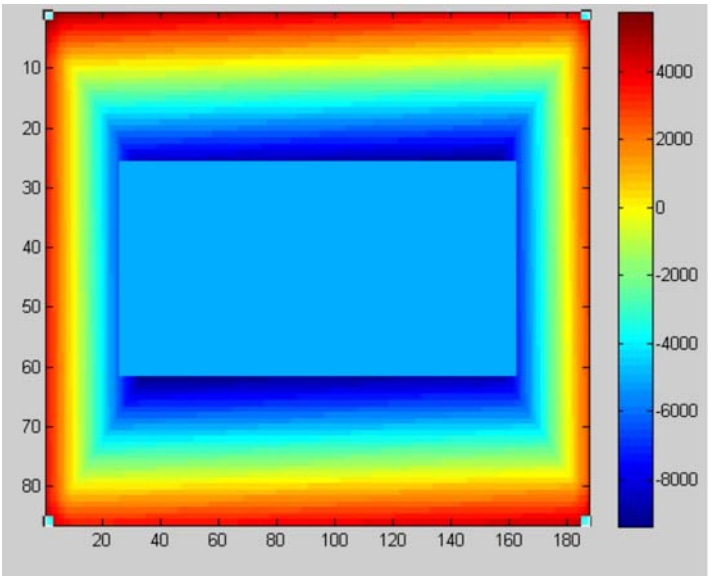

Figure 16: A slice through the distance map. The constant negative value on the inside of the rectangular prism was placed manually to decrease computation time.

b. Experiment Two

While the ability to drill a specific volume is a good way of testing a system, the object of the drill disablement system is to prevent the drill from entering into the restricted region. For this reason, Experiment Two was designed. The goal is to measure the ability of the system to protect unobserved vital anatomical structures. Because accuracy of tracking is known to depend strongly on the distance from the CRF [26], it is desirable to determine the accuracy of the system as a function of distance from the CRF. To test this accuracy, a restricted region that consists of three rectangular prisms was also created. Each prism was parallel to the CRF and each was further away. To create these prisms, the same method as described in Equations (2.5.4) through (2.5.19) was employed. However, the method was used once for each prism, and the minimum value calculated was saved into the distance map. A slice through the distance map is shown in Figure 17.

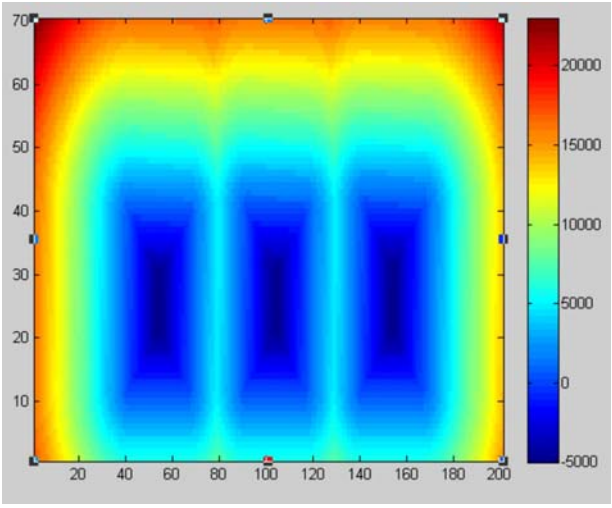

Figure 17: A slice through the distance map. The view is parallel to the CRF, so the CRF would be located to the right of the image.

2. Measurement Technique

In order to acquire data, the drill disablement system was tested on the phantom. A CT image of the balsa block was taken, the balsa was drilled in a manner to test the efficacy of the drill disablement system, and finally a second CT of the resultant balsa is taken in order to quantify the data. In this section the term “frame” is used to refer to the object that contains the fiducial markers and is attached to the balsa block, and the term “stand” is used to refer to a large metal plate that is used to stabilize the CRF and the frame during calibration.

First, a CT image of the phantom is taken with fiducial markers attached, as seen in Figure 14. This CT image is used as reference, and is also used to define the restricted regions, as explained in Section 1 of this chapter. In order to minimize the number of CT scans necessary, one side of the block was used to conduct the first experiment, while the opposite side was used for the second.

The frame that contains the fiducial markers needs to be immobilized in order to correctly register the system. For this reason, the frame is removed from the block of balsa wood and mounted on a heavy metal plate that acts as a stand. In this new position, the location of the fiducial markers can be reliably found. This setup can be seen in Figure 18.

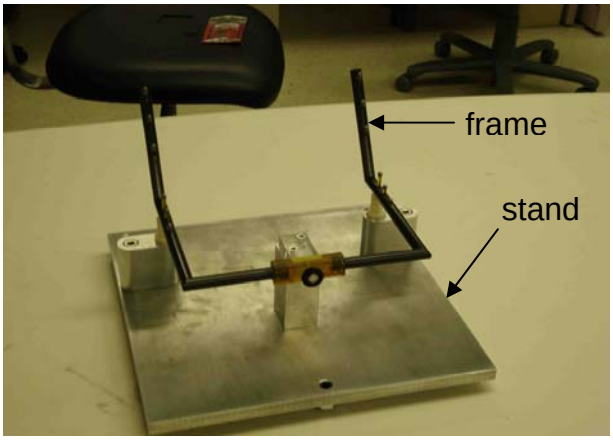

Figure 18: The frame with fiducial markers mounted onto its stand. This setup was used to stabilize the frame in order to more accurately localize the fiducial markers

At this point, the stand is placed in front of the camera in such a way that the surgical tool is visible while it is in contact with each of the fiducial markers. Because the CRF is not visible, it is imperative that the both the stand and the tracker remain stationary during the calibration process. The application is launched, and the process of calibration, segmentation, and localization of the fiducial markers is undertaken.

