The Precision Freehand Sculptor

A Robotic Tool for Less Invasive Joint Replacement Surgery

Gabriel Brisson

CMU-RI-TR-08-18

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh Pennsylvania 15213

March 2008

Carnegie Mellon

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

Thesis

The Precision Freehand Sculptor
A Robotic Tool for Less Invasive Joint Replacement Surgery

Gabriel Brisson

Submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the field of Robotics

ACCEPTED:

[illegible]
Takeo Kanade Thesis Committee Chair

May 21, 2008
Date

[illegible]
Reid G. Simmons Program Chair

June 6, 2008
Date

APPROVED:

[illegible]
Randal E. Bryant Dean

June 9, 2008
Date

Abstract

This thesis presents a tool for less invasive joint replacement surgery. Although many surgical procedures have been converted to endoscopic "keyhole" approaches, joint replacement incisions have changed little in 30 years. Recent efforts to adapt conventional joint replacement instrumentation for less invasive approaches have demonstrated improved short-term outcomes. However, the procedures are more challenging to use accurately and are only suitable for highly skilled surgeons.

The Precision Freehand Sculptor (PFS) is a handheld intelligent tool designed to enable less invasive joint replacement surgery. A small rotary blade at the tip of a long, slender nose allows the surgeon to shape the bone to accept the implant. The blade can extend and retract behind a guard under computer control. As the surgeon moves the tooltip freehand over the surface of the bone, the computer extends and retracts the blade so that only the appropriate material is removed.

An optical tracking camera continuously monitors the 3D position of infrared markers attached to the tool and bone. The PFS computer compares the blade's location on the bone to the *target shape* which it has been instructed to cut. It retracts or extends the blade accordingly. The target shape is specified to mate well with the implant and position the implant for proper biomechanics.

The PFS has the potential to make less invasive joint replacement accessible to more surgeons without sacrificing accuracy. The computer controlled blade ensures an accurate cut even if the tip of the tool is obscured from view. The long, slender nose is ideal for operating through small incisions. A computer display provides additional guidance to the surgeon when visibility is limited.

The biggest technical challenge in developing the PFS was cutting accurately enough. This thesis describes how the PFS predicts user motion so that it can begin retraction early to compensate for sensing and actuation latency. We also describe potential sources of inaccuracy and measure them experimentally. Identification of the largest sources of error will guide future development. Examining potential error sources also enhances our understanding of the PFS and can guide design of future PFS tools for other applications.

Acknowledgements

I would like to thank the entire committee for enormous patience, for it has taken two years since my defense to arrive at this final document.

Thanks to Takeo Kanade for always steering me toward rigorous engineering and application of the scientific method; and for many bits of wisdom on how to write papers, posters, and talks.
Branko Jaramaz always made time to talk, and provided more day-to-day guidance of my efforts.
Tony Digioia was invaluable in providing a surgeon's perspective to ensure that the PFS actually addresses the issues important to surgeons.
Russel Taylor helped me to sharpen the justifications for the PFS, and I am very grateful for the detailed level of feedback that he provided on the document.

During my first years at the Robotics Institute, I was also advised by Hagen Schempf. From him I learned some very important things about mechanical design, which have been useful in the PFS project.

Thanks to my wife Vanessa, who not only put up with the years of writing, but also produced many of the figures, including all those that required artistic talent.

During the two years leading up to my thesis defense, I was at Blue Belt Technologies, a startup company built around commercialization of the PFS. Blue Belt provided the manpower to develop the infrastructure necessary for this project. Vic Eggenberger did mechanical design of the final PFS tool. Ben Hollis and Jason Cipriani were programmers, and both introduced important techniques in the OpenGL implementation. Costa Nikou adapted the HipNav tools to build a complete toolchain for the PFS, including surgical planning and registration.

Table of Contents

Abstract 3

Acknowledgements 5

Table of Contents 7

Chapter 1. Introduction 12

1.1 Less Invasive Surgery and Computer Assisted Surgery in Joint Replacement 13

1.2 Cutting Accurately 15

1.3 Thesis Overview 16

Chapter 2. Towards Robotic Tools for Less Invasive Joint Replacement Surgery 19

2.1 Joint Replacement Surgery: Task Description 20

2.2 Less Invasive Joint Replacement Using Conventional Orthopedic Tools 22

2.3 Computer Assisted Orthopedic Surgery (CAOS) 25

2.3.1 Active Robot Arms 25

2.3.2 Autonomy-Passive Robots 27

2.3.3 Mechanically-Passive Robots 27

2.3.4 Limited Footprint Robots 28

2.3.5 Navigation in Joint Replacement Surgery 29

2.3.6 Advantages of Semiactive CAOS Systems: Safety and Synergy 30

2.4 Application of Robots and Navigation to LIS 31

Chapter 3. The PFS: A New Tool for Less Invasive Joint Replacement 33

3.1 PFS Overview 33

3.2 PFS System Components 35

3.2.1 Handheld tool 35

3.2.2 Workpiece Model 36

3.2.3 Computer Display 36

3.2.4 Blade Control Software 37

3.2.5 Position Tracking System 38

3.3 Advantages of PFS 42

3.4 Systems Similar to PFS 44

Chapter 4. A Target Procedure 45

4.1 PFS-specific Unicondylar Knee Replacement (UKR) 46

4.1.1 Materials and Setup ... 46
4.1.2 Approach Angles and Order of Cuts ... 47
4.1.3 Tracking Arrangement ... 51
4.1.4 Finishing the Cuts ... 51
4.1.5 Additional Concerns ... 52
4.1.6 Surgical Scenario ... 53
4.2 Efficiency Requirement ... 54
4.3 Accuracy Requirement ... 55
4.3.1 Requirements for Fit Accuracy ... 55
4.3.2 Fit Accuracy and Conventional Instrumentation ... 57
4.3.3 Requirements for Positioning Accuracy ... 58
4.3.4 Summary of Accuracy Requirement ... 60
Chapter 5. The PFS Handheld Tool Mechanism ... 62
5.1 Initial Investigations with Unactuated PFS Mockup ... 63
5.2 Blade and Drive System ... 64
5.3 Guard Design for LIS ... 65
5.4 Cutting Control by Extension and Retraction ... 67
5.5 Optical Calibration Routine ... 73
5.6 Electronics and Wiring ... 75
Chapter 6. Workpiece Modeling ... 78
6.1 The Heightfield Model ... 78
6.2 Updating the Model ... 80
6.3 Graphical Display ... 85
6.3.1 3D View ... 86
6.3.2 Cross Section Views ... 87
Chapter 7. Blade Control for Accurate Cutting ... 90
7.1 Predicting Future Tool Positions (Extrapolate Position and Snap to Surface). 93
7.1.1 Tool Position Extrapolation ... 93
7.1.2 Tool position snap-to-surface ... 95
7.2 Allowable Blade Extension ... 104
7.3 Prediction of Multiple Timesteps ... 119
7.4 Dynamic Constraint ... 119

7.5 Choose Extension Direction ... 121
7.6 Timing Considerations ... 124
Chapter 8. Framework for Understanding PFS Cutting Process ... 128
8.1 A Model for Understanding Cutting Error ... 129
8.1.1 A Limit for Worksurface Slope Based on the Extension Multiplier α ... 129
8.1.2 A Limit on Cutting Error Based on Limited Worksurface Slope ... 134
8.2 Applications of the Cutting Error Model ... 139
8.2.1 Application of the Model to PFS Design Parameter Selection ... 140
8.2.2 An Improvement to the Cutting Algorithm ... 145
Chapter 9. Experimental Validation of Cutting Accuracy ... 149
9.1 Foam Block Experiments ... 149
9.1.1 Measured Accuracy ... 150
9.1.2 Cutting Time ... 155
9.1.3 User Velocity and Acceleration ... 156
9.1.4 Position Prediction Error ... 158
9.1.5 Typical Retraction Distances ... 159
9.1.6 Blade Response and Latency ... 160
9.2 Small Incision Experiments ... 161
9.2.1 Measured Accuracy ... 162
9.2.2 Cutting Time ... 163
9.2.3 User Velocity and Acceleration ... 163
9.2.4 Position Prediction Error ... 165
9.2.5 Typical Retraction Distances ... 166
9.3 Practical Experience Gained ... 167
9.4 Bug Discovered ... 168
Chapter 10. Analysis of Cutting Error ... 170
10.1 Analysis of Execution Error ... 175
10.1.1 Error due to Missed Optotrak Frames ... 180
10.1.2 Error due to Prediction ... 182
10.1.3 Error due to Blade Response Time ... 194
10.1.4 Error due to Not Cutting Enough ... 198
10.1.5 Summary of Execution Error ... 199

10.2 Analysis of Modeling Error 201
10.2.1 Tracking, Calibration, and Registration 204
10.2.2 Blade Positioning 212
10.2.3 Software Rate 219
10.2.4 Data Synchronization 220
10.2.5 Heightfield Resolution 221
10.2.6 Summary of Modeling Error 222
Chapter 11. Evaluating the Usefulness of Prediction 224
11.1 Simulating the Response of a Simple (Non-predictive) Algorithm on Recorded Data 224
11.2 Evaluating Algorithms Based on Cutting Efficiency 226
Chapter 12. Conclusion 229
12.1 Contributions 229
12.2 Future Work 232
12.2.1 Suggested Improvements for Increased Accuracy 232
12.2.2 Ongoing Work 233
References 234

Chapter 1. Introduction

The Precision Freehand Sculptor (PFS) is a handheld intelligent tool (Figure 1-2) designed to enable less invasive joint replacement surgery. A small rotary blade at the tip of the PFS allows the surgeon to shape the bone to accept the implant. The blade can extend and retract under computer control. As the surgeon moves the tooltip freehand over the surface of the bone, the computer extends and retracts the blade so that only the appropriate material is removed.

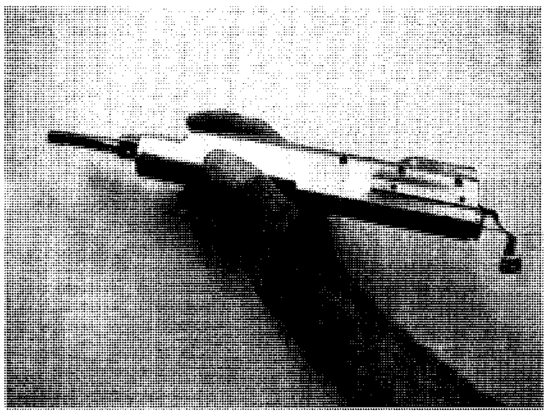


Figure 1-2: PFS Mechanical Prototype

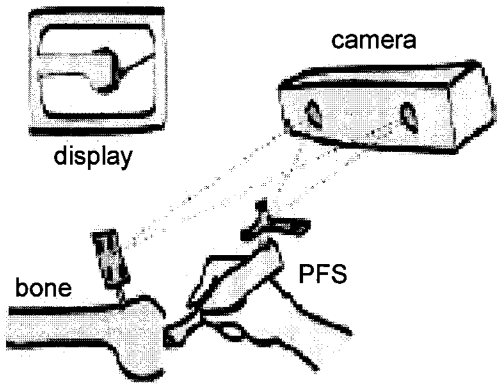


Figure 1-2: PFS Mechanical Prototype Tip Close-up

An optical tracking camera continuously monitors the 3D position of infrared markers attached to the tool and bone (Figure 1-2). The PFS computer compares the blade's location on the bone to the *target shape* which it has been instructed to cut. It retracts or extends the blade accordingly. The target shape is specified to mate well with the implant and position the implant for proper biomechanics.

The goal of the PFS is to enable less invasive joint replacement surgery. Although many surgical procedures have been converted to endoscopic "keyhole" approaches, joint replacement incisions have changed little in 30 years. Recent efforts to adapt conventional joint replacement instrumentation for less invasive approaches are more challenging to use accurately and are only suitable for highly skilled surgeons.

The PFS has the potential to make less invasive joint replacement accessible to more surgeons without sacrificing accuracy. The computer controlled blade ensures an accurate cut even if the tip of the tool is obscured from view. The long, slender nose is ideal for operating through small incisions. A computer display provides additional guidance to the surgeon when visibility is limited.

The biggest technical challenge in developing the PFS was cutting accurately enough. This thesis describes how the PFS predicts user motion so that it can begin retraction early to compensate for sensing and actuation latency. We also describe potential sources of inaccuracy and measure them experimentally. Identifying the largest sources of error will guide future development. Examining potential error sources also enhances our understanding of the PFS and can guide design of future PFS tools for other applications.

1.1 Less Invasive Surgery and Computer Assisted Surgery in Joint Replacement

Joint replacement surgery involves replacing diseased articulating surfaces of a joint with metal or plastic components. Each bone must be cut accurately to accept the appropriate implant. The bone must fit the implant accurately or the implant may loosen. The bone must also position the implant accurately for proper biomechanics. Conventionally, saws and drills are used to shape the bone. A carefully designed set of metal guides is used to align these tools to achieve accurate cuts.

Less Invasive Surgery (LIS)

Much effort has recently been devoted to adapting these guides to allow less invasive surgery (LIS). Researchers attempt to reduce invasiveness not only by shrinking incisions but also by altering the approach to limit damage to key tissues, such as the quadriceps muscles in knee surgery. Advocates claim that the decreased surgical trauma leads to faster recovery, less operative blood loss, reduced hospital stays, and lower short-term pain scores. [Price 2001] [Muller 2004] [Bonutti 2003] [Tria 2003]

However, adapting conventional instrumentation to LIS presents several challenges. With a smaller incision, it may be harder for the surgeon to see the tools in the surgical site and harder to locate anatomic landmarks for placing jigs. Smaller jigs may also be less accurate. These factors increase the technical difficulty of the procedure and make it harder to achieve accurate results. The resulting procedures are only appropriate for highly skilled surgeons.

Computer-assisted Orthopedic Surgery (CAOS)

Another field that has been developing is computer-assisted orthopedic surgery (CAOS). These systems use robotic techniques to achieve accurate bone cuts for joint replacement surgery. In some techniques, a robot arm is used to prepare the bone for joint replacement surgery. In another technique, called navigation, the positions of standard surgical tools are tracked, and a computer screen guides the surgeon to the right position.

CAOS systems are designed to ensure accurate cutting. Meanwhile, LIS can improve patient outcomes, but the techniques make it more challenging to achieve accurate results. This makes CAOS an ideal tool for enabling LIS without sacrificing accuracy. Some work has been done to use existing CAOS systems toward this goal.

Semi-Active Operation in CAOS Systems

Some of the robot-arm approaches use a "semi-active" mode of operation. This means that instead of moving autonomously, the robot is moved by the surgeon. The surgeon holds the robot tooltip and moves it directly, while the robot restricts the surgeon's motion to ensure accurate cutting.

One advantage of semiactive operation is synergy: the combination of the strengths of both human and robot. The robot ensures accurate cutting, while the human can identify and maneuver around soft tissues better than any robot.

The PFS: A New Tool for Less Invasive Joint Replacement

The PFS is a new type of tool to enable less invasive joint replacement. Unlike most prior art, it has been designed from the ground up for LIS. The PFS features a long slender nose to reach deep into small incisions. The PFS also achives excellent synergy with the surgeon because of its freehand mode of interaction. The primary significance of the PFS is the combination of these two: The long slender nose provides the ideal form factor for LIS, and excellent synergy enhances surgeon dexterity to enable operation through smaller incisions.

The freehand mode of operation used by the PFS provides excellent synergy with the surgeon. Unlike any current CAOS arm, it allows full 6DOF and unrestricted range of motion. Freehand manipulation of the tool also offers more immediacy of interaction than manipulating a tool attached to the end of a robot arm. These qualities allow the surgeon to manipulate the tool as naturally and dexterously as possible. This is not just a "soft" benefit: increased dexterity allows the surgeon to safely and efficiently maneuver in smaller incisions.

Navigation also offers freehand operation. However, the capabilities of navigation are limited to certain simple types of cuts. Navigation would not be suitable for controlling the PFS.

1.2 Cutting Accurately

The biggest technical challenge in implementing the PFS was cutting accurately enough. To achieve accuracy, the PFS control software predicts the allowable blade extension 12-48ms in the future. Prediction is necessary for two reasons: First, the blade retraction speed is limited. Second, the Optotrak only reports the tool position every 12ms. Therefore, 12ms of overcutting could occur before the PFS even became aware of it.

To predict the allowable blade extension, the PFS software predicts the position of the PFS, and then calculates how far the blade may extend from that position without violating the target shape. Prediction of the PFS position is based on the assumption that the surgeon presses the PFS against the workpiece, so that the PFS maintains contact with the workpiece. Thus predicting the PFS position requires an accurate model of the current workpiece shape. The algorithm uses a heightfield model to keep track of the current workpiece shape as cutting progresses.

The cutting control algorithm includes several geometric routines which operate on the heightfield. For computational efficiency, these were hard-coded for the shape of the PFS cutter and guard. This is necessary to perform all the required calculations every 12ms Optotrak cycle. The necessary calculations are as follows:

- The heightfield is updated based on the current blade position. This is done using a published algorithm that finds the intersection of a ray with a capsule shape.
- Four future timesteps are predicted. For each, the predicted PFS position is adjusted to lie on the workpiece surface.
- At each future timestep, the allowable blade extension is calculated in five candidate extension directions.
- Finally, for each candidate extension direction, the amount of material removal is estimated, in order to find the direction which maximizes material removal.

The PFS guard also plays a very important role in cutting accuracy. The guard allows the PFS to regulate the depth of cut on a finer timescale than the Optotrak updates. Because the guard cannot penetrate the bone, it limits the depth of cut, relative to the workpiece surface. When the PFS sets the blade extension, the guard limits how far the blade can cut in the 12ms before the next Optotrak update. Without the guard, there would be no restriction on how deep the blade could cut.

1.3 Thesis Overview

This document can be divided into three sections.

Background and Task Description

The first section describes the background and task description for the PFS.

- Chapter 2 describes prior art in less invasive surgery (LIS) and computer-assisted orthopedic surgery (CAOS). Conventional LIS techniques have shown promise for improving patient outcomes, but increase the difficulty of accurate cutting.

CAOS techniques, which were developed to improve accuracy, are a natural fit for enabling LIS.

- Chapter 3 describes the PFS and its relation to prior art. The PFS enables LIS by combining CAOS accuracy with the ability of the surgeon to dexterously maneuver around soft tissues.
- Chapter 4 describes the unicondylar knee replacement (UKR) procedure that we will use to prove the PFS concept.

PFS Implementation

The next section of the thesis describes technical implementation of the PFS.

- Chapter 5 describes the mechanical handheld tool.
- Chapter 6 describes the heightfield data structure used to represent the current workpiece shape.
- Chapter 7 describes the algorithm that controls PFS blade extension to ensure accurate cutting.
- Chapter 8 describes some theoretical tools which can be used to better understand and optimize PFS cutting error. In particular, we examine how error in predicting the PFS position affects cutting error.

PFS Evaluation

The final section of the thesis describes experimental results of cutting accuracy.

- Chapter 9 describes experiments to test PFS cutting accuracy. Foam blocks and foam Sawbones were used. We compare "open" cutting with cutting through a small incision in flexible foam "skin". In addition to accuracy, useful parameters such as user speed and acceleration were measured.
- In Chapter 10, we analyze the experimental results from Chapter 9 to determine the largest sources of cutting error.
- In Chapter 11 we analyze the experimental results from Chapter 9 to demonstrate that the prediction algorithm achieved more accurate results than a purely reactive algorithm would. In particular, we introduce a measure of cutting efficiency, and demonstrate that the prediction algorithm improved accuracy without sacrificing efficiency.

Chapter 2. Towards Robotic Tools for Less Invasive Joint Replacement Surgery

Joint replacement surgery involves replacing diseased articulating surfaces of a joint with metal or plastic components. For proper implant attachment and biomechanics, the bones must be accurately cut to receive the implant. Conventionally, cutting accuracy relies on a series of metal jigs with slots that guide saws and holes that guide drills.

Surgeons have recently begun developing less invasive techniques for joint replacement surgery. By operating through smaller incisions, these techniques cause less damage to the muscles and other structures necessary for joint stability and motion. However, these less invasive techniques are more challenging, and it is harder to achieve the required accuracies.

Another recent development in joint replacement surgery is the use of robotic technologies to achieve accurate bone cutting. Excellent accuracy has been achieved with robotic arms, but surgeon enthusiasm has been limited due to factors such as safety and user-friendliness. Another approach is "navigation", which uses position tracking technology instead of a robot arm. A position-tracking instrument tracks the position of conventional surgical tools and a computer provides on-screen guidance to the surgeon to aid in accurate positioning of the tools. Navigation has seen considerably more acceptance by surgeons.

One exciting possibility is how these robotic technologies can enable less invasive techniques. The challenge with less invasive techniques is achieving the necessary accuracy through a limited incision, and robotic technologies can ensure that accuracy. Navigation has been applied to less invasive techniques with success. However, the capabilities of navigation are limited: it can only guide the cutting of simple shapes. Robot arm solutions are capable of cutting much more complex shapes, but have seen much less application to less invasive techniques. In this thesis we describe a tool that can achieve complex shapes like robotic techniques, but with a mode of interaction similar to navigation. Unlike most of the systems described in this chapter, the tool we developed was designed from the ground up for less invasive surgery.

2.1 Joint Replacement Surgery: Task Description

Joint replacement surgery involves replacing the diseased articulating surfaces of a joint with metal or plastic implants. The two most common joint replacement procedures are total knee replacement (TKR) and total hip replacement (THR). In TKR, the knee surfaces of the femur and tibia are replaced (Figure 2-1). Optionally, the wear surface of the patella may be replaced as well. Although the knee is dominantly a hinge joint, the implants are two separate parts that slide over each other rather than being explicitly joined as a hinge. In THR, a ball-headed implant is used on the femur and a cup-shaped implant in the pelvis. In this thesis, we will focus on the application of knee replacement.

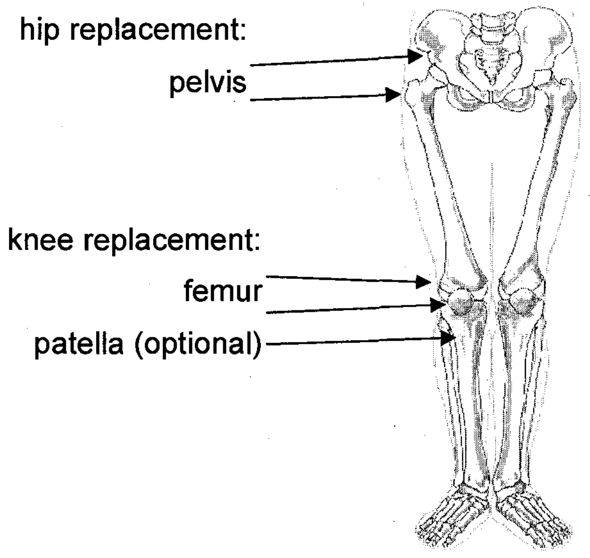


Figure 2-1: In knee replacement, implants are installed on the femur, tibia, and optionally patella. In hip replacement, implants are installed on the femur and pelvis. (From [Villarreal 2007])

The primary task of joint replacement surgery is to cut the affected bones to accept the implants. The bone is cut to a shape that mates closely with the implant. The implant may be attached with cement, or it may be press-fit, achieving long-term fixation by bone ingrowth into porous imlant surfaces. For TKR, the prepared shape of the bone is typically composed of flat facets, to facilitate the conventional use of saws for cutting the bone (Figure 2-2). For additional stability, there may also be holes drilled in the bone which mate with posts on the implant.

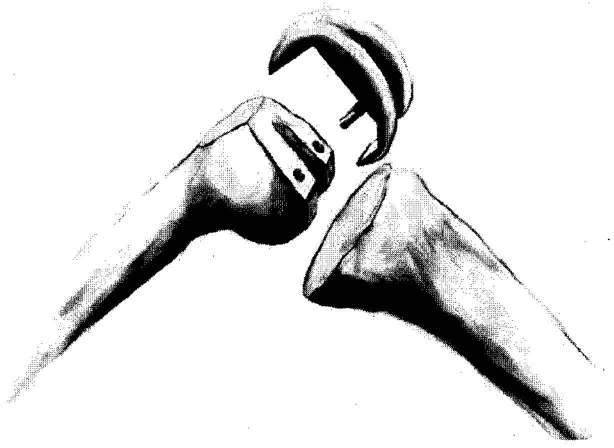


Figure 2-2: Total knee replacement implant. The femur is cut to a faceted shape that mates with the implant. A second implant is installed on the tibia.

Accuracy of cutting is critical to success of the implant. In particular, *fit* and *position* of the implant are important. The implant must *fit* closely with the bone or it may loosen. It must also be *positioned* properly on the bone for correct biomechanics, or the joint may suffer dislocation, restricted range of motion, or premature wear.

Conventionally, powered saws and drills are used to prepare the bone for TKR. To ensure accuracy, a carefully designed series of guides is used which feature slots to guide saws, and holes to guide drill bits. To position the implant properly, the surgeon aligns the cutting guides with "anatomic landmarks", which are features on the bone that can be accurately located. For instance, in Figure 2-3, the surgeon must visually align the cutting guide with the posterior femoral condyles to set the rotation of the implant about the axis of the femur. Holes drilled through slots in the guide establish the proper 3° rotation of the implant about the mechanical axis. A separate guide will be aligned with these holes to guide a saw to cut at the proper angle. The saw will be inserted through a narrow slot in that guide to constrain it to the desired cutting plane. This example illustrates a typical problem with conventional guides: since one guide builds on the work of another, cutting errors can add up.

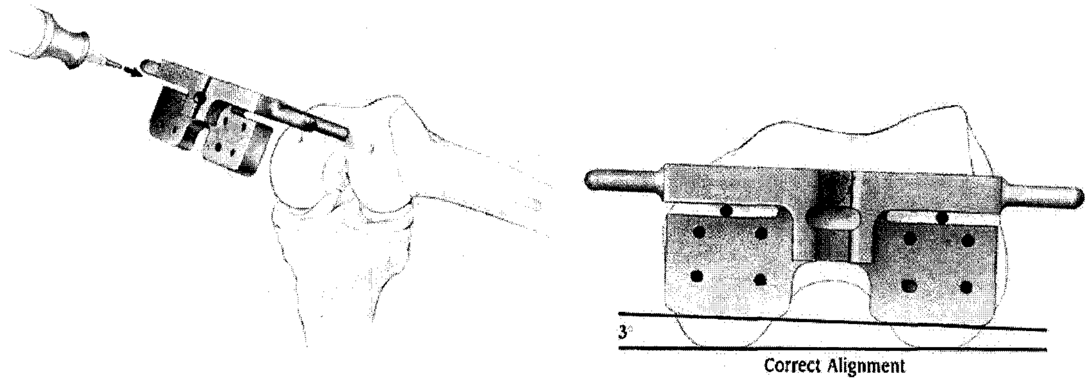


Figure 2-3: Example cutting guide for TKR. The surgeon positions cutting guides by aligning them with anatomic landmarks. (From [Zimmer 1997])

2.2 Less Invasive Joint Replacement Using Conventional Orthopedic Tools

The field of joint replacement surgery has seen much recent interest in less invasive surgery (LIS) and the ambitiously named minimally invasive surgery (MIS) [Price 2001] [Muller 2004] [Bonutti 2003] [Tria 2003]. Advocates claim that the decreased surgical trauma leads to faster recovery, less operative blood loss, reduced hospital stays, and lower short-term pain scores.

The goal of LIS is not simply smaller incisions. [Tria 2003] states, "The length of the skin incision does not define minimally invasive surgery ... The management of the quadriceps tendon and surrounding muscles is the defining feature." Thus, smaller incisions are the result of the desire to preserve specific anatomic structures. This desire may also constrain the incision to a less convenient place than the surgeon would otherwise choose, resulting in a more challenging approach.

The surrounding muscle which has received the most attention in LIS knee surgery is the vastus medialis, which is the most medially located of the quadriceps muscles (Figure 2-4). Conventional knee replacement uses an incision that extends along the medial edge of the patella and up through the vastus medialis. As examples of LIS approaches, [Tria 2003] describes an incision that follows the contour of the patella and "does not violate the quadriceps tendon or the vastus medialis muscle; it is purely a capsular incision".

[Bonutti 2003] describes a slightly larger incision that includes a 2cm snip in the vastus medialis. In conventional TKR, the patella is everted (flipped over) to provide excellent exposure of the joint. Neither of these LIS incisions allows for patellar eversion, which makes surgical access to the joint more difficult.

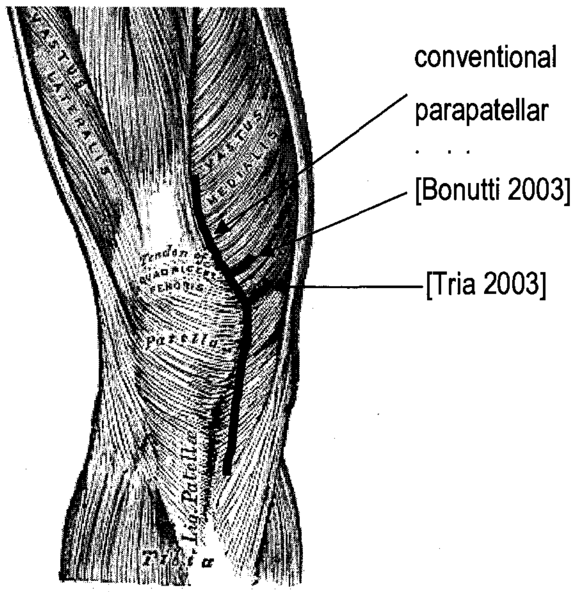


Figure 2-4: Treatment of the vastus medialis in knee surgery. LIS approaches attempt to minimize damage to the vastus medialis, which is significantly damaged by conventional approach.

To enable TKR through LIS incisions, the conventional cutting guides used must be modified. Attempts to modify conventional guides for LIS face several challenges: First, the guides must be small enough to fit in the incision. Second, the incision must accommodate the angle at which tools are inserted into the guide. Third, whatever anatomic landmarks are used to align the guides must be accessible through the incision.

These obstacles may be partially overcome. In [Tria 2003] the femoral guides are smaller and designed to be inserted from the side. All saw cuts are also performed from the side, instead of the broad range of front approach angles used in traditional knee instrumentation. These modifications make the small incision, quadriceps-sparing technique possible. In preliminary, retrospective results, the LIS technique resulted in lower initial pain scores, less blood loss, and decreased hospital stay. Postoperative x-rays showed good alignment, and LIS patients recovered flexion faster.

However, such techniques risk decreased accuracy. Smaller guides will fit in smaller incisions, but with a smaller baseline they are potentially less accurate. Also, if a smaller incision makes anatomic landmarks less accessible, it will be harder for the surgeon to correctly align the guides that reference them. Finally, any smaller or less ideal incision makes the surgery more difficult because visualization is reduced – there is less visual context to guide the surgeon through the anatomy.

Thus, LIS modifications of conventional approaches make it more challenging to achieve the required accuracy. Although the surgeons who pioneered these approaches have demonstrated success with them, the approaches may not be appropriate for most surgeons.

Note: Some critics fear that the risks of LIS outweigh the benefits, and caution that marketing hype and personal aggrandizement are driving premature adoption of the techniques. [Lilikakis 2004] [Ranawat 2003]. They point to the lack of well-designed, long term studies. Most results reported so far have been retrospective, and have lacked a control group. Long-term results are not available. Also, challenging LIS techniques may not be suitable for all surgeons.

One must always remember that the insistence on high-quality studies is not merely formality. Improvements in retrospective results may be due to improvements in the surgeon's technique, changes in anesthesia or rehab protocols, or other factors. For instance, although LIS hip replacement has been widely publicized, [Ogonda 2005] found no improvement in short-term results for LIS hip replacement in a prospective, randomized, controlled trial with 219 patients.

Arguing for or against LIS is of course beyond the scope of this thesis. Here we assume that LIS is a desirable approach, and propose a better tool for the task.

2.3 Computer Assisted Orthopedic Surgery (CAOS)

The past ten years have seen rapid growth in the field of Computer Assisted Orthopedic Surgery (CAOS). One significant class of CAOS systems are robot arms equipped with a rotating cutter to remove bone. A different approach is "navigation", in which position tracking equipment monitors the position of conventional surgical instruments and offers on-screen guidance to help the surgeon position the tools accurate. Initially, most CAOS systems were designed simply for enhancing accuracy, but recently [Levinson 2000] [Cobb 2004] the application of LIS has also been considered. With their ability to perform accurate cuts, it is reasonable that these systems may soon make inroads in enabling LIS.