After all of the fiducial markers are localized, the frame is removed from the stand while keeping the stand stationary. The CRF is then attached to the stand using the same connector that is used to attach the frame. This setup can be seen in Figure 19.

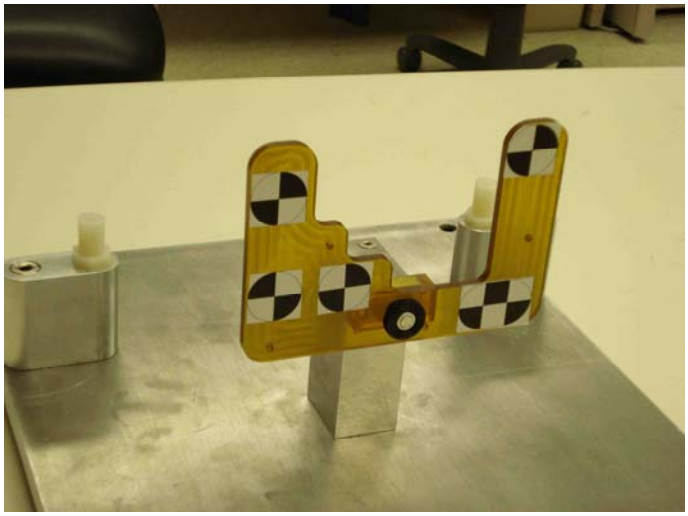

Figure 19: The CRF mounted to the stand. Because the CRF is mounted in the same location as the stand, the spatial relationship between the CRF and the balsa is known.

After the spatial location of the CRF has been recorded, the CRF is removed from the stand. The stand is then set aside, and the CRF is mounted onto the balsa block using the same connector that the frame used while the CT was taken. Because the spatial relationship between the frame, the CRF, and the balsa block remain constant, the balsa block can now be reliably tracked using the CRF. A photograph of the CRF attached to the balsa block can be seen in Figure 20.

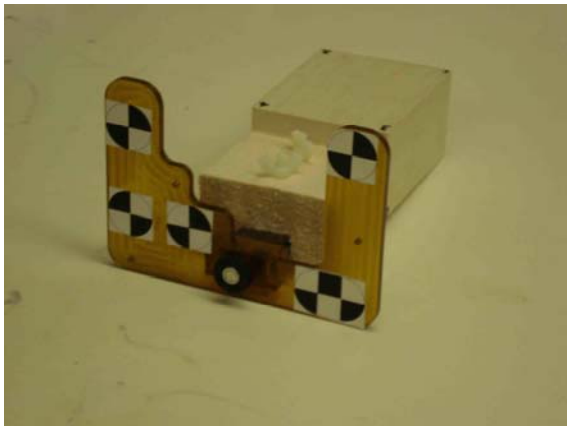

Figure 20: The CRF attached to the balsa block.

Both experiments were performed on the balsa block, one on each side. For the first experiment, the balsa was completely drilled away to expose the entire top half of the restricted region. The assistance of a neuro-otologist (Omid Madjani, Department of Otolaryngology, Vanderbilt University Medical Center) was obtained to complete this experiment. For the second experiment, a regular grid of points was marked on the balsa wood above the restricted regions. Holes were drilled perpendicular to the wood's surface until the drill was disabled or until a maximum depth was reached. This approach led to a simpler analysis of the data. Additionally, the CRF was turned upside down in this second experiment to prevent the tracker's view of the drill during resection. Figure 21 displays the balsa block with the CRF in this alternate orientation.

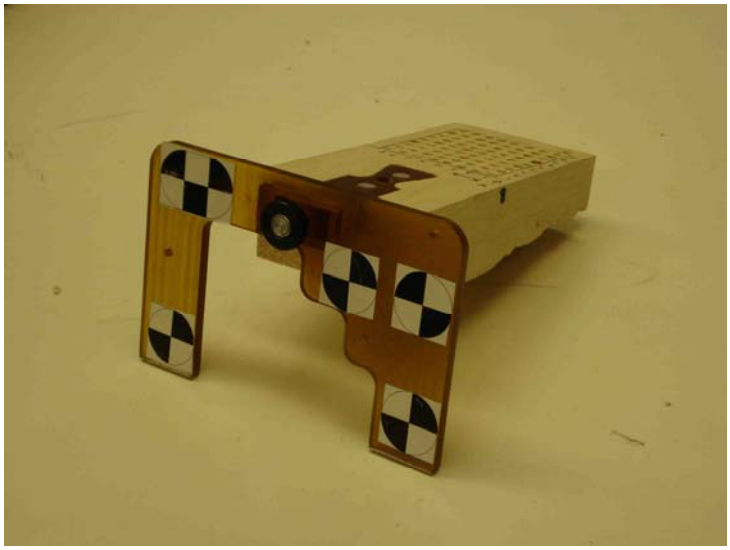

Figure 21: The balsa block with the CRF attached upside-down. In this orientation, the CRF does not occlude the tracker's view of the drill.