2.3.1 Active Robot Arms

One of the earliest types of CAOS systems is active robot arms. Robodoc [Taylor 1994] was the first robot arm with a cutter designed for joint replacement surgery. It was designed to increase accuracy and decrease the risk of femoral shattering in hip replacement surgery. It is an industrial robot arm with a rotary ball-end cutter (Figure 2-6) which cuts a channel in the femoral shaft to accept the implant. This hole is non-round in cross section. In conventional hip replacement surgery, the hole is started with a drill and then squared with a broach. Not only does the broach make a very inaccurate cut, but if hammered too hard it can crack or shatter the femur. Robodoc is able to make a much smoother, tighter fitting cut in the femur and mitigate the risk of shattering the femur. The creators also claim that the more accurate cut will yield better implant fixation. In surgery, large screws hold the femur in place while Robodoc automatically cuts the planned shape.

Robodoc succeeded in reducing femoral fractures. [Bargar 1998] reported 0 femoral fractures in 65 patients for Robodoc, compared with 3 femoral fractures in 62 patients for the control group. Additionally, radiographic evaluation showed better implant positioning for the Robodoc group. Otherwise, outcomes were not statistically different at 1 and 2 year followup. Robodoc has since been applied to knee replacement [Wiesel

2001] and revision hip replacement [Taylor 1999], which is more difficult than primary hip replacement because cement from the original hip implant must be removed.

While Robodoc solved some of the problems of THR, it raised new concerns. The fact that it operated autonomously, independent of surgeon control, raised concerns about safety and surgeon acceptance. The possibility of a "run-away" failure worried some. (We should note that Robodoc uses multiple independent watchdog systems, and has never experienced such a "run-away" failure.) Finally, the large footprint of the robot was unwelcome in the crowded operating room.

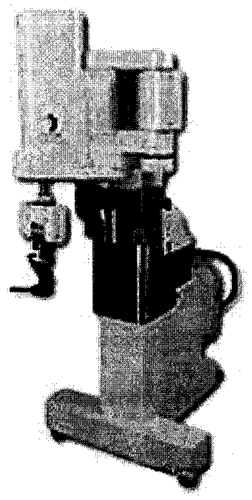


Figure 2-6: Robodoc

Figure 2-6:: Acrobot

A great variety of robot arm designs were proposed to address these issues with Robodoc. Many of these have been grouped under the name "semi-active robots". In truth, this group includes two separate classes of robot which address two separate issues: the autonomy issue, and the run-away possibility. We refer to these classes as "autonomy passive" and "mechanically passive." Autonomy passive means that the robot does not perform the cut on its own, but rather in close coordination with the surgeon. Mechanically passive means that the robot physically lacks the actuators to move under its own power. Thus it is incapable of making sudden and damaging motions.

In addition to autonomy and mechanically passive robots, new designs have been proposed with much smaller footprints than Robodoc.

2.3.2 Autonomy-Passive Robots

Acrobot [Harris 1999] (Figure 2-6) is the leading example of an autonomy passive robot. Like Robodoc, Acrobot is a robot arm with a rotating cutter. Acrobot actually consists of a large gross-positioning arm, which is locked in place during surgery, and a smaller 3DOF motorized stage on the end of the gross-positioning stage. The surgeon pushes a force-sensitive handle at the tip of the robot to move the cutter around. Acrobot allows free motion within an area, but its motors resist the surgeon's motion and create a "virtual wall" through which the cutter cannot pass. By moving the cutter over the entire virtual wall, the surgeon cuts the shape of the virtual wall into the bone. The resulting bone shape is the proper shape for attaching the implant. Like Robodoc, Acrobot requires the bone to be held in place with screws.

Acrobot has been applied to preparing the bone for TKR. Postoperative CT for the first 7 clinical tests showed leg alignment to be within 2° of plan. Additionally, the fit of the implant was excellent. [Jakopec 2003]

2.3.3 Mechanically-Passive Robots

Mechanically passive robots are by nature autonomy passive as well. Several have been developed that operate on the same principle as Acrobot: The surgeon moves a rotating cutter on the tip of the robot arm, while the robot creates a virtual wall for the surgeon to follow. Instead of using motors, these systems limit the surgeon's motion in other ways. [St Erbse 1998] used brakes on the robot joints, PADyC [Troccaz 1998] used one-way clutches, and cobots [Moore 1999] used continuously variable transmissions. Each of these devices is incapable of generating motion in the robot arm: the surgeon is entirely responsible for the energy used to move the robot, and the robot only limits that motion.

Another type of mechanically passive approach was used by [Kienzle 1992]. A conventional robot arm was used, which held sawing and drilling guides. The arm moved the guide into proper position, and then locked in place while the surgeon inserted a saw into the guide to make the cut.

Even mechanically active robots usually employ mechanical limitations for inherent safety. The Robodoc arm was an off-the-shelf industrial arm, but was modified to use smaller motors [Kazanzides 1999]. Acrobot has a limited range of motion, and is designed with just enough strength to perform the task. "Consequently, the robot is relatively safe, because potential damage is limited in terms of force and is constrained to a small region" [Jakopec 2001].

2.3.4 Limited Footprint Robots

Another limitation of large robot arm approaches is their footprint in the operating room. Not only is the physical size of these robots large, but they require the bone to be held in place with screws. This takes up space and prevents the surgeon from manipulating the leg, as is sometimes done conventionally to access different parts of the anatomy and to test implant fit. Recent work has included small size robot manipulators, and bone-attached robot manipulators.

CRIGOS [Brandt 1999] is a small parallel manipulator designed for multiple orthopedic operations such as milling or holding drill guides. It is attached to the surgical table during use. CRIGOS still requires the bone to be held rigidly in place.

Several bone-attached robots have been developed. These are small and do not require the bone to be rigidly immobilized. MBARS [Wolf 2005] is a small bone-attached parallel manipulator that is initially targeting patellafemoral arthroplasty. [Chung 2003] describes a bone-attached robot for preparing the femur for hip replacement. It is attached to the femur with clamps. Both of these robots attach to the bone in a way that requires significant exposure. Praxiteles [Plaskos 2005] is a bone-attached robot that specifically considered LIS in its mounting method. It attaches to the femur for knee surgery with two closely-spaced screws so that the necessary exposure is minimized. Praxiteles uses a combination of actuated and unactuated degrees of freedom to guide the cut.

For all of these small robots, the limited range of motion can be seen as a mechanical limitation that brings some safety. [Brandt 1999] states, "the restricted workspace reduces the area of potential collision and brings on additional safety for both the patient and the medical staff."

One final interesting system is modiCAS [Pieck 2003], in which a robot arm is used without needing the bone to be held in place. Instead, the bone is outfitted with an infrared marker and its position is tracked with an optical tracking system. The robot arm then moves to follow the sensed motion of the bone.

2.3.5 Navigation in Joint Replacement Surgery

Surgical navigation is a technology that takes a different approach. Navigation involves no robot arm that moves cutting tools in the surgical field. Instead, the navigation system senses the position of conventional surgical tools and provides the surgeon on-screen positioning feedback such as a crosshair. The first navigation system for joint replacement was HipNav [Simon 1997]. In HipNav, an optical tracking camera senses the position of flashing infrared flags attached to the pelvis and cup insertion tool in hip replacement surgery. The surgeon achieves the proper angle for the cup by moving the cup insertion tool until two crosshairs line up on the computer screen. In laboratory experiments, HipNav was found to achieve cup orientations better than 1°.

Navigation has been applied to knees surgery as well. For knee surgery, navigation is used to position a cutting guide, which then guides a saw as with conventional surgery.

Navigation addresses many of the issues with Robodoc and other robotic approaches. The surgeon is directly in control of the procedure. The system is autonomy and mechanically passive. The system footprint is minimal. Safety is maximized, because if the system fails the surgeon can transition seamlessly to conventional technique.

A further advantage of navigation is that it uses tools the surgeon is familiar with, so surgeon acceptance is high. Navigation has been rapidly gaining popularity, and many commercial navigation systems are now available [Stulberg 2002] [Sparman 2003] [Chin 2005].

2.3.6 Advantages of Semiactive CAOS Systems: Safety and Synergy

Safety

Most recent CAOS systems use some sort of "semiactive" mode of operation, i.e. autonomy-passive or mechanically-passive. These paradigms were set up in response to perceived problems with active robot arms, especially safety issues.

One of the first safety issues that springs to mind with CAOS systems is run-away failure. Mechanically passive systems are designed to reduce the danger of run-away failure. Mechanical passivity offers the strongest guarantee against run-away failure because the system physically cannot more on its own.

However, with properly designed independent watchdog systems, the danger of run-away failure is minimal. Robodoc offers a good example of how adequate safety systems can prevent run-away failure. A software watchdog timer ensures that the software is functioning properly. Independent joint encoders monitor the robot's position to ensure that it does not leave a predefined boundary corresponding to the planned resection. Finally, a force sensor is monitored to ensure that the robot does not exert unexpectedly large forces on its environment. These precautions have allowed Robodoc never to experience a run-away failure.

The potential damage from run-away failure is high. However, with properly designed safeguards, the risk of run-away failure is remarkably low.

Another type of safety hazard which is much more likely than run-away failure is cutting in the wrong place. This can occur if a tracking marker or the bone fixation is bumped and moves, or if there is an error locating the planned cut on the bone. Unlike run-away failure, the challenge with this type of error is to detect it. If detected early, it can be corrected. If it is not detected, the damage can be severe: if too much bone is removed, it may be impossible to install the implant correctly.

One benefit of autonomy passivity is that it can help in detecting the error of cutting in the wrong place. Since the surgeon is more directly involved in performing the cut, it may be easier to recognize that the cut is in the wrong place before too much harm is done.

Synergy

Another important benefit of autonomy-passive approaches is synergy: combination of the strengths of robot and surgeon. The primary strength of the robot is accurate positioning. The strengths of the surgeon are looking at and understanding the anatomy, manipulating soft tissues more nimbly and intelligently than any robot, picking up on cues such as noise, vibration, and forces both from cutting and pressing the side of the tool against soft tissue; and reasoning about novel situations unlike any robot. Additionally, autonomy-passive designs offer very intuitive usability, because the surgeon directly manipulates the surgical tool, which often resembles a "smart" conventional tool.

Autonomy-passive approaches are sometimes justified as a play to the surgeon's ego, saying that "the surgeon wants to be in control." However, the synergy between surgeon and robot is a very compelling advantage.

2.4 Application of Robots and Navigation to LIS

Development in less invasive surgery has delivered promising results, but LIS techniques make it harder to achieve the accuracy required for good patient outcomes. Since CAOS techniques have been designed to improve cutting accuracy, it seems natural to team these technologies with LIS, enabling less invasive approaches without compromising accuracy.

In fact, some work has been done on using navigation and robots to enable LIS. In [Levinson 2000], HipNav was used to enable less invasive hip replacement. Average incision size was reduced from 19.6cm to 12.1cm. Limp and stairclimbing were significantly better at 3 and 6 months for LIS HipNav patients. No differences were seen at 1 year. In addition to accuracy, another feature of HipNav that enabled LIS is on-screen visualization. The smaller incision limits direct visualization, making it harder for the surgeon to maneuver tools inside the incision. The HipNav display can supplement direct visualization of the surgical site.

[Cobb 2004] reports preliminary results on using Acrobot for unicompartmental knee replacement (UKR). UKR is a replacement of the left or right half of the knee only. The smaller implants allow for a smaller incision than TKR. The paper is titled "Robot assisted minimally invasive unicompartmental knee arthroplasty," but the approach used is not less invasive than the conventional UKR technique. Here, the conventional approach is already LIS, but is difficult to perform accurately, and Acrobot enables more accurate results. Postoperative CT scans showed the implants were positioned within 2mm and 2° of the planned position.

The synergy between surgeon and robot that is achieved by autonomy-passive systems is especially valuable for LIS. With a small incision, it may be necessary to manipulate soft tissues one way and then the other to access the full surgical site. Having the surgeon in direct control of the tool motion makes it possible to work in these more confined spaces without damaging soft tissues. The surgeon can see the soft tissues and maneuver around them more deftly than any robot could.

For instance, [Honl 2003] and [Bach 2002] state that Robodoc requires a larger incision than conventional approaches because of the autonomous nature of the robot. However, [Bach 2002] found no difference in functional gait analysis between Robodoc and conventional patients.

Chapter 3. The PFS: A New Tool for Less Invasive Joint Replacement

3.1 PFS Overview

The goal in this thesis is to create a tool that allows bone to be prepared for joint replacement with as little invasiveness as possible. A good model for this is arthroscopy, in which a long slender tool is inserted through a single-point incision. Joint replacement is slightly different: a single-point incision cannot be used, since the incision must allow for insertion of the implant. Our goal is for the tool not to be the limiting factor in incision size.

Rather than using a robotic arm, the goal was to develop a tool similar to navigation. The tool should be freehand, and should give the surgeon significant control over the operation. This mode of interaction combines the strengths of surgeon and robot: the surgeon can deftly maneuver among soft tissues, while the robot ensures accurate cutting.

The tool developed in this thesis is the Precision Freehand Sculptor (PFS). It is a handheld tool with a small rotary blade at the tip that allows the surgeon to shape the bone to accept the implant. (Figure 3-1) The blade can extend and retract behind a guard under computer control. As the surgeon moves the tooltip freehand over the surface of the bone, the computer extends and retracts the blade so that only the appropriate material is removed.

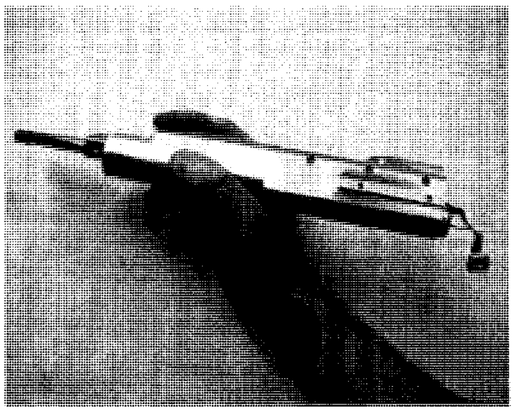


Figure 3-1. PFS handheld tool

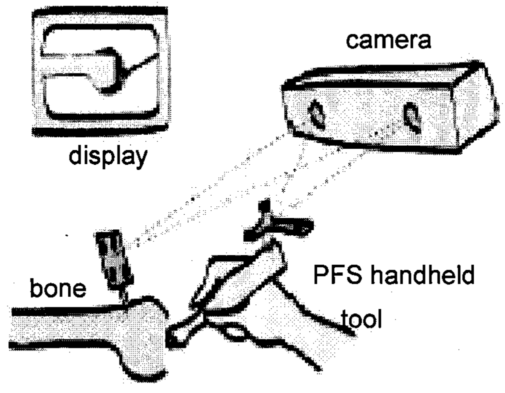


Figure 3-2. Complete PFS System includes handheld tool, tracking camera, and computer display.

An optical tracking camera continuously monitors the 6D position of infrared markers attached to the tool and bone (Figure 3-2). Based on the tracked positions, the PFS computer calculates the blade's position with respect to the *target shape* which it has been instructed to cut in the bone. The target shape partitions the bone into *waste material*, which should be removed, and *good material*, which should not be removed. It is the objective of the computer to control blade extension and retraction so that the surgeon removes all of the waste material and none of the good material.

The PFS system also includes a computer display. It shows the tool moving across the bone in 3D and cross-section views, which helps the surgeon manipulate the tool if the tooltip is obscured within the incision. The display is updated as bone is removed so that the surgeon can see what material must still be removed.

The complete surgical usage scenario for the PFS is as follows: Prior to surgery, the surgeon specifies where the implant should be placed on the bone, which determines the target shape. In surgery, the surgeon begins cutting away the bone material in smooth, even passes. The PFS blade retracts and extends to ensure that only waste material is removed. As cutting nears completion, the blade extends and retracts frequently and quickly to remove the last remaining bits of waste bone. The blade extends only a fraction of its maximum travel to cut away these final shavings of material. The guard that surrounds the blade rests on the bone surface and gives the tool good control over how far the blade penetrates. When the graphical display shows that all waste material has been removed, the remaining bone has been shaped to fit the implant. The surgeon installs the implant and completes the surgery.

3.2 PFS System Components

The complete PFS system includes the handheld tool, computer display, and tracking system. These components are connected to a PC which runs the PFS control software. The major components of the software are a heightfield model which keeps track of the target shape and the current shape of the bone, and the blade control algorithm which calculates how far to extend and retract the blade for accurate and efficient cutting. Below we describe each of these hardware and software components.