An image of the drill experiment in progress can be seen in Figure 22.

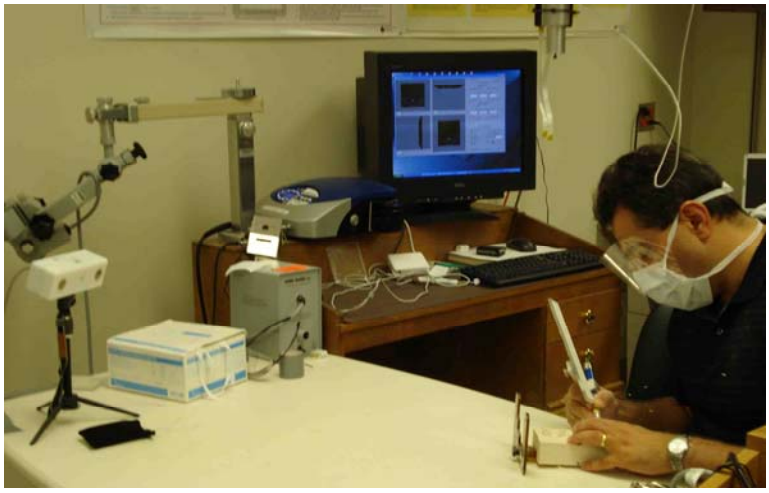

Figure 22: Neuro-otologist using the surgical drill on the balsa block. The top half of the large rectangular prism restricted region was exposed during this experiment.

CHAPTER V

RESULTS

The CT image of the balsa block after drilling was performed was obtained. From a comparison of this image to the original CT, data about the drilling can be acquired. As a necessary step, the transform that registered the CT taken after drilling to the CT taken before drilling was calculated using the fiducial markers on the frame present in both CT images. The method used to calculate this transformation was discussed in Chapter 2, Section 3. To account for the tracker imprecision, processing delays while the drill is moved by the operator, and inertia which prevents instantaneous disablement, the application attempts to stop the drill 0.25 mm from the restricted region. Two experiments were carried out and are described below.

1. Experiment One

The first experiment, completely drilling out the top half of the large phantom, was completed on one half of the balsa block. Drilling was done methodically and slowly (approximately 45 minutes), attempting to drill as much of the balsa away as possible. The CT image of the resultant block was then analyzed.

A raw CT image of the drilled phantom was acquired. The raw image is simply a three-dimensional matrix of integer values that correlate to the gray value present at each voxel in the CT. This image was then loaded into MATLAB, and then cropped so that only the relevant section of the CT remained in memory. A binary mask was then created in

which a zero represented a voxel whose intensity correlated to air and a one represented a voxel whose intensity correlated to balsa wood. From this mask, the "edge" air voxels were chosen as voxels whose mask was zero but which also had an immediate neighbor voxel whose mask value was one. These voxels represent the surface of the balsa wood remaining upon completion of drilling.

The signed distance from the border voxels to the restricted region was then calculated for each voxel. The minimum values of these distances represent the errors made by the disablement system. A value of 0.25 represents perfection, because, as pointed out above, a buffer of this distance was added to the original distance map. Positive values indicate the disablement system stopped the drill outside the region, while negative values indicate the region was violated by the drill. To calculate the distance, the voxel index was translated into a spatial coordinate by multiplying the index by the dimension of the voxel. This point was then transformed into the coordinate system of the original CT that was taken before drilling occurred using the transformation calculated beforehand. The distance map value for this point was then looked up and used as the distance from the edge to the structure.

Figure 23 displays the balsa block after drilling, with the restricted region exposed. The axes used in the following data are also indicated. The origin of these axes is placed at the centroid of the frame's fiducial markers for convenience. The locations of the axes in the figure are placed for readability; they are not accurate with respect to the origin location but are accurate with respect to direction. The centroid of the CRF is located

approximately on the z axis at z = -198 mm. Thus, displacements from the CRF, as opposed to displacement from the fiducial markers on the frame, can be calculated by adding 198 mm to the z coordinates.

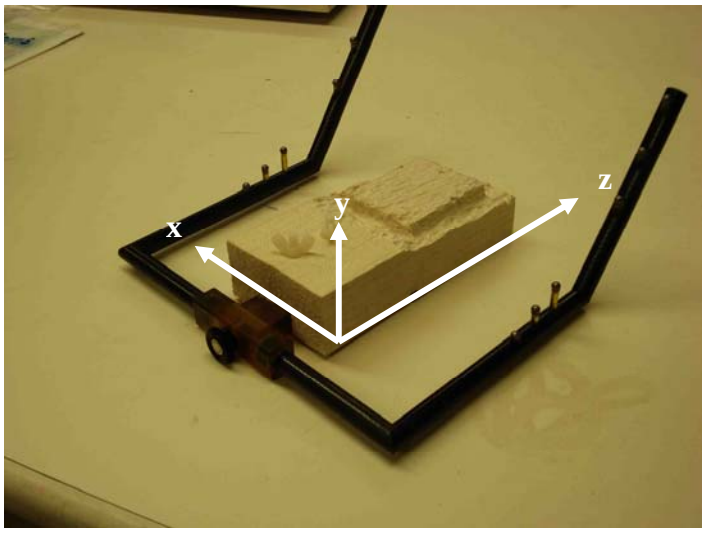

Figure 23: The balsa block after Experiment One. The axes used in the data sets are indicated in the figure.