3.2.1 Handheld tool

To enable less invasive surgery, the PFS tool we designed features a long, slender nose to allow it to operate through small incisions. The guard that surrounds the blade makes it easier to insert the blade into cramped areas without accidentally cutting the wrong tissues. The size of the blade was chosen to be as small as possible while still providing adequate material removal rate. Likewise, the entire tool was designed to be as small and lightweight as possible.

The primary job of the handheld tool is to enable and disable cutting as commanded by the PFS computer. The guard plays an important role in this function. As the surgeon presses the tool against the bone, the blade cuts into the bone surface while the guard rests on the surface. The guard allows the computer to limit how far the blade can cut. When the blade retracts, the guard continues to rest on the surface, so that the tool does not "give way" under the surgeon. This makes cutting transitions quick and seamless.

The blade can extend and retract axially or radially, and any direction in between, through a 90° range. This ensures that the surgeon is not restricted in what angle the tool must approach the work, so that the approach angle can be dictated by the requirements of the incision, not the tool.

In every direction, the blade can extend 2mm beyond the surrounding guard and retract 2mm behind it. It can cover this distance in 100ms. The blade can also extend to intermediate distances as required.

Axial extension of the blade is actuated by an ultrasonic motor fitted with an encoder. Radial extension is actuated by a DC gearmotor fitted with an encoder. Rotation of the blade is provided by an off-the shelf orthopedic drill handle which is inserted into the back of the tool, and rotates up to 70,000 RPM. The blade is an off-the-shelf orthopedic bur, capsule-shaped and 6mm in diameter.

3.2.2 Workpiece Model

As the tool cuts, the PFS software maintains a model of the bone shape. This model is used by the graphical display and by the software that decides how far to extend the blade. The model is updated using the Optotrak position data, by assuming that all material that intersects the perceived blade position has been removed.

The workpiece model is represented using a heightfield. The target shape serves as a base for the heightfield model, and that base is tiled with points at which the height of the actual surface above the target surface is recorded. The heightfield allows for easy updating as the bone is being cut, and provides floating-point resolution in the thickness of waste material. In this work the spacing of points on the heightfield was roughly 1mm.

3.2.3 Computer Display

The computer display (Figure 3-3) enables LIS by reducing the surgeon's dependence on direct visualization. To do this, the display must be very intuitive, so the surgeon can easily move the tool to the desired position or along the desired path simply by watching the screen. The display features 3D and cross-sectional views of the tool moving over the bone. The display uses the heightfield model so that the display is updated as material is removed. Many design iterations and practical usage experience were used to improve the display.

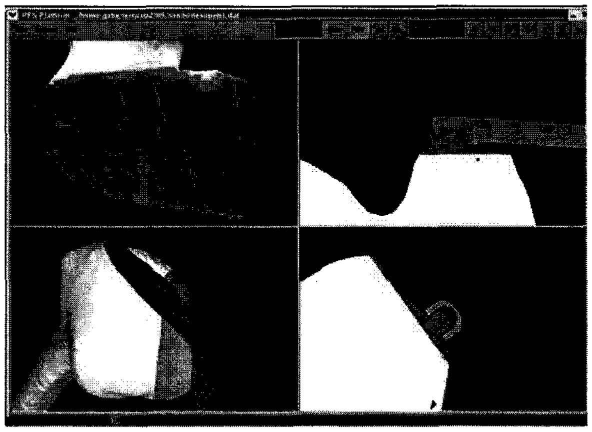


Figure 3-3. Computer display includes 3D (left) and cross-section (right) views.

Aside from improving visualization, the display must also highlight the waste bone material so that the surgeon can locate it. This is important because there is no material difference between waste bone and good bone, so the only way for the surgeon to ensure that all waste bone has been removed, or to locate the small areas that still need to be removed, is by looking at the display.

3.2.4 Blade Control Software

The blade control software extends and retracts the blade to make sure that only the appropriate material is removed. The calculations are based on the Optotrak estimate of the tool's position with respect to the workpiece and target surface. In addition to calculating the maximum blade extension for the current instant, the software predicts the maximum blade extensions for several future timesteps as well. Using prediction enables the software to begin retracting in time to compensate for limited blade retraction speed and limited Optotrak update rate.

To predict future constraints on blade extension, the software simulates the future path of the tool as the surgeon moves it across the bone. To predict the path, the software extrapolates position based on velocity and acceleration, but assumes that the guard of the tool remains in contact with the bone surface. The software uses the heightfield model of the bone surface to adjust the extrapolated position to maintain contact with the bone.

Once the present and future allowable blade extensions are calculated, the algorithm computes how far the blade may be commanded to extend based on the dynamic constraints of the blade extension motors.

In addition to determining how far the blade may extend, the algorithm must also decide in what direction to extend the blade. It chooses the direction to maximize waste material removal.

Computation time was a major challenge for the blade control software. All calculations must be performed every 12ms as incoming Optotrak frames arrive. To achieve this goal, the algorithm uses hard-coded functions that directly incorporate the geometry of the blade and the guard.

3.2.5 Position Tracking System

The tracking system is the only off-the-shelf component of the major system components listed here. In addition to the tracking camera, which tracks the location of markers, the PFS relies on registration, calibration, and preoperative planning technologies, which locate the tool, bone, and target shape with respect to the tracked markers. These technologies have been developed by other researchers, and are not the focus of this work.

The tracking marker on the bone is attached via one or more screws in the bone. Navigation systems have developed the hardware for attaching the marker to the bone, and the PFS borrows from them. Typically the marker is attached though a separate small incision, away from the primary surgical site.

Choice of Tracking System

Tracking is one of the largest sources of cutting error. To minimize cutting error, the tracking system should be chosen on the basis of tracking accuracy and update rate.

Tracking accuracy directly affects PFS cutting accuracy. Tracking update rate affects how quickly the PFS can react to user motion to avoid overcutting. A slower update rate also decreases the accuracy of the model that the PFS maintains of the workpiece shape as it is cut.

Commercially available tracking systems include optical, electromagnetic, and mechanical arm varieties, each with its own advantages and limitations. Optical systems require line-of-sight to the markers. Electromagnetic systems may be affected by metal or electric motors. Mechanical systems can only track one or two objects and are less ergonomic. To maximize the chances of success with the PFS project, we chose the Optotrak 3020, the most accurate and fastest of the optical systems. It also outperforms any electromagnetic systems available. The Optotrak update rate is 83Hz (12ms) when used with 6 markers as in the UKR procedure. The nominal accuracy is 0.1mm RMS per LED.

Calibration, Registration, and Preoperative Planning

The Optotrak reports the position of the markers that are attached to the tool and bone. For these data to be useful, we must also know the positions of the tool and bone with respect to their attached markers, and of the target shape with respect to the bone. Several important technologies have been developed in the field of computer-assisted orthopedic surgery to do so. Calibration techniques find the rigid transformation between the tool and its tracking marker. Registration techniques find the transformation between the bone and the tracking marker attached to it. Preoperative planning technologies help the surgeon determine a target shape and its relationship to the bone.

Calibration

Calibration techniques are used to find the relationship between the tool and its tracking marker. Theoretically, if the tool could be constructed accurately with the marker attached in a known location, the calibration would be known by construction. However, inaccuracies in the manufacture of tracking LEDs and their attachment to a tracking marker limit the accuracy with which the frame-of-reference of the tracking marker is known. Therefore, techniques are employed for calibration which can find the location of the tool coordinate system without assuming a known location of the marker frame-of-reference.

One of the most basic calibration techniques used is pivot calibration [Cleary 2007], which finds the location of the tip of a pointed probe. The probe point is placed in a divot and the probe is pivoted around while the location of the marker is tracked. The coordinate frame of the marker traces out a sphere. By finding the center of this sphere with respect to the marker coordinate frame, the location of the probe tip $_{probemarker}P^{probetip}$ is located. Pivot calibration can then be used to carry out more complex calibrations. For instance, a probe can be pivot calibrated and then used to locate the important parts of another tool.

To calibrate the PFS, a divot on the tip of the tool is pivot-calibrated with a probe, and then two additional divots on the tool are touched with the probe to find the orientation of the tool.

Registration

Registration is similar to calibration, but it finds the relationship between the bone and the tracking marker attached to it. This can be more difficult because the bone does not have precise points of interest like the divots machined into the PFS.

Point-based registration is one common registration technique [Simon 1995]. A pivot-calibrated probe is used to collect the positions of a large number of points on the bone surface. These are matched to a 3D model of the bone, which might for instance be generated from a patient CT scan. Fiducial-based registration [Taylor 1994] is an older technique, where a calibrated probe is touched to fiducials such as screws, which have been attached to the bone in an earlier surgery. Presently, landmark-based approaches [Leitner 1997] [Bathis 2003] are being studied where a probe is used to locate specific anatomic landmarks, such as Whiteside's line or the femoral epicondyles in knee replacement.

Pre-Operative Planning

In pre-operative planning, the surgeon specifies where on the bone the implant will be placed. Preoperative planning is a step that is used in conventional surgery as well as computer-assisted surgery.

For conventional surgery, preoperative planning is done on radiographs. The surgeon places transparent overlays depicting the implant profile onto a full-size radiograph in order to choose the implant size and desired position. The surgeon chooses the implant position to fulfill surgical goals, such as alignment of the hip, knee, and ankle joints. The surgeon then measures the implant position with respect to anatomic landmarks that will be accessible during surgery. For instance, in knee surgery, the angle between the axis of the implant and the axis of the femur is used.

For computer-assisted techniques that rely on a computer model of the bone, the surgeon must tell the computer where to place the implant with respect to that computer model. Typically this is done with some kind of graphical computer interface. Planning on the computer can provide the additional benefit of features such as 3D simulation of range-of-motion and identification of impingement sites [DiGioia 1995]. This feedback can allow the surgeon to revise the plan to fine-tune the results.

However, computer-assisted planning can require additional time, which some surgeons object to. To address this complaint, *image-free* techniques have been developed which combine registration and planning into one step, which takes place during surgery [Leitner 1997] [Bathis 2003]. In image-free techniques, the surgeon identifies anatomic landmarks on the bone, typically with a tracked point-probe. The landmark positions determine the desired implant position without requiring a computer model of the actual bone shape. In this way, registration and planning are combined into one step.

The PFS is independent of any particular methods of registration or preoperative planning. For the PFS, the purpose of registration and preoperative planning is to specify the location and shape of the target shape with respect to the tracking marker on the bone. Any of the methods described can perform this task. However, it should be noted that the PFS requires an initial model of the bone surface before any cutting begins, and image-free techniques do not provide such. Using the PFS with image-free techniques would require modifications to provide at least a rough estimate for the initial bone surface.

3.3 Advantages of PFS

The PFS was designed to enable LIS joint replacement. The two most important features for enabling LIS are the long slender nose and the freehand mode of interaction. The long slender nose offers the ideal form factor for LIS. Freehand use improves surgeon dexterity, allowing smaller incisions.

Freehand use combines the strengths of the surgeon and the robot. The primary strength of the robot is accurate cutting. The strengths of the surgeon are many, such as picking up on cues from cutting forces and sounds; and recognizing and dealing with novel situations. Surgeon strengths that are especially useful for LIS are knowledge of the anatomy, the ability to visually identify soft tissues, and the ability to dexterously maneuver around them. The surgeon may coordinate the motion of the PFS with retractors, or may manipulate the leg to access different parts of the surgical site. This ability of the surgeon to work in small incisions, and to do so in a speedy manner, makes the synergy between the PFS and the surgeon an invaluable aspect of the PFS strategy for LIS.

Freehand motion enables greater synergy than autonomy-passive approaches that use a robot arm. Freehand motion offers 6DOF and unrestricted range of motion, which has not been demonstrated in any existing autonomy-passive robot arm. Another advantage of freehand motion is that it offers more immediate interaction than moving a tool by moving an entire robot arm or by pushing a force-sensitive handle. These benefits should give the surgeon greater dexterity with the PFS, allowing operation through smaller incisions.

While navigation offers freehand motion, navigation is not suitable for controlling the PFS. Navigation works best for point-and-shoot type operations, where the surgeon aligns the tool once with no time constraint. With the PFS, the surgeon would need to constantly servo to the target surface, and any deviation would leave a permanent error. It would require extreme concentration from the surgeon, with high probability of error.

The key significance of the PFS is the combination of freehand operation with a long slender nose. I feel that these are the two most important factors for enabling less invasive surgery.

Additional Benefits

The PFS has several other benefits which can aid in LIS:

- The graphical display enables LIS. Small incisions offer less visual context to indicate what part of the bone is visible, and generally make it harder for the surgeon to see what's happening inside the incision. The graphical display can replace supplement direct visualization by showing the relative locations of tool and bone. This is a benefit of other CAOS systems for LIS as well.
- The PFS can address the bone from a broad range of angles, so that the incision is not limited by what approach angle the PFS needs.
- The PFS does not require the bone to be fixed in place. This is important for LIS, because the surgeon may use a "moving window" approach, in which the leg is moved to adjust what part of the anatomy is accessible through the incision.

The PFS also has several benefits that can aid surgeon acceptance:

- The PFS freehand mode of interaction is very similar to navigation. We hope that the qualities that have led to rapid surgeon acceptance of navigation apply to the PFS as well.
- Like navigation, the PFS has a small footprint in the sterile field.
- The PFS is nearly mechanically passive. Although the blade can move a small distance under computer control, the potential damage from run-away failure is much smaller than for a robot arm.

Accuracy: Comparison to CAOS Arm Systems

One question about the PFS suitability is cutting accuracy. Cutting accuracy is in fact the biggest technical challenge in PFS development. It is quite probable that the PFS will not achieve the same level of cutting accuracy that robot arms can achieve. However, accuracy is not a goal unto itself, but a means to better patient outcomes. Accuracy beyond a certain threshold no longer improves results. Therefore, the PFS does not necessarily need to cut as accurately as robot arms, but simply to the level of "good enough". Accuracy requirements are discussed in the next chapter.

3.4 Systems Similar to PFS

Other researchers have proposed systems similar to the PFS. None of these solutions involve retracting the blade behind a guard, but simply controlling rotation of the bur. [Kneissler 2003], [Heldreth 2003] and [Labadie 2005] suggest starting and stopping rotation of a handheld bur, or controlling its speed, to control what the user cuts and thus achieve a desired shape in bone. [Koulechov 2004] controls the speed of a drill bit to ensure the user drills along a desired axis for dental surgery. [Koulechov 2005] controls rotation of a handheld bur not for the purpose of cutting a desired shape, but simply to protect delicate vascular and nervous structures from damage in sinus surgery.

I believe that retraction behind a guard in the PFS will allow more accurate cutting than simply regulating blade speed. The guard allows the computer to control to a fine level exactly how much material the user can remove. This is especially important given the sensing latency of 12ms. The computer can set the tool for a very fine cut and be sure the user will not cut beyond that limit before the next position update.

Haider and Walker [Haider 2007] [Forman 2004] have taken a different approach to freehand cutting, using navigation to directly guide a standard sagittal saw operated freehand for TKR. They report 400% better implant positioning, but 200% rougher surfaces compared with conventional jigs. This is a promising approach, although it relies on the surgeon very carefully following the guidance. With the PFS, there is less potential for mistakes because the computer controls accuracy.

Chapter 4. A Target Procedure

To guide development and demonstrate the potential of the PFS for less invasive surgery (LIS), we chose unicondylar knee replacement (UKR) as the first target procedure. UKR involves replacing only the left or right half of the knee. UKR implants come in two varieties: non-inlaid types are installed with techniques and instrumentation very similar to that for total knee replacement. Inlaid implants are installed with a bur that is operated entirely freehand. Both types are attached with cement.

UKR is an ideal procedure for the PFS for two reasons: the amount of material removed is smaller than TKR so execution time can be more competitive versus saws, and the potential for LIS is greater because the implant can fit through a smaller incision than with TKR.

A PFS-specific UKR procedure was developed to best suit the capabilities of the PFS. This procedure was developed in parallel with the PFS handheld tool mechanism. The requirements of the procedure influenced the design of the mechanism, but the constraints of the mechanism also influenced the design of the procedure.

To be accepted by surgeons, it is critical that PFS operative time is competitive with conventional techniques. To this end, the mechanism and software should enable aggressive material removal. Usability and ergonomics also play an important role, helping the surgeon to operate the tool as efficiently as possible.