The signed distances for all voxels directly above the restricted region are the minimum distances for the region and represent the disablement errors. A map of these distances can be seen in Figure 24.

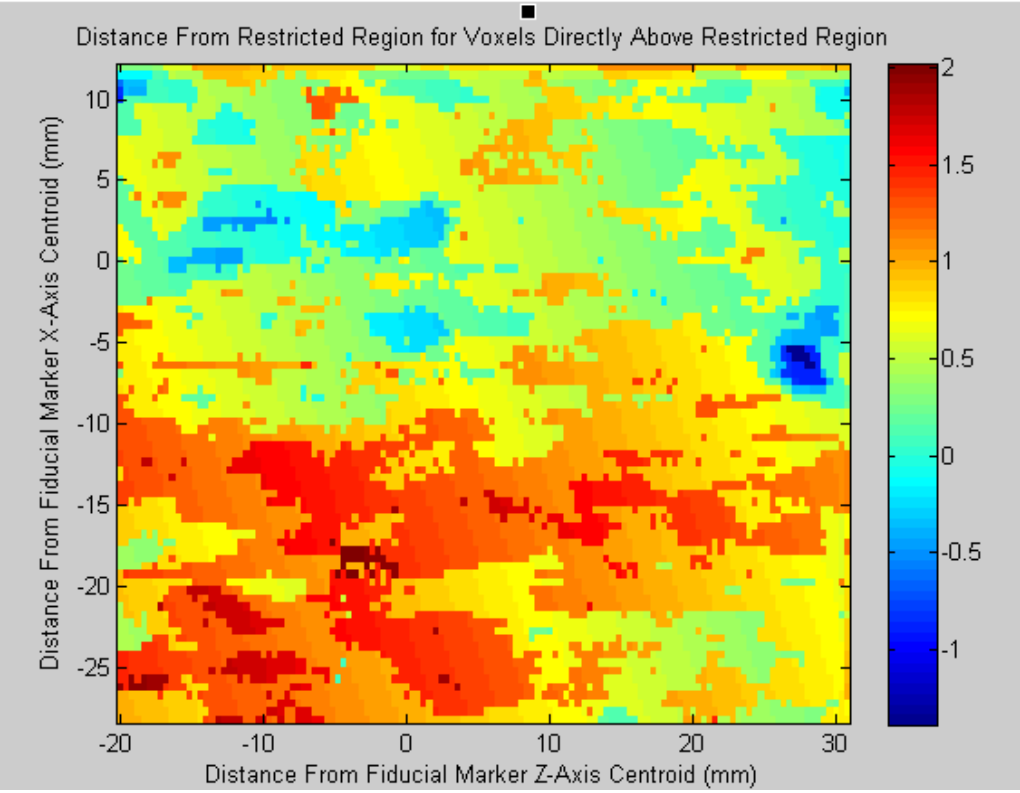

Figure 24: A map of the distance from the surface of the balsa wood after drilling to the restricted region for the area directly above the region. The directions of the *x* and *z* axes are shown in Figure 23. The distance values listed are in millimeters. The centroid of fiducials is at $x = 0$, $z = 0$.

The mean distance value recorded was 0.74 mm, which is 0.54 mm from the perfect distance of 0.25 mm, while the standard deviation was 0.46 mm. The dark blue section, seen around $z = 28$ mm, $x = -7$ mm, represents an area where the drill entered the restricted region. This occurred when the drill slipped and entered the balsa before the system could successfully turn it off. The delay between the drill entering into the restricted region and the drill being switched off was responsible for allowing the drill to enter into the structure. Overall, 5.3% of the surface was found to be inside the restricted region, with an average depth of entry within this 5.3% of -0.24 mm. If the slip is

disregarded, the average depth of penetration is -0.17 mm. Table 1 tabulates some of the important statistics for the entire collection of points.

Table 1: General data about the distance voxels.

Mean Distance	0.74 mm
Maximum Distance	2.02 mm
Min Distance	-1.39 mm
Median Distance	0.73 mm
25th Percentile	0.43 mm
75th Percentile	1.08 mm
Standard Deviation	0.46 mm

The average distance from the surface to the restricted region is 0.74 mm, and 94.7% of the surface lies outside of the restricted region.

Additionally, the distance values were averaged along both the x-axis and z-axis. These plots can be seen in Table 1 and Figure 26. Discussion of these data can be found in Section 3.