The PFS must also perform the required cuts with sufficient accuracy. Although the requirements for cutting accuracy are not completely understood experimentally, some literature gives reasonable estimates as to what accuracy is necessary and what accuracy conventional saw-based techniques achieve. Every component of the PFS contributes to cutting error, so every component must be designed with accuracy in mind.

4.1 PFS-specific Unicondylar Knee Replacement (UKR)

The PFS-specific UKR procedure was designed to minimize invasiveness and maximize cutting efficiency. Design of the procedure encompasses incision, order of cuts, location of tracking markers and camera, and other considerations.

4.1.1 Materials and Setup

The PFS-specific UKR procedure was developed on foam Sawbones™ wrapped in a sheet of flexible foam with an incision through it. (Figure 4-1) The implants were a cemented, non-inlaid type based on CAD models of a major manufacturer's UKR system. The actual implant models shown in Figure 4-2 were 3D printed with FDM. The femoral component features a large distal cut, and smaller chamfer and posterior cuts. The tibial implant requires the tibial plateau cut down to a flat surface which is bounded by a vertical wall near the tibial keel. Both implants featured two lugs which fit into holes drilled in the bone. The lugs were removed to focus on the shape cut by the PFS.

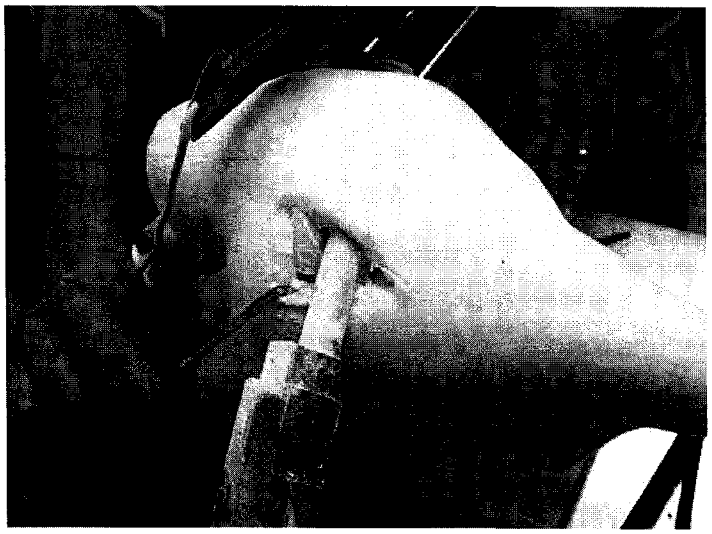


Figure 4-1. PFS operating on sawbones knee covered with foam sheet.

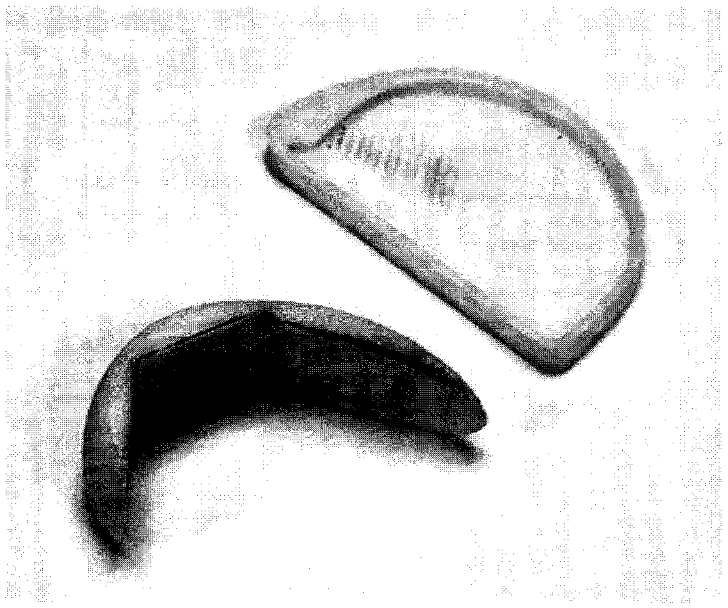


Figure 4-2. Femoral (bottom left) and tibial (top right) implants for UKR.

Preoperative planning for the procedure was ad-hoc. One Sawbones tibia and femur were CT scanned. The CT models were converted to surface models using HipNav software, and a modified version of the HipNav pre-operative planner was used to position the implants on the bone by eye. The CT scan and planning were only done once, with the assumption that subsequent Sawbones were sufficiently identical to the first.

Each tracking marker was attached to the bone using 2 bicortical screws at a site away from the primary surgical site. Point-based registration was used to register the position of the markers on the bones. The bone was registered to the 3D bone model derived from the CT image. Registration points were collected from all over the bone instead of limiting collection to areas that would actually be surgically accessible.

The CT-scanned surface model was also used for the graphical display.

4.1.2 Approach Angles and Order of Cuts

In addition to the need for accuracy and efficiency, two major constraints drove the design of the procedure: placement of the incision, and the necessity to create space between the femur and tibia for the PFS to operate. These two constraints influenced the order of cuts, and approach angles for each cut.

The incision location in knee replacement surgery is generally constrained to the area between the patella and the collateral ligaments because that is the least damaging to the important structures of the knee. The main freedom is in choosing how far the incision extends proximally and distally. The incision we used started just distal of the tibial plateau and was about 4cm long. Ideally this should be compatible with a quadriceps-splitting or quadriceps-sparing approach, but this can only be verified with cadaver testing. At minimum, it appears that the incision allows the operation to be performed without patellar eversion.

This incision limits the PFS to approach the bone directly from the front, or from the medial side to a limited extent. For maximum efficiency, the approach angle should be selected to allow the PFS to cut with the flat cylindrical part of the blade. The procedure we developed (Figure 4-3 a,b,c) allows the cylindrical part of the blade to be used for all cuts. The PFS approaches the distal femoral cut from the side, and the other cuts from the front. Retractors were used to expose the desired areas. Flexing and extending the knee can also expose different areas through the incision. For instance, extending the knee may be useful to expose the anterior part of the femoral distal cut.

The second major constraint on the procedure is the need to open up space between the femur and tibia. Initially, the femur and tibia are held tightly together by ligaments, making the region of contact difficult to access with the PFS. Although the PFS could cut into this area incrementally by plunge cutting with the tip, we wish to use the more efficient cylindrical portion of the blade. To open the space so that all cuts could be performed with the cylindrical part of the blade, the following series of cuts were used:

Distal femur: Figure 4-3a. The knee is flexed so that the distal femoral cut is accessible. The tool approaches from the medial side. If necessary, the knee can be extended slightly to expose the anterior portion of the distal cut.

Tibia: Figure Figure 4-3b. The knee is extended so that the cut distal femur apposes the tibia. Now the cut femur provides space for the PFS to access the tibia. The PFS approaches from the front. The surgeon must rotate the tool 90° around its axis to cut the short side-wall of the tibial cut. The knee should be partially flexed when operating on the posterior section of the tibial cut, to protect important structures at the rear of the knee.

Femoral chamfer and posterior cuts: Figure 4-3c. Finally, the leg is flexed again for the chamfer and posterior cuts. The cut tibia now provides space for the guard to fit in between the bones and access the posterior femur. Once again, the PFS approaches from the front.

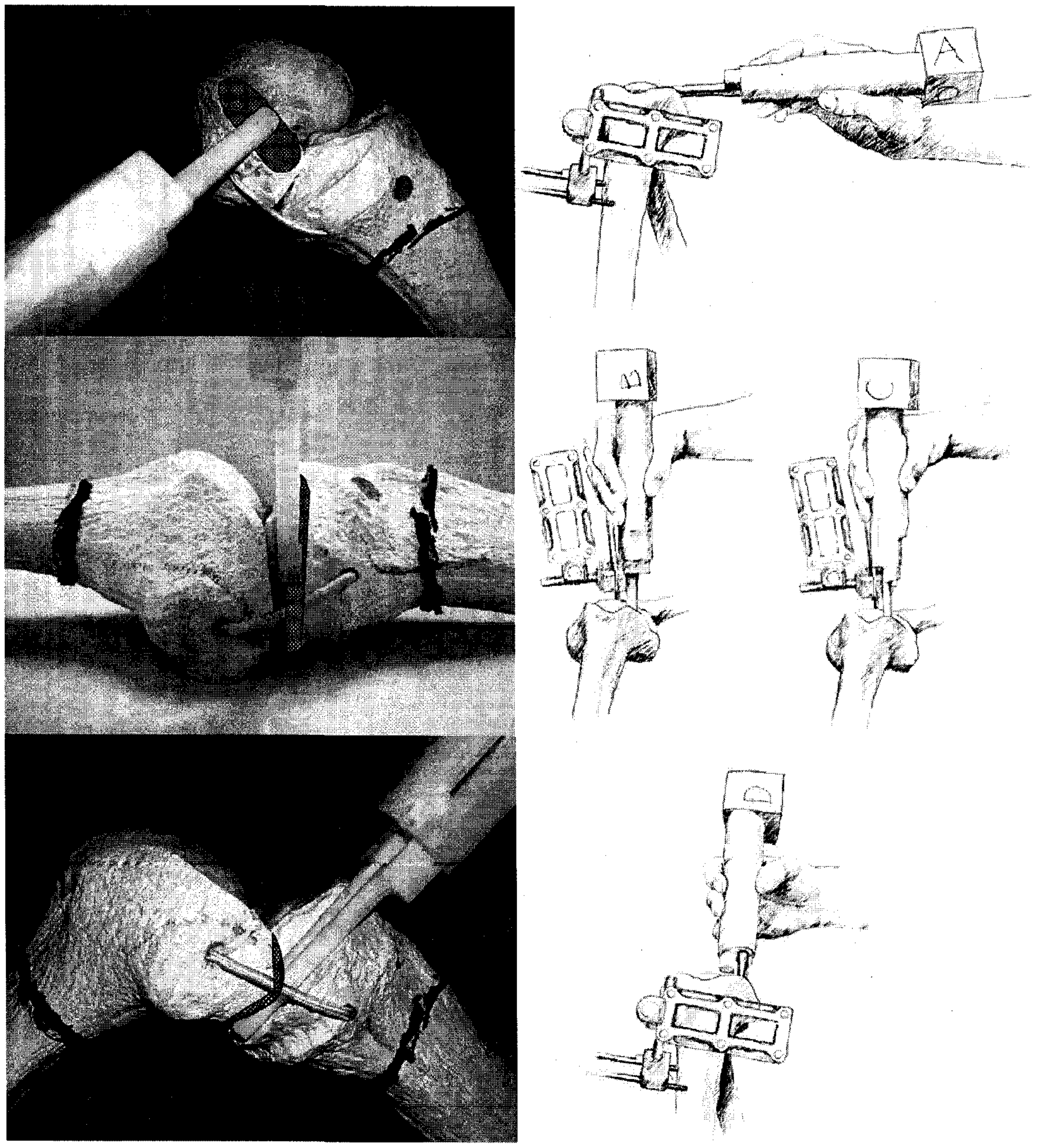


Figure 4-3. Left column: (a) Distal femoral cut. (b) Tibial cut. (c) Posterior femur and chamfer cuts.

Right column: camera's eye view of tracking. 4 Different tracking markers are needed on the tool, labeled A,B,C,D.

(d) Distal femoral cut. (e) Tibial plateau cut and tibial sidewall cut. (f) Posterior femur and chamfer cuts.
Tibia marker removed for clarity in (d) and (f), and femur marker removed in (e).

4.1.3 Tracking Arrangement

The placement of tracking markers and camera must be chosen to accommodate the range of motions that the tool and bones go through for the PFS-specific UKR procedure. The camera was placed at the head of the patient, about 7 feet off the floor. This is the least disruptive location, and the height allows good visibility over the surgical drapes and anesthesia equipment. The PFS tool required 4 tracking markers: one for the distal femur cut (Figure 4-3d), two for the tibial plateau and side-wall cuts (Figure 4-3e), and one for the chamfer and distal femur cuts. (Figure 4-3f) On the tibia, the PFS may be used at any angle between facing the tibial plateau and facing the sidewall. Instead of setting the two markers that correspond to these cuts at 90 degrees to each other, the markers are angled toward each other in order to minimize the maximum viewing angle the occurs in the transition. This minimizes optical tracking error.

Each bone marker was attached to the bone with two bicortical screws. The markers were attached towards the lateral side of the bone to avoid being obstructed by the tool cutting. Because the tibia goes through a range of motions, the angle of the tibial marker is set to halfway between the direction to the camera with the knee flexed and in extension.

4.1.4 Finishing the Cuts

After PFS cutting is complete, the bone still requires some final work before it can accept the implants. First, the PFS cannot create a sharp interior corner on the tibial cut, so it creates a rounded corner that will need to be squared in some way to accept the implant. This might be done with hand tools or with a power saw. The sharp internal corner can be seen as an artifact of the conventional use of saws. Future implants that are designed around the PFS might eliminate this requirement.

The other important finishing step is to drill holes for the lugs on the implant. I propose that these holes could easily be drilled using navigation for the drill, because all of the tracking infrastructure is in place. We did not implement the navigation because it was not the focus of this work.

4.1.5 Additional Concerns

Aside from the basic questions of how to make the cuts and where to place the tracking markers, there are several issues that are important to the success of the procedure. The Sawbones model was too basic to investigate these questions, but they bear future consideration.

Protecting Soft Tissues. Protecting soft tissues is important because if the PFS blade catches a structure such as a ligament it can be wrapped up around the blade almost instantly. To protect soft tissues such as the ACL from damage, a combination of hardware and software may be used. Retractors may be placed in front of the ACL to guard it. The software could also be programmed to retract when too near the ACL. With a more integrated future version, blade rotation could be stopped as well.

Waste removal. Due to the high speed of the bur, the cutting process results in a fine "dust" of bone which must be removed from the joint. In contrast, conventional saw-based techniques result mostly in large, easy-to-manage pieces of waste. There is, however, some precedent that the burring waste can be removed: Burring is used as the primary bone shaping method for some established UKR procedures.

The bone debris is contained within the joint by the joint capsule. Typically irrigation and suction are used to flush the capsule of debris. Note that he smaller incisions enabled by the PFS will make debris clearing more challenging. It may be that the technique necessary to flush the joint requires a larger incision than would otherwise be necessary, or requires supplementary portal incisions.

One possible solution for waste clearing is to include integral suction like arthroscopic tools do. However, arthroscopic tools make space for integral suction by using metal-on-metal instead of ball bearings. This may be acceptable for the 5,000 RPM typical of arthroscopic tools, but not for the 70,000 RPM the PFS currently uses. However, switching to 5,000 RPM might significantly lengthen the surgical time. Unfortunately, there is no obvious solution to integrate suction while maintaining high speed.

Cement Removal. When a cemented implant is installed on the bone, cement squeezes out from between the implant and the bone. This cement must be removed. This can be especially difficult towards the less accessible posterior edges of the implant, and smaller incisions will only increase the difficulty. Difficulty of cement removal is another factor that might limit how small the UKR incision can be made.

4.1.6 Surgical Scenario

To give a better sense of how all these pieces fit together, we will describe the overall surgical scenario. Note that this is one particular scenario. In particular, the PFS is compatible with other registration and preoperative planning technologies. The PFS is also applicable to surgeries other than UKR.

Prior to surgery, the patient is first CT scanned, and a 3D surface model is extracted from the CT scan. The surgeon plans the implant location on the computer using software similar to the HipNav Planner.

In surgery, a point probe is calibrated before any incision is made. The tool may be calibrated at this time also, or a previous calibration may be used. The surgeon attaches the tracking markers through secondary incisions, and then makes the primary incision. The point probe is used to collect points on the femur and tibia for point-based registration of the bone. The surgeon next uses the PFS to prepare both the femur and tibia. The leg is manipulated both to adjust what parts of the bone are accessible through the incision, and to open space between the femur and tibia. The surgeon watches the on-screen display to help guide the tool when the tool tip is hidden within the incision. As cutting completes, the surgeon uses the display to locate remaining areas of waste bone that must still be removed.

When the surgeon decides that all the waste bone has been removed, the cuts still need finishing work. First the rounded inside corner of the tibial cut is squared with hand tools or conventional power tools. Next, navigation is used to drill holes for the lugs on the implant.

The surgeon then applies trial implants to test the fit of the joint. If cuts need to be adjusted to adjust the implant fit, the surgeon can easily adjust the cut on the computer screen and then recut the bone with the PFS. (This capability was not implemented as part of this work.) Having the cut computer-controlled provides the flexibility to make easy updates like this, which would be more difficult with conventional techniques.

Finally, the surgeon applies cement to the implant, installs the implant, and removes excess cement squeeze-out. Then the incision is closed and the surgery is complete.

4.2 Efficiency Requirement

To gain surgeon acceptance, the execution time for a PFS procedure must be competitive with conventional instrumentation. For a typical implant facet, a saw cut can be performed in less than 10 seconds, whereas to bur away the material with the PFS may require several minutes.

However, the total procedure time for each approach incorporates more than just cutting time. Conventional procedures require extra time to align and attach the sawing guides. For each saw cut, a new jig must be aligned to anatomic landmarks and secured to the bone, and then removed after the cut is made. The PFS requires no such setup phase in between facet cuts, but does require a setup step at the start of surgery to attach the tracking markers and register the bones. This step is identical to the setup step used in navigation, which [Haaker 2005] reported to add about 10 minutes to a standard case.

Thus the PFS eliminates the time needed to setup sawing guides, but introduces other tasks that add operative time. If the PFS cutting time is fast enough, the net change in operative time can be small. One surgeon (Dr. Yram Groff) who used the Sawbones procedure thought the burring time compared favorably with the composite setup and sawing time of conventional instrumentation. Timing results will be presented in Chapter 9. Even if the PFS procedure does take slightly longer than conventional surgery, surgeons may find, as with navigation, that the benefits of the PFS are worth the extra time.

4.3 Accuracy Requirement

The accuracy requirements for the PFS can be broken down into *fit* accuracy and *position* accuracy. The implant must fit the bone closely, or it may loosen and need to be replaced. It must also be positioned correctly on the bone to ensure proper biomechanics.

The requirements for position accuracy are very well studied. Unfortuately, no studies adequately address the requirements question for fit accuracy. However, by examining several studies on required accuracy, along with studies on the accuracy of conventional instrumentation, we can generate a reasonable picture of the required accuaracy.

4.3.1 Requirements for Fit Accuracy

The accuracy required for implant fit depends primarily on whether or not the implant is designed to be attached with cement. Other implant-specific factors such as geometry and surface coating affect fit requirement as well. The majority of knee implants are presently cemented, but the trend is towards uncemented. The intended advantage of cementless implants is that they will hold firm longer and generate less debris because the interface between bone and implant continuously regenerates. However, cementless implants require more accuracy because they are press-fit onto the bone. Cemented implants are more forgiving because the cement can fill gaps.

The target shape for cemented implants is designed to produce a cement mantle of the desired thickness. In hip replacement, for which more literature is available, the femoral component cement thickness can range from 2-10mm. Cement thinner than 2mm risks cracking. [Cristofolini 2007] I am not aware of any papers on necessary cement thickness in knees, but a cement thickness of 2mm or less is more often used. Given a cement thickness of 2mm, a goal of ±1mm is probably a good guideline for cemented knee implants.

Another important requirement in preparing bone for cemented implants is to avoid sharp high points (cusps) on the finished surface. Cusps on the finished bone surface create sharp concave corners in the cement, which are stress risers and can lead to early cement failure.

Cementless implants have a porous mating surface for the bone to grow into. Typically the implants require a press fit to achieve fixation before the bone grows into the implant. Sometimes screws are used to supplement initial fixation. The implant must fit tightly to the bone because bone will only grow across a small gap. If the gap is too large it will fill with weaker fibrous tissue instead of bone. In the worst cases this can lead to implant loosening. However, complete ingrowth is not required. Even in successful cases, current techniques often achieve marginal ingrowth with cementless knee components. [Turner 1989]

To study bone growth across gaps, [Sandborn 1988] inserted a cylindrical cementless implant into the femoral shaft of dogs. The implant had separate regions which created a 0mm, 0.5mm, 1mm, and 2mm gap between the implant and the bone. All gaps filled in eventually and exhibited bone growth. However the smaller gaps filled faster. In the cancellous region, initial bone ingrowth was seen at 3weeks for 0mm, 0.25mm, and 0.5mm gaps and at 6 weeks for 1mm and 2mm gaps. [Dalton 1995] repeated this methodology to study the effect of hydroxyapatite-coated implants and found that hydroxyapatite improved fixation for gaps of 1mm or less over all timescales. [Søballe 1990] used a similar methodology and found that bone ingrowth for hydroxyapatite-coated implants was just as strong with a 1mm gap as it was for press-fit.

Although fit accuracy is requisite for good bone ingrowth, current findings indicate that micromotion of the implant also plays a significant role in ingrowth formation. [Kienapfel 1999] Micromotion refers to the microscopic motion of an ostensibly well-anchored implant due to the repetitive loading and unloading from tasks such as walking. Repetitive motion as small as 75μm between the bone and implant may impede development of a bony connection to the implant. Knee implants differ from those in [Sandborn 1988] because they are weightbearing and thus potentially subject to micromotion. Although bone grew across all gaps in [Sandborn 1988], the authors of that study suggest that gaps of 0.5mm or smaller may be important so that the bone can heal before the patient increases activity. Gaps that don't achieve ingrowth before the patient becomes fully active may never achieve ingrowth because of micromotion.

[Turner 1989] implanted 6 dogs with cementless TKR implants and examined bone ingrowth in the tibial components after 6 months. Bone ingrowth was most prevalent on and near the 3 pegs of the implant. The authors suggest this is because micromotion is smallest at and around the pegs. I believe an additional factor is that holes can be drilled accurately to form a press-fit for excellent bone apposition, whereas the accuracy of the tibial cut is significantly worse. However, these canine TKRs were successful despite lack of ingrowth. It may be that surface preparation accuracy is not critical for cementless implants because implant pegs or stems provide sufficient bony fixation when combined with fibrous ingrowth elsewhere.

Putting a number on required cutting accuracy for ingrowth remains elusive because it is a combination of gap size and micromotion, which depends on implant geometry. Existing studies fall into two categories: controlled gap size with "toy" implants [Sandborn 1988] [Dalton 1995] [Søballe 1990], and retrieval or radiological studies with real implants where the initial gap size is unknown [Turner 1989] [Berger 2001]. Neither is sufficient to completely address the question of accuracy requirements for bony ingrowth. In the face of these limitations, the acceptable gap size most frequently cited in the literature is 0.5mm from [Sandborn 1988].

4.3.2 Fit Accuracy and Conventional Instrumentation

Although the literature is sparse on what fit accuracy is required for knee implants, what is known is that conventional instrumentation performs adequately. An alternate way to specify a PFS accuracy requirement is to require cutting accuracy similar to that achieved by conventional instrumentation.

[Toksvig-Larsen 1994] studied the flatness of the tibial implant surface achieved with conventional instrumentation for TKR. Dental putty was used to make impressions of the prepared tibiae of 26 TKR patients. Positive plaster casts were made and scanned with a

CMM. The standard deviation of the surface was 0.26mm, with max-to-min value of 1.71mm.

[Otani 1993] identified blade toggle and guide motion as the two largest contributors to cutting error with conventional saw guides. Blade toggle refers to out-of-plane motion of the saw in the guide slot. The authors tested worst case error by cutting while toggling the blade all the way in one direction or the other. This is an angular error, so cutting error increases with distance from the guide. The worst case error from blade toggle was found to be 0.49mm at 5cm cutting depth.

To test the effect of vibration, translation of the guide during cutting was measured in two axes. Several standard methods of attaching the guide to the bone were tested. Vibration of the guard ranged from 0.2mm to 1mm peak to peak, depending on attachment method. Motion of the centerpoint of vibration ranged from 0.05mm to 0.6mm.

These two studies are complementary. [Toksvig-Larsen 1994] indicates that the accuracy for a single saw cut with respect to a flat plane is 0.26mm RMS. [Otani 1993] concerns the ability of the guides to position two saw cuts in proper relation, which is necessary for fit accuracy. Blade toggle and guard vibration were each found capable of introducing errors of 0.5mm or more.

4.3.3 Requirements for Positioning Accuracy

Implant *positioning* accuracy is important to ensure proper biomechanics. The requirements depend on the type of implant, but generally are not as stringent as requirements for fit accuracy. For knees, the primary goals of implant positioning are to achieve proper varus/valgus (bowlegged / knock-kneed) alignment and to equalize flexion and extension gaps for uniform tension in the knee.

Proper varus / valgus alignment of the knee ensures even distribution of weight between the medial and lateral condyles. The line of force extending from the center of the hip to the ankle should pass through the center of the knee to ensure even loading. [Rand 1988] reported 90% survivorship at 10 years for TKA implants with 0° - 4° varus alignment of the knee.

For UKR, the same guidelines apply. However, UKR and TKR differ in how varus / valgus alignment is controlled. In TKR, the angles of the implants control alignment. In UKR, the line between the replacement condyle and the unaffected one controls alignment. This means that in UKR, translation of the implant, rather than angular alignment, affects varus / valgus. To achieve an angular accuracy of ±2°, assuming 5cm between femoral condyles, the required translation accuracy is 1.7mm.

If the patient's leg in not initially in proper varus / valgus alignment, the surgeon will position the implant to correct the alignment. However, changing this angle affects the tension in the ligaments on either side of the knee. One side will become tighter and the other side looser. The surgeon can equalize the tension with techniques such as partially separating the tighter ligament from the bone, but only to a limit. The surgeon's judgment is required to decide how much the alignment can be corrected without damaging the ligaments.

The other major requirement in knee replacement is equalizing flexion and extension gaps. This creates even tension in the knee throughout its range of motion. When the knee is in extension, contact is between the distal femur and the tibia. In flexion, contact is between the posterior femur and the tibia. For the knee to operate properly, the distance between the femur and the tibia must be the same in both of these positions. Translation of the implant in the anterior-posterior direction changes the flexion gap and translation in the proximal-distal direction changes the extension gap. Small flexion / extension gap disparities can be corrected by soft tissue release, but larger disparities require bone cutting. I am unaware of any references that quantify these accuracy requirements. However, the recutting guides in knee instrumentation typically remove 2-5mm of bone [Zimmer 1997] so I assume disparities smaller than 2mm can be corrected with soft tissue release.

4.3.4 Summary of Accuracy Requirement

The accuracy goal for the PFS is to be accurate enough for use with cementless knee implants. Research on the fit accuracy requirement for cementless implants is not ideal, but the most agreed-upon result is that cementless knee implants can tolerate a gap of 0.5mm between the bone and the implant. Accuracy achieved by conventional saw-based instrumentation is 0.26mm RMS for a single flat facet. For the multifaceted shape used for the femoral implant, cutting error may exceed 1mm. The accuracy required for implant positioning is on the order of 1mm-2mm.

Because of the way the PFS cuts, fit accuracy is the challenging requirement. The PFS cuts a small amount of material at a time, so cutting error between neighboring points is relatively uncorrelated. This means that the accuracy of any one point with respect to the bone (position accuracy) is close to the point's accuracy with respect to its neighbors (fit accuracy). The PFS creates surfaces that are locally rough, but positioned properly overall. Since the requirement for fit accuracy is more demanding than the requirement for position accuracy, it is the primary requirement for PFS cutting accuracy. In fact, the biggest challenge in designing the PFS is achieving the required fit accuracy.

The accuracy goal for the PFS is therefore to achieve an implant-to-bone gap of 0.5mm or less over a significant portion of the implant.

To measure fit accuracy, we will laser scan samples cut by the PFS and find a least squares fit between the scan and the target shape. Of course, this assumes that some of the bone material intersects the implant. A better estimate of fit accuracy would require that we calculate how the implant rests on the high parts of the cut surface. However, this is complicated by the fact that some of the peaks on the surface will be leveled by the implant. There are two reasons for this: first, some bone material is compressed by the press fit. Second, additional bone material may subside if the load of the implant is concentrated on a few small peaks. Because of the complexity of estimating how much the peaks will be flattened, we will use least squares error to approximate the implant fit.

The one factor that affects position accuracy but not fit accuracy is registration. This document largely ignores registration error, and position accuracy. There are several reasons for this. First, as described above, fit accuracy is the more challenging requirement. Because of the way the PFS cuts, if fit accuracy can be achieved position accuracy will most likely be achieved as well. Second, registration error depends on the registration technique used, which is not the focus of this thesis. Registration algorithms have been developed and are used in CAOS systems which have proven accurate, and the PFS is compatible with most registration techniques. As a corollary to this, most registration techniques are compatible with most CAOS systems, so registration error is not a differentiator between the PFS and other CAOS systems.

Chapter 5. The PFS Handheld Tool Mechanism

Good cutting performance begins with a good PFS tool mechanism suited to the application. The PFS concept can be applied to many applications, and the optimal size, shape, and characteristics of the PFS tool will vary for each. Here we describe the PFS tool designed for the less-invasive PFS-specific UKR procedure. In addition to application-specific requirements such as size and shape, every PFS mechanism must enable accurate and efficient cutting. For cutting accuracy, the tool should allow the computer to reliably control cutting, and should enable accurate modeling so that the computer knows exactly what has been cut. Cutting efficiency depends on both mechanical and ergonomic factors.

The PFS tool developed for the PFS-specific UKR is seen in Figure 5-1. The figure also shows a closeup of the blade and the guard that it retracts behind. The blade can extend and retract independently in the axial and radial directions. In each direction, the blade can extend up to 2mm beyond the guard and retract up to 2mm behind the guard. It can cover this distance in 100ms. The 6mm capsule-shaped blade, and the drill motor that drives it, are off-the-shelf orthopedic tools. Radial blade extension is driven by a DC gearmotor fitted with encoder. Axial blade extension is driven by an ultrasonic motor fitted with encoder.

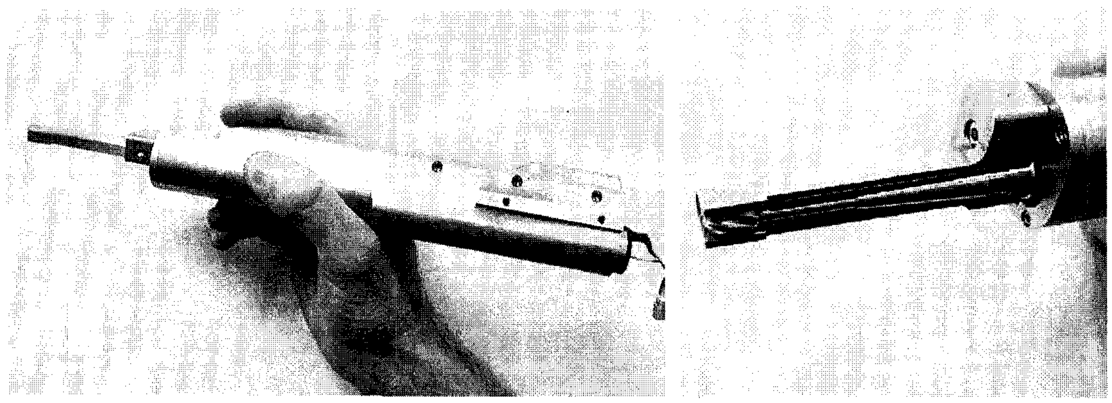


Figure 5-1: PFS surgical tool (left), and closeup of tip (right)

The guard of the tool was designed to fulfill the needs of the PFS-specific UKR, and to enable less invasive surgery. The slender part of the guard is long enough to reach the back of the tibia. The thickness of the guard is the largest size that can fit between the femur and tibia during the procedure. For maximum efficiency, the blade is the largest that can fit in the guard.

For the PFS-specific UKR, the tool is tracked with 4 tracking markers arrayed around the back of the tool. The tool is calibrated with a ball-pointed probe which identifies 3 precisely located divots on the guard of the tool.

The tool was designed by Vic Eggenberger, a mechanical engineer consultant, per specifications I developed based on the UKR task.

5.1 Initial Investigations with Unactuated PFS Mockup

Designing a PFS mechanism from scratch is difficult, because many of the characteristics of the cutting process involve human interaction or complex cutting processes and cannot be easily calculated theoretically. To this end, development of the surgical prototype was guided by early experiments with the unactuated mockup in Figure 5-2. The mockup features a detachable guard and a blade that can be manually positioned at a desired extension and locked in place.