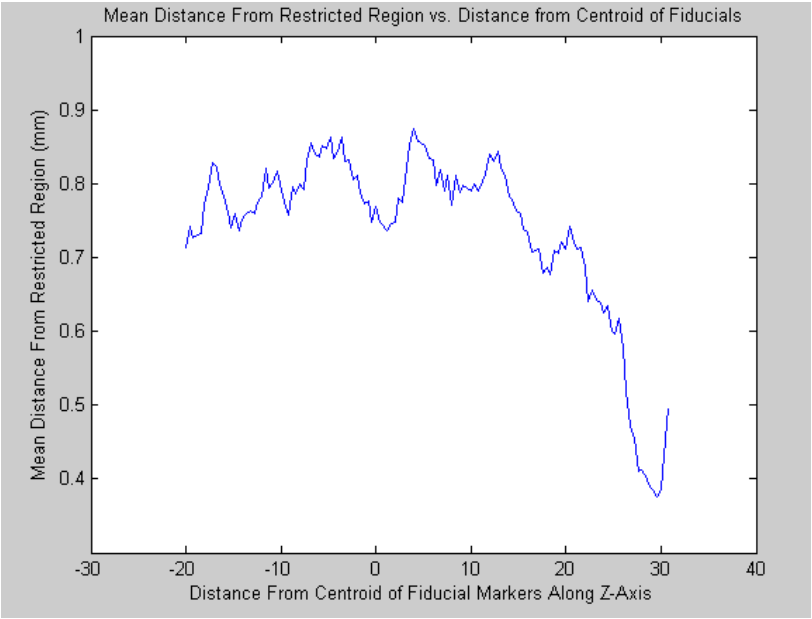

Figure 25: Mean distance from the restricted region as a funtion of z (z is in mm). The directions of the *x* and *z* axes are shown in Figure 23. The centroid of fiducials is at $x = 0$, and $z = 0$.The application seems to allow the drill to get closer to the phantom as it moves away from the centroid.

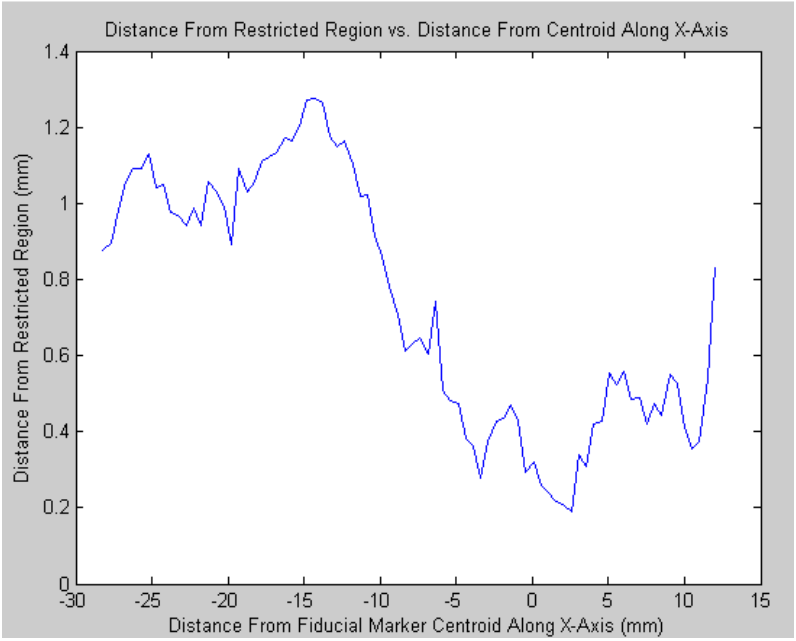

Figure 26: Mean distance from the restricted region as a function of *x*. The directions of the *x* and *z* axes are shown in Figure 23. The centroid of fiducials is at $x = 0$, $z = 0$. As the drill moves away from the centroid of the fiducial markers, the accuracy seems to decrease.

2. Experiment Two

For the second experiment, holes were drilled in a regular array into the balsa block until either the drill was disabled or it reached a specified depth, representing the base of the area to be resected. This experiment was carried out on the opposite side of the balsa block use in Experiment One. Figure 27 shows the balsa block after the drilling for the second experiment. The holes are indexed into a grid to allow for easy presentation of the data.

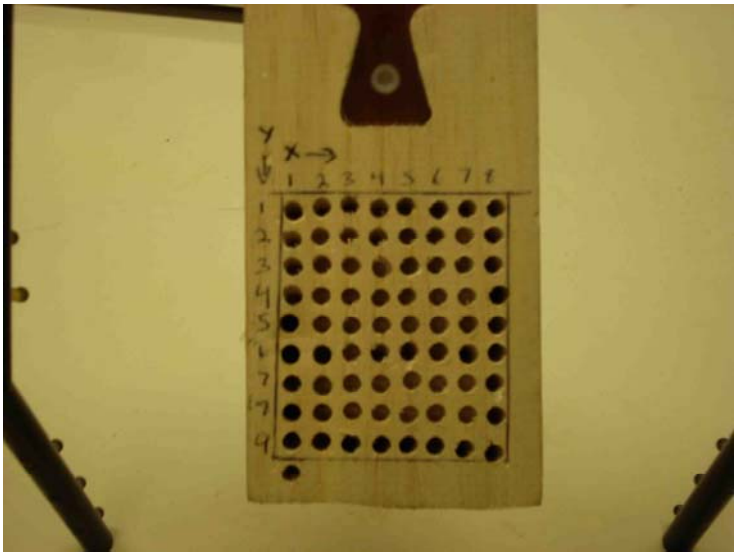

Figure 27: The balsa block after the Experiment Two. The holes are drilled in a regular array and are indexed on the block.

Similar to Experiment One, the raw CT image was cropped and loaded into MATLAB. Again, the edge voxels were determined and the distances from the voxels to the restricted region were looked up in the distance map.

For this experiment, the minimum distance from each hole drilled to the restricted regions was determined. To calculate this distance, each edge voxel was examined. If any of its neighbors had a lower distance value, the lower distance value was assigned to the voxel being examined. This process was repeated until no distance value changed. An image map of these minimum distances as viewed looking into the drilled holes is found in Figure 28.

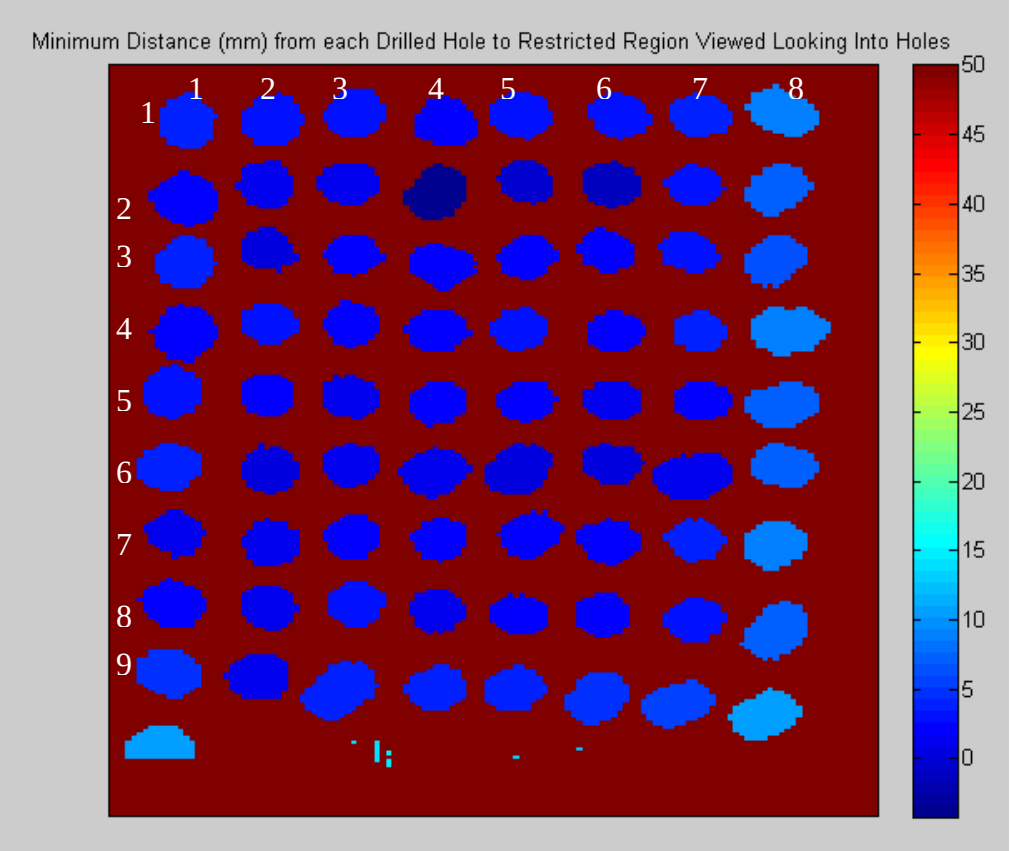

Figure 28: An image map of the minimum distance from each drilled hole to the restricted regions. The image is viewed looking down into the holes. A value of 50 mm represents regions where the balsa wood was not ablated.

The holes located in rows 1 and 8, and in columns 1 and 9 did not come near the restricted region, and therefore the drill did not stop during resection. These holes were used to verify that the drill would allow drilling through non-restricted regions. In no case did the system disable the drill in these columns. For the purposes of quantifying the performance of the drill disablement, these holes were ignored. Distance values for the remaining holes are recorded in Table 2. The index value can be referenced from Figure 28. Values are omitted for six holes on at the edges of the restricted region. The drill was

not disabled in these holes because, as was determined from the second CT, it was aimed away from the region and therefore would never enter it.

Table 2: The minimum distance value for each drilled hole.

Index	2	3	4	5	6	7	8
2	1.19	-0.11		1.5	0.25	1.1	0.8
3	0.65	1.3	1.8	1.0	0.9	1.6	2.3
4		1.4	1.7	1.6	0.4	1.6	0.9
5		1.3	2.1	1.5	-0.13	1.7	1.9
6		1.7	1.0	1.0	-0.23	1.2	1.4
7	2.27	2.2		1.7	1.1		2.7
Mean	1.37	1.31	1.69	1.38	0.38	1.44	1.67

A blank cell indicates that the drill did not stop in that hole. The mean for each column of holes is listed on the last row.

The mean value for all of the distances is 1.3 mm while the standard deviation was 0.69 mm. With 36 samples, the standard error of the mean is $0.69/\sqrt{36}$ = 0.11 mm.

3. Discussion

Without a doubt, Experiment One returned very exciting results. The drill, for the vast majority of the drilling, did not enter into the prohibited region, while allowing it to get within a millimeter of the surface. Additionally, aside from the one point where the drill slipped and drilled into the restricted region, the drill did not enter into the restricted region by more than about 0.25 mm.

However, Experiment Two did not yield results that were as promising. While the drill never entered into the restricted region, the system tended to disable the drill too far from

the restricted region, stopping at a mean of 1.3 mm from the restricted region. While the vital anatomy is successfully protected, during a mastoidectomy, it may be necessary to drill inside this boundary in order to accomplish a complete resection.