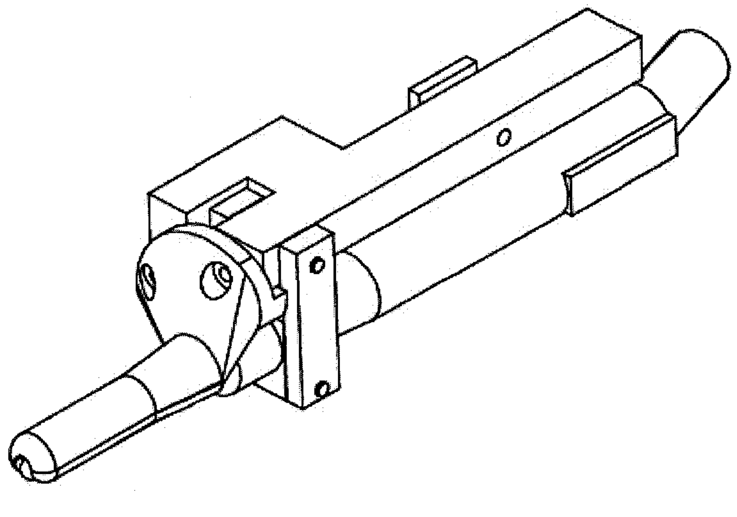


Figure 5-2: Unactuated PFS mockup helped to choose design parameters.

The mockup was used to cut cow bones with the blade locked at a particular extension. Off-the-shelf blade designs and drill handles were evaluated with this design. Guard designs were printed with FDM and could be attached and tested for rapid design iterations. The cutting efficiency of various extension distances was evaluated by measuring the time necessary to remove approximately the amount of bone removed in UKR.

5.2 Blade and Drive System

Two drill handles were evaluated for the PFS using the mockup. The Hall Surgairtome air-powered tool was used first. However, this tended to stall under heavy cutting. The PFS allows surgeons to make much heavier cuts than are usual with tools like the Surgairtome, because the PFS guard regulates cutting depth, and provides some extra stability when the guard rests on the bone. We next evaluated and selected for use the Stryker TPS MicroDrill. This electric tool spins at up to 70,000 RPM, and features electronic speed control, so that the motor does not stall.

The blades used are modified Stryker TPS blades, 6mm diameter and capsule shaped. These come in standard and "aggressive" models. The aggressive model has fewer, larger flutes, and is the only blade we tested that had adequate material removal rate. The capsule shape was selected because the spherical tip allows cutting from any angle, but the cylindrical body allows faster material removal when it can be used. The 6mm diameter was chosen to be as large as possible while allowing the blade and guard to fit between the femur and tibia.

Cutting with the mockup demonstrated that irrigation of the blade was necessary. Without irrigation, bone debris would fill the inside of the guard and then jam the gullets of the blade, making it impossible to cut. The guard of the PFS surgical tool features a 0.045" ID tube embedded into the inner wall of the guard, which delivers an irrigation spray directly to the side of the blade. The channel that the tube lies in is filled with sterilizable epoxy to present a smooth surface for easy cleaning.

5.3 Guard Design for LIS

The PFS guard was designed to enable less invasive surgery. The guard is long and slender to allow access into constrained areas and through small incisions. It is also designed to allow cutting from a wide variety of angles.

The UKR procedure presented two major constraints on the shape of the guard. It must be long enough to reach the rear of the tibial cut, and it must be small enough to fit in between the femur and tibia, once the appropriate cuts are made to open up space. We made the slender part of the guard 2.5" long, and the guard height 0.35". (Figure 5-3)

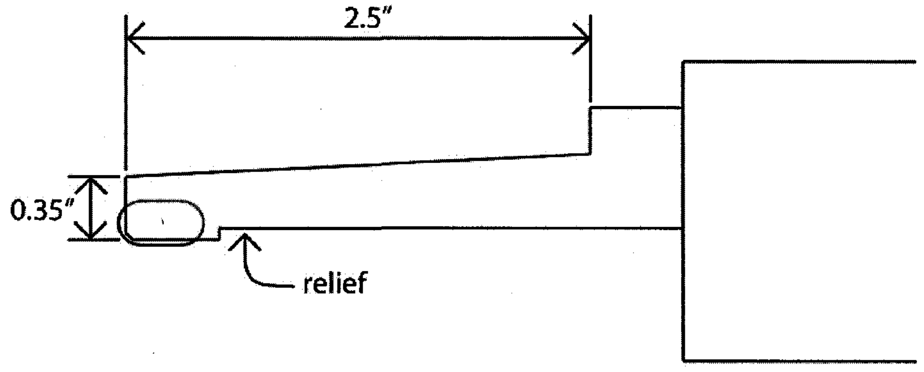


Figure 5-3: Guard dimensions for UKR procedure, and relief behind the guard.

Another important aspect of the guard design is that the contact surface of the guard is relieved behind the blade position as in Figure 5-3. During tests with the mockup, the rear of the guard would rest on the workpiece and prevent the blade from making contact with the workpiece and cutting. (Figure 5-4) Adding this recess makes it easier to engage the blade.

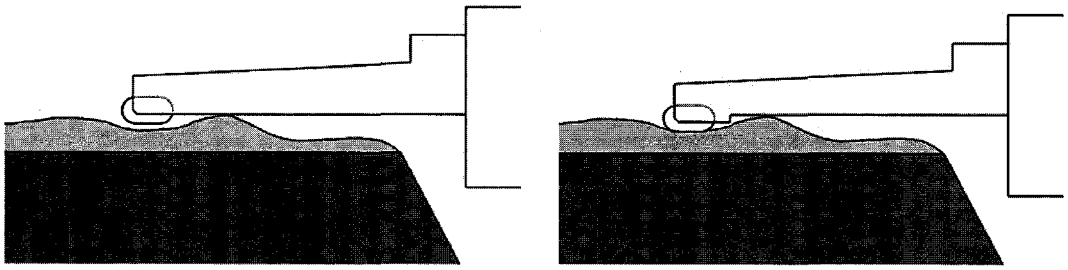


Figure 5-4: Left, guard rests on workpiece and prohibits cutting. Right, guard is relieved to enable blade to reach workpiece.

One final important factor in design of the guard is the shape of the *cutting surface*. Cutting surface refers to the surface spanning the opening in the guard, through which the blade extends. For cutting efficiency, the guard needs to rest nearly flat on the workpiece so that the blade can fully engage the work (Figure 5-5). This means that the normal of the workpiece should match the normal of the cutting surface at the point of contact. On the PFS tool, the cutting surface curves through 90° to cover the front and side openings of the guard, through which the cutter extends axially and radially, respectively. Compared to just a planar cutting surface, the curved cutting surface allows the surgeon a greater range of angles with which the tool may approach the bone.

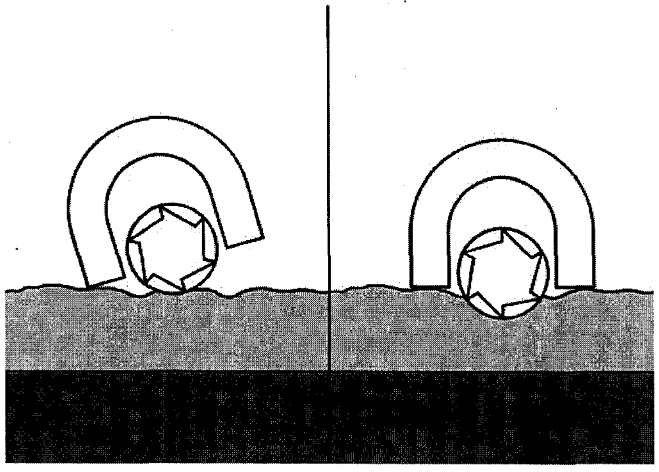


Figure 5-5: Left, guard does not rest flat on workpiece and cutting ability is limited. Right, guard rests flat on workpiece for maximum cutting efficiency.

The curved cutting surface of the PFS is especially important for LIS, because the incision may limit the tool's approach angle. The limiting case for LIS is arthroscopic access, where the tool is inserted through a single fixed portal in the skin. This allows the tool 4DOF: The tool can rotate in three axes about the insertion point, and can translate in the insert/remove direction. (Figure 5-6)

A planar cutting surface, with a single normal vector, requires 5DOF to position it tangent to an arbitrary point on the target surface. However, the curved cutting surface of the current PFS design requires only 4DOF to do so. Therefore, with an endoscopic-type access, a planar cutting surface could in general not be positioned for maximum cutting efficiency, whereas the curved cutting surface of this PFS design could. Although the PFS does not actually operate through an endoscopic portal, the effect is similar with a limited size incision: the small incision restricts the angle with which the tool can approach the bone, and the curved cutting surface may be necessary to fully engage the tool.

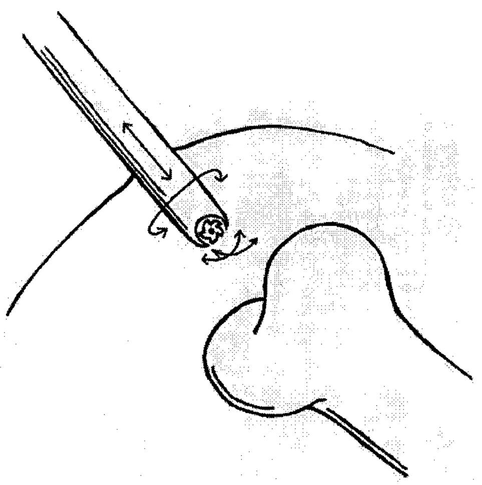


Figure 5-6: Arthroscopic entry offers 4DOF.

5.4 Cutting Control by Extension and Retraction

The PFS computer extends and retracts the blade to control what material the PFS removes. The blade can extend independently in the axial and radial directions to a maximum extension of 2mm beyond the guard and maximum retraction of 2mm behind the guard. The blade can also be positioned at any intermediate distance.

As the surgeon moves the tool across the bone surface, the blade must extend and retract to remove only the appropriate bone. As cutting nears completion, the blade must transition quickly and frequently between extension and retraction. For accurate results, the tool must extend and retract as quickly as possible. It is also important that transitions between extension and retraction be as seamless as possible so that the surgeon's motion is not interrupted.

Accurate blade location is also important for cutting accuracy. Mechanical accuracy is important in the blade extension mechanism so that the tool cuts exactly the right amount of material. It is also important that the computer model of the target shape be as accurate as possible, because it affects cutting accuracy down the road. To achieve an accurate model, the Optotrak-based measurement of blade position should be as accurate as possible. The optical calibration method for the tool was carefully designed for accuracy.

Extension Distance

The maximum extension distance is set at 2mm based on experiments with the mockup. This extension was necessary to remove the amount of bone for UKR in an acceptable amount of time. However, the maximum extension should not be increased arbitrarily, because a larger extension requires faster retraction speed to retract in a given amount of time. One further consideration for maximum extension distance is curvature of the target surface. If the target surface is concave, the blade needs to extend at least so that it can reach the target shape. (Figure 5-7)

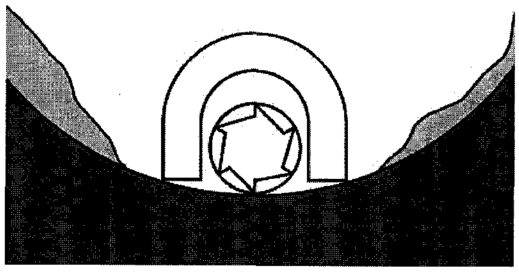


Figure 5-7: Blade needs to extend further when cutting concave target shape.

The maximum retraction distance is also 2mm. With a flat target shape, the blade should only need to retract to the level of the guard, but if the target shape is convex, more retraction is necessary to avoid cutting the target shape. (Figure 5-8) On the other hand, greater retraction distance requires a taller guard. The convex corners on the femoral target shape require about 1.4mm retraction to fully avoid overcutting. Corners on the boundary where the target shape meets uncut bone are sharper, and may require more retraction. The 2mm maximum retraction is a compromise between overcut prevention and a low-profile guard.

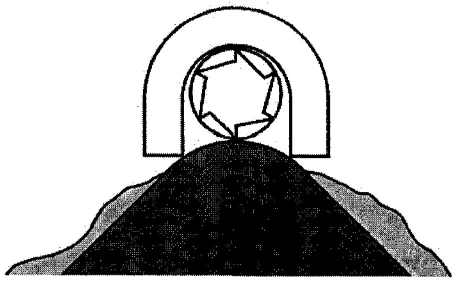


Figure 5-8: For convex target shape, more retraction is necessary.

Extension / Retraction Speed

Specifying the required retraction speed is challenging. Faster retraction is always better, but what retraction speed is adequate to achieve a desired cutting accuracy? In Chapter 8 we will develop some tools to address this question. When the PFS tool was developed, however, none of this was understood. The retraction speed was specified as 100ms over the 4mm range of motion simply because that seemed reasonably fast, but achievable.

Forces on the Blade

To measure the cutting forces that the mechanism and blade extension motors would need to resist, we attached samples of bovine cortical bone to a force sensor (Nano-17, ATI Industrial Automation). The samples were cut freehand with the TPS drill and the cutting reaction force on the bone was measured. The blade was fed into the bone in conventional cutting direction with the highest feed rate that did not cause it to chatter violently. The maximum forces seen were 2.5N in the radial direction, 6N in the axial direction, and 10N in the tangential direction.

Extension Mechanism

The blade extension mechanism is illustrated in Figure 5-9. The blade and spindle (a) turn in the spindle carrier (b), which moves to extend and retract the blade. The spindle carrier slides on the post (c) for axial motion and approximates radial motion by rotating around the post. Axial motion is driven by an ultrasonic motor, USR30-B3 from Shinsei Corp, Tokyo; and radial motion is driven by a DC gearmotor, Faulhaber 1524SR with gearbox and encoder. Both motors are fitted with encoders on the motor shaft. The DC gearmotor was position-controlled with an off-the-shelf motion controller, Faulhaber MCDC2805. The ultrasonic motor was driven by a driver box that is sold with the motor, and position controlled by the PC fitted with an encoder counter card.

The dynamic response of the blade extension motors was tested by recording the encoder positions while commanding the motors from full extension to full retraction. The results are illustrated in Figure 5-10 and Figure 5-11. The DC gearmotor for radial extension is measured to have an acceleration of 2400mm/s^2 and a top speed of 52mm/s. The ultrasonic motor has virtually no acceleration phase, but there is an odd "elbow" in the curve where it starts and one constant velocity and then transitions to a faster velocity after about 30ms, which is currently unexplained. The velocity of the initial stage is about 17mm/s.

Unfortunately, the data in these tests were not synchronized with the command to the motors, so the latency from command to the reaction of the motors was not measured. This latency was later found to be significant, as we will discuss in Chapter 9.

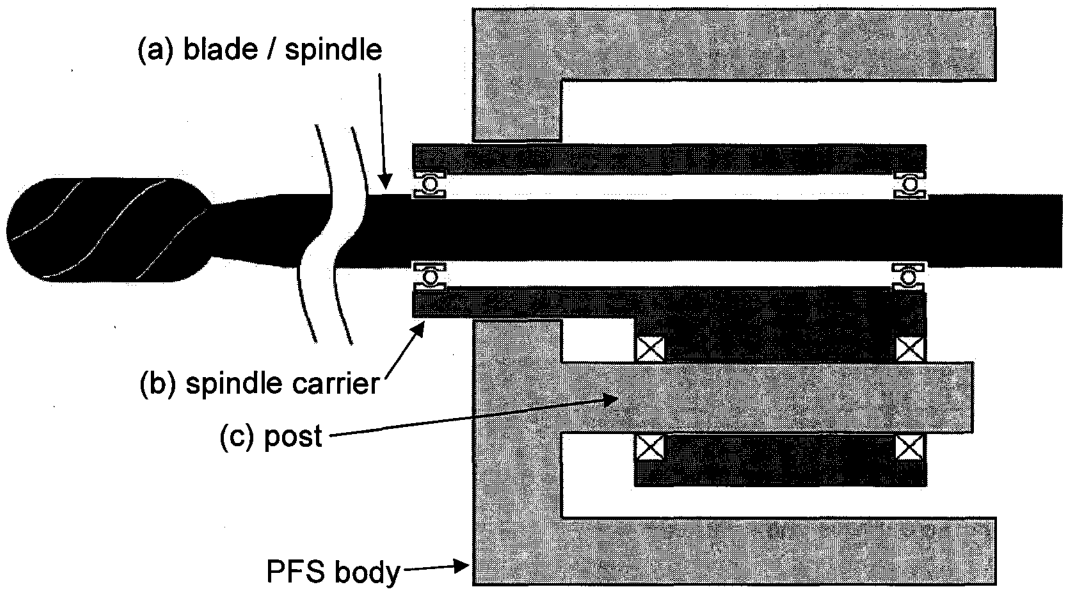


Figure 5-9: Blade extension mechanism in cross-section. The spindle carrier (b) slides and rotates on post (c).