Figure 25 and Figure 26 both seem to suggest trends with respect to tracking accuracy for Experiment One. As the drill moves in the z-direction, the application seems to allow the drill closer to the restricted region. Conversely, as the drill moves away from the centroid in the x-direction, the application seems to stop the drill further from the restricted region. However, this behavior may not be indicative of the general behavior of the accuracy of the system, because it is based on only one registration between image and physical space. Repeated experiments will doubtless exhibit different behaviors. Previously published results on tracking indicate that, as the drill moves away from the CRF, the accuracy would be expected to decrease [26]. Because of the limited range of distances from the CRF (180 mm to 220 mm) in these experiments, this trend would be expected to be small and therefore it not possible to test it without more experiments.

As with all data, the data presented above contains sources of error. The largest source of error comes from the voxel size of the CT image used to validate the drill. These voxels measure 0.5 by 0.5 by 0.4 mm. Due to this size, it is impossible to be accurate in our validations to within 0.35 mm. In the case of Experiment One, many of the distances were within the width of one voxel of the surface. This error could mean that many of the voxels that were calculated to be inside the restricted region were in reality outside

the region. Additionally, many of the points outside the region could either actually be inside the region or located further away.

Another source of error is due to the nature of the balsa wood. When ablating the wood, balsa fibers, unlike bone, tended to shred, leaving a rough surface on the balsa block. This shredding made it extremely difficult to differentiate between the balsa and air on the CT image at the edges. This problem was exacerbated by the closeness in intensity that air (-1024) and balsa (about -900) share on a CT. For this reason, there is an uncertainty in choosing the edge voxels, and many artifacts were present when masking the CT.

Yet another source of error occurs during the tracking process. Part of this error is caused by inaccuracies in locating the probe tip during registration, causing registration error. The target registration error (TRE) found during the probe calibration was found to be 0.4 mm. Additionally, error in calibrating the drill itself directly causes inaccuracies during its tracking. The TRE found during the drill calibration was 0.45 mm. Fitzpatrick et. al. provide a good explanation of TRE [27]. Furthermore, according to Claron's documentation [18], the MicronTracker is accurate only to 0.25 mm, introducing additional error.

With all of these errors present, it is remarkable how well the system worked in its first set of tests. In Experiment One only 5.8% of the surface of the restricted region was violated and 65% of the disablement errors were in a range of zero to 1 mm. In

Experiment Two, the registration accuracy was clearly not as high, but the violation rate was still only 8.5% and 78% of the errors were in the range from zero to 2 mm. A 2 mm error and an 8.5% violation rate may have a large impact on the results for some procedures. However, the feasibility of this drill can be said to have been successfully tested and verified.

CHAPTER VI

CONCLUSIONS

A drill disablement system was successfully designed, created, and tested. The drill system was used to resect a balsa block phantom with "restricted regions" defined inside to imitate patient anatomy. From the tests, several conclusions can be drawn.

Firstly, the software is stable and easy to understand. Throughout thorough testing, the application experienced no crashes. Use of the application is intuitive and the indicators present the status of the system in an obvious way. However, additional feedback in the GUI, such as the fiducial registration error (FRE) during registration, could prove to be useful. Additionally, an attempt should be made to reduce the delay between moving the drill and having the application update. This delay contributes to error during drilling, and reducing it should improve the accuracy of the system.

Secondly, the feasibility of the drill disablement system has been confirmed. The drill was consistently disabled as it neared the restricted regions inside the phantom. In Experiment One, the drill was stopped, on average, 0.74 mm from the restricted region. The drill entered the restricted region for only 5.3% of the area drilled, and only entered an average of 0.24 mm in these areas. These results are extremely promising and confirm that the drill disablement system could be successfully used in a surgical environment. The drill was not quite as accurate during Experiment Two. However, when considering

the error inherent in the system, the results were still accurate enough to encourage further development of the system.

Despite these positive results, it would be extremely advantageous to perform further experiments using the system. Each test consisted of one only registration. Error in the registration transformation affects the entire test. To gather good statistical evidence on the performance of the drill, additional tests should be performed. Additionally, these tests should reveal trends that can be compared with earlier published trends [26].

In conclusion, the drill disablement system shows promise, and its development should be continued. With this development, it could prove to be an extremely useful safeguard during mastoidectomies.

CHAPTER VII

FUTURE WORK

To continue development of the drill disablement system, several new directions could be pursued to enhance the system. First, additional experimentation should be performed on the system to further quantify its performance. Further detail about its behavior could lead to adjustments that would enhance the accuracy of the system. Additionally, these experiments could be used to quantify the time delay of the system. It would be particularly interesting to compare delay in the application when the CT image is resliced during tracking as compared to when that functionality is turned off. The CT image requires a large memory footprint due to its large size, so the system may slow down when accessing the image.

As a corollary to determining the time delay of the system, methods could be implemented to reduce this time delay as much as possible. First, multithreading could be employed with the communication between the tracker and the application to reduce the delay that occurs when obtaining information from the MicronTracker. Secondly, the methods used to put both the CT image and camera output on the screen could also be put in their own thread. This approach could reduce computation time for the main thread, especially if used in conjunction with a modern multi-core computer.

Lastly, the system should be adapted for other, more accurate tracking systems. The Claron MicronTracker has many advantages, such as its low price point and its passive nature. However, other trackers, such as the IR-based Polaris tracker (Northern Digital, Inc. Waterloo, Ontario), are already commonly used in operating rooms. These systems have a wider viewing area and can resolve positions more accurately.

REFERENCES

1. E. A. Spiegel, H. T. Wycis, M. Marks and A. L. Lee "Stereotaxic apparatus for operation of the human brain", *Science,* **106,** No. 2754, 349–50 (Oct 10, 1947).

2. B. A. Kall, P. J. Kelly, S. J.. Goerss, "Interactive stereotactic surgical system for the removal of intracranial tumors utilizing the CO_2 laser and CT-derived database", *IEEE Trans. Biom. Eng.*, **BME-32**(2), 112-116 (Feb 1985).

3. P. J. Kelly, "Computer-assisted stereotaxis: New approaches for the management of intracranial intra-axial tumors", *Neurology*, **36**(4), 535—541, (Apr 1986).

4. D. W. Roberts, J. W. Strohbehn, et al., "A frameless stereotaxic integration of computerized tomographic imaging and the operating microscope", *J. Neurosurgery*, **65**, 545-549 (1986).

5. C. R. Maurer*, Jr., J. M. Fitzpatrick, M. Y. Wang*, R. L. Galloway, Jr., R. J. Maciunas, G. S. Allen, "Registration of head volume images using implantable fiducial markers", *IEEE Transactions on Medical Imaging* **16**, 447–462 (Aug 1997).

6. R. F. Labadie, O. Majdani, J. M. Fitzpatrick, "Image-guided technique in neurotology", *Otolaryngology Clinics of N. Amer*. **40**(3), 611-24 (Jun 2007).

7. T. M. Peters, "Image-guidance for surgical procedures", *Phys. Med. and Biol*., **51**, R505-R540 (2007).

8. A. Jurik, "Drill disablement via image-guided position feedback: Version 2.0", Report for Independent Study, Dept. Elec. Eng. and Comp. Sci, Vanderbilt University, Nashville, TN (May 2007).

9. R. F. Labadie, O. Majdani, J.M. Fitzpatrick. "Image-Guided Technique in Neurotology". *Otolaryngologic Clinics of North America,* **40,** 611-624 (2007).

10. E. Mein, A. Alaani, R.V. Jones, "Consent for mastoidectomy: A patient's perspective. *Auris, Nasus Larynx,* **34** 505-509 (2007).

11. G. Strauss, K. Koulechov, et al., "The Navigation-Controlled Drill in Temporal Bone Surgery: A Feasibility Study". *The Laryngoscope,* **117,** 434-411 (2007).

12. K. Cleary, IGSTK team, "IGSTK: An Open Source C++ Software Library" Gaithersburg, MD: Signature Book Printing (2007)

13. NLM Insight Segmentation and Registration Toolkit, http://www.itk.org/

14. The Visualization Toolkit, http://www.vtk.org

15. Fast Light Toolkit, http://www.fltk.org/

16. Claron Technology. “MicronTracker 2.8 Developer’s Manual”, 5-10 (September 2007)

17. Digital Imaging and Communications in Medicine, http://medical.nema.org/

18. Claron Technology. “MicronTracker 2.8 Developer’s Manual”, 22 (September 2007)

19. Berthold K.P. Horn. “Closed-Form Solution of Absolute Orientation using Unit Quaternions. *“Journal of the Optical Society of America,* **106**, 629 (April 1987).

20. Thomas Edwards. “Drill Cut-Off Circuitry” Vanderbilt University, Nashville, TN (June 2006).

21. CMake Cross Platform Make, http://www.cmake.org/

22. R. F. Labadie, R. J. Shah, S. S. Harris, E. Cetinkaya, D. S. Haynes, M. Fenlon, S. Juscyzk, R. L. Galloway, J. M. Fitzpatrick, “Submillimetric Target-Registration Error using a Novel, Non-Invasive Fiducial System for Image Guided Otologic-Surgery”, J. Computer-Aided Surgery, **9(4)**, 145-153 (December 2004).

23. R. F. Labadie, P. Choudhury, E. Cetinkaya, R. Balachandran, D. S. Haynes, M. Fenlon, S. Juscyzk, J. M. Fitzpatrick, “Minimally-invasive, image-guided, facial-recess approach to the middle ear: demonstration of the concept of percutaneous cochlear access in vitro”, Otology and Neurotology, **26**, 557-562 (July 2005).

24. E. W. Weisstein, “Point-Plane Distance.” From *Mathworld*—A Wolfram Web Resource. http://mathworld.wolfram.com/Point-PlaneDistance.html.

25. Y-M Li, J.X. Zhang, “Java Graphics Programming” http://www.infogoaround.org/BookSite.html

26. J. B. West and C. R. Maurer, Jr., “Designing optically tracked instruments for Image-guided surgery”,IEEE Trans. Med. Im., vol. 23, no. 5 (May 2004).

27. J. M. Fitzpatrick, D.L.G. Hill, C.R. Maurer Jr. “Image Registration”. Medical Image Processing, Volume II of the Handbook of Medical Imaging, M. Sonka and J. M. Fitzpatrick, ed., SPIE Press. 447-513 (2000).