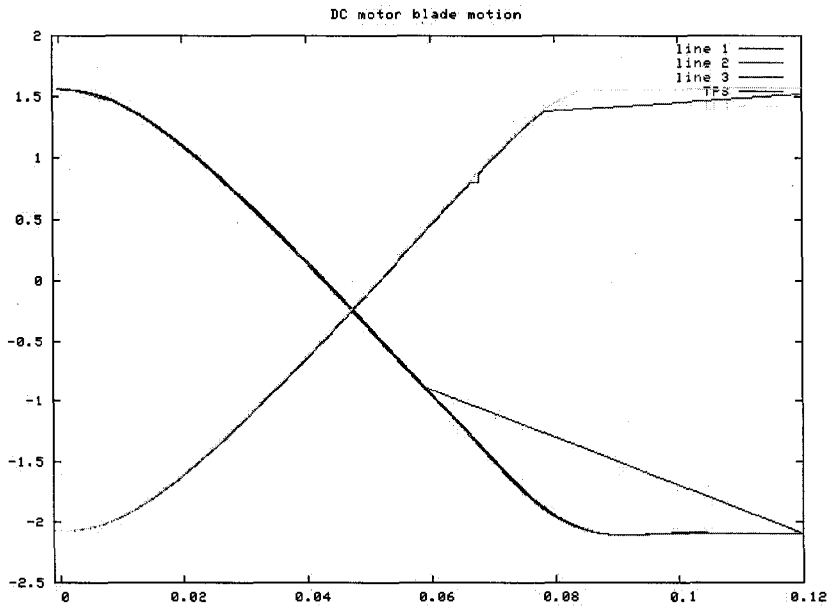


Figure 5-10: Dynamic response of DC gearmotor for radial blade extension. The spurious straight lines are caused by missed samples.

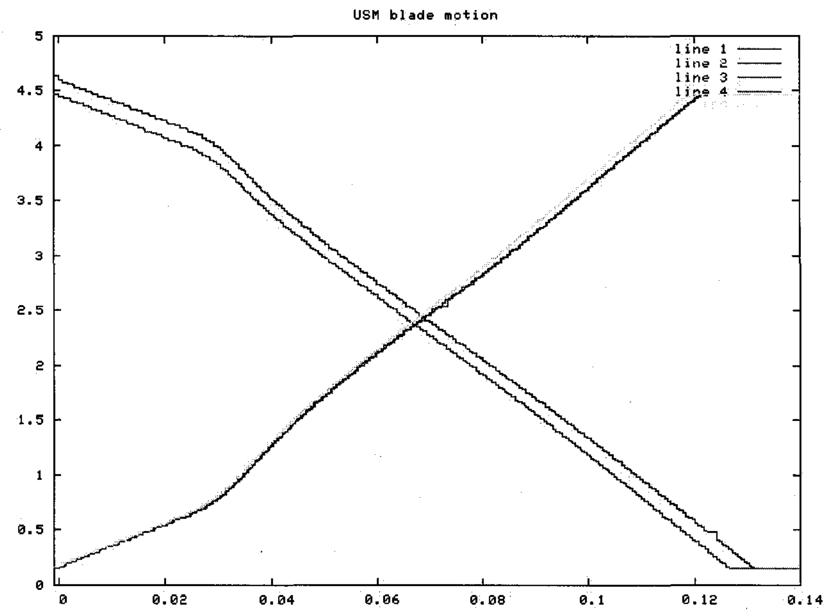


Figure 5-11: Dynamic response of ultrasonic motor for axial blade extension. The "elbow" in the curve is unexplained.

We chose the ultrasonic motor (USM) on a trial basis based on its superior transient response. The USM has roughly the same power output (0.1Nm * 250RPM = 1.3W) as a DC gearmotor of comparable size, but with a maximum shaft speed of 250RPM, the motor reaches top speed much faster than a DC gearmotor. Although its performance has been impressive, several reliability issues have led us to decide against using this motor in the future.

Since the blade extension motors are instrumented with incremental optical encoders, the position of the blade must be homed each time the tool powers up. This is done by running the motors to the end of travel. The offset between the end of travel and the position of zero extension with respect to the guard is calibrated once and then used every time the tool is homed.

Importance of the Guard for Cutting Control

One of the most important aspects of how the current PFS prototype controls cutting is the role of the guard. The role of the guard in this prototype cannot be overstated. The guard serves two important functions. First, it allows the computer to precisely select the depth of cut. Second, it makes the transitions between cutting and not cutting seamless.

One role of the guard is to precisely control the depth of cut. Since the guard cannot penetrate the workpiece, it limits how far the blade can penetrate. This is important because the software can only respond to user motion at the rate it receives Optotrak position readings. With the guard, the software can precisely set how far the blade should be allowed to cut before the next Optotrak update. In contrast, we can consider two alternative configurations: the PFS controls cutting by moving the blade, but there is no guard; or the PFS controls cutting by stopping and starting blade rotation. In both of these, the PFS software has no way to control how far the blade cuts in the time between Optotrak updates.

The guard also serves an important role in making cutting transitions seamless. When the blade retracts, the force with which the surgeon presses the tool against the bone is taken up by the guard, so that the tool does not give away. Likewise, when the blade extends it does not cause the tool to jump because the surgeon's force is just transferred from the guard to the blade. Seamless transitions are important, because the PFS needs to transition very frequently as cutting approaches completion. If each transition disrupted the tool's motion, the PFS would be impossibly slow and cumbersome to use. For instance, if the PFS controlled cutting by starting and stopping blade rotation, each start might cause the blade to kick off the surface and interrupt the surgeon's motion.

5.5 Optical Calibration Routine

For accurate cutting, the PFS software must accurately know where the blade is with respect to the workpiece. This is important for controlling cutting at the current moment, but also for accurately updating the model of the workpiece to reflect what has been cut. The worksurface model is used in calculating how far to extend the blade, so errors in the model can lead to cutting errors down the road.

An accurate calibration is necessary for accurately calculating the position of the blade based on Optotrak data. The calibration represents the position of the blade with respect to the tracking markers mounted on the tool. Because of manufacturing inaccuracies, the location of each optical marker's frame of reference with respect to the metal frame of the marker is unknown. [Crouch 2005] Therefore the calibration must be determined experimentally.

Pivot calibration is the standard technique for calibrating objects such as a probe, which have a sharp or ball-shaped tip whose position must be known. The probe tip is kept stationary by touching it to a stationary divot, while the rest of the probe is pivoted around that point and tracked by the tracking system. The frame of reference of the optical marker attached to the probe moves along a sphere centered at the probe tip. By calculating the center of this sphere as traced by the origin of the probe marker frame of reference, the location of the probe tip with respect to the marker frame of reference is determined.

Typically, to calibrate more complex tools, a probe is calibrated and then touched to specific points on the complex tool to find their positions with respect to the complex tool's marker. However, this secondary calibration introduces one more measurement error as compared to direct pivot calibration, and so is not preferred. We calibrate the surgical tool with pivot calibration directly for maximum accuracy.

For accuracy, the calibration routine uses a ball-tip probe instead of a sharp-tipped probe. The ball probe is pivoted in a cone-shaped hole. The accuracy of this pivoting relies only on the sphericity of the ball and the radial symmetry of the hole.

The guard of the tool has 3 holes for pivoting: (Figure 5-12) one near the tip, used for position and orientation, and two further back, used for orientation only. The wide baseline between the divots minimizes orientation errors in calibration. The ground-truth position of these divots can be found by placing the ball-tip probe in each divot and measuring the position of the ball with a micrometer or CMM.

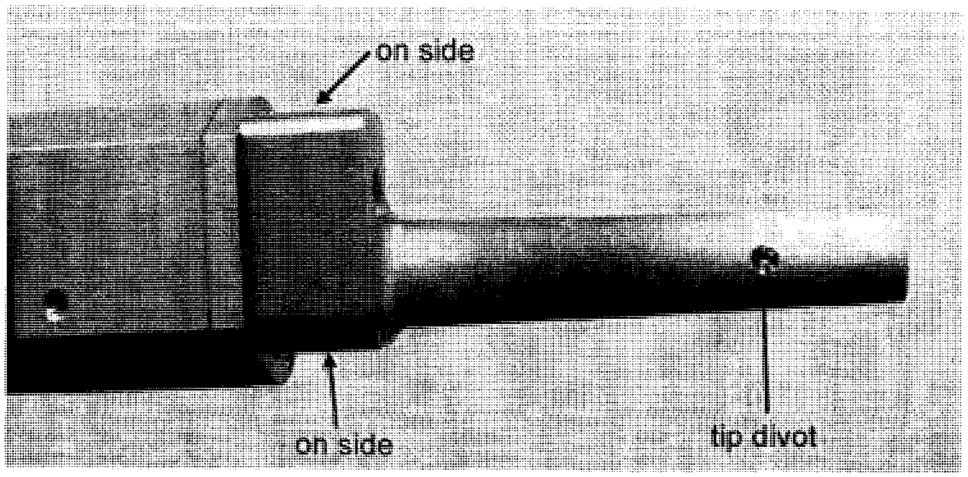


Figure 5-12: Calibration pivot point locations. A ball-tip probe is pivoted in the tip divot, and then touched to the other two divots to determine orientation.

To calibrate the PFS, the probe is pivoted in the divot closest to the tip. This serves as a pivot calibration not just for the probe, but also for the tool: both the probe and the tool pivot around the center of the sphere. The probe is then touched to the other two divots to determine their locations. Horn's method [Horn 1987] finds the best match between the sampled divot locations and their measured ground truth locations, yielding a transformation between the marker coordinate system and the tool coordinate system. Because pivot calibration was used for the tip divot, its position is known with the best possible accuracy. Therefore we shift the calibration to have zero error at the tip divot. Since the blade is very close to the tip divot, the effect of orientation errors on the blade's perceived position will be small.

5.6 Electronics and Wiring

The PFS handheld tool is connected via wires to the devices which power and control it. (Figure 5-13) These devices are: the PC that runs the PFS software; a box of custom electronics that control and drive the blade extension motors; the TPS drill control unit, and the Optotrak control unit.

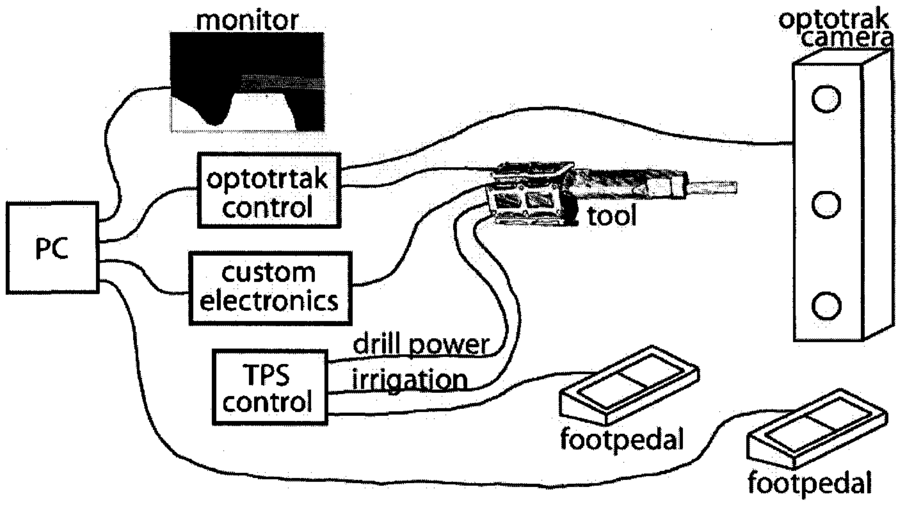


Figure 5-13: PFS is connected to PC, custom electronics, TPS drill control unit (off-the-shelf), and Optotrak control unit (off-the-shelf).

Several cables connect the PFS handheld tool to these off-board devices. The design goal was to minimize the encumbrance that the cables add to the tool, although complete integration was impractical because of limited development time. The cables attached to the PFS are: one cable for the TPS drill, one cable for the actuation motors, and one cable for each of the four tracking markers. Additionally, a tube delivers irrigation fluid to the tool. The cables and tube all attach to the rear of the PFS tool.

The TPS control unit is the off-the-shelf driver that controls the TPS drill. The control unit, drill, and cable which connects them were used without modification. To interface the drill with the PFS, a custom hex-socket bit is attached to the drill, and the drill is inserted into the rear of the PFS and locked in place with setscrews. The cable can be attached and detached. The drill speed is controlled via a footpedal attached to the TPS control unit.

The TPS control unit also features an irrigation pump which we use to supply irrigation to the tool. The control unit turns the pump on and off as the drill is turned on and off.

The tracking markers were also used unmodified. They were screwed to a holder attached to the PFS tool. The cables emerge at the rear of the tool and were tied together to be more manageable. The tracking cables are very thin and light and did not add much to the encumbrance of the tool. The tracking cables attach to the Optotrak control unit.

The custom electronics box which interfaced to the blade extension motors contained drive electronics and power supplies, and interfaced to the PFS tool on one end and the PC on the other. The DC gearmotor was position-controlled by an integrated controller / driver, Faulhaber MCDC2805. A serial port was used to send setup commands from the PC to the MCDC2805, but was not fast enough for continuous position updating. Therefore, an analog output from the PC was sent to the MCDC2805 to command position. The encoder signal from the DC gearmotor was split to go to the MCDC2805 for position control, and to the PC for position monitoring.

The ultrasonic motor (USM) was driven by a driver box that comes with the motor. This box creates a precise sine-wave output, causing a ring of piezo elements in the USM stator to move the rotor. For position control, it was adequate to turn off the USM when the desired encoder count was reached. The USM locks its position when turned off. To turn off the USM at the desired position, the encoder card in the PC generates an interrupt, and the PC interrupt handler turns off the USM.

The PC was fitted with an encoder counter card which monitored the position of both extension motors, and a multipurpose I/O card which provided digital and analog lines to interface with the custom electronics box.

The PC also drives the monitor for the graphical display. A footpedal connected to the PC allows the surgeon to step through viewpoints appropriate to each cut in the procedure, and to rotate one 3D view in the display.

Chapter 6. Workpiece Modeling

The PFS requires a model of the worksurface (working surface of the bone) that is updated as material is removed. This is used by the computer display, but more importantly it is used by the blade control algorithm to predict how closely the tool will approach the target surface when the guard rests on the workpiece. For accurate blade extension, the model must accurately reflect the waste thickness.

6.1 The Heightfield Model

The most common candidates for implementing the worksurface model are voxel-based and surface-based (e.g. triangular mesh) methods. Voxel methods are easy to update as cutting progresses, but the accuracy of measuring waste thickness is limited to the voxel resolution. To achieve the accuracy we desire (better than 0.1mm), the number of voxels becomes intractable (64 million for a 40mm cube with 0.1mm voxels).

On the other hand, surface-based models can accurately represent the thickness of waste, but updating is difficult. With an Optotrak update rate of 83 readings per second, a surface model will quickly become intractable unless some provision is made for trimming vertices and simplifying the mesh.

In light of these shortcomings, we have chosen instead to use a 2½D heightfield ("displacement map") model, where the surface is represented by a set of displacements from some fixed "base" surface. In the case of PFS, the target surface serves as the base surface. A heightfield represents the thickness of the waste bone as a floating point number, unlike voxels which have a limited resolution. Unlike triangular mesh models, the heightfield can be easily updated by updating the recorded displacements.

To represent the worksurface, a grid of points is arranged over the base surface. Emanating from each point is a vector, the tip of which represents a point on the worksurface. We refer to this vector as a "heightvector", and the length of the vector as the "height" associated with the vector. The base of the vector, which lies on the base surface, we refer to as a "base point", and the tip of the vector, which lies on the worksurface, as a "height point". The surface represented by the heightfield is updated by changing the height of the heightvectors. The direction of the heightvectors remains constant.

To display the worksurface, the graphical display requires the heightfield model to generate a triangular mesh of the worksurface. If a triangular mesh connectivity of the base points is known, it can be applied to the heightpoints for this purpose. However, if the heightvectors cross over each other, the triangular mesh of the worksurface may have self-intersections which are visually confusing. The opposite problem is that diverging heightvectors may leave a large space unsampled. (Figure 6-1)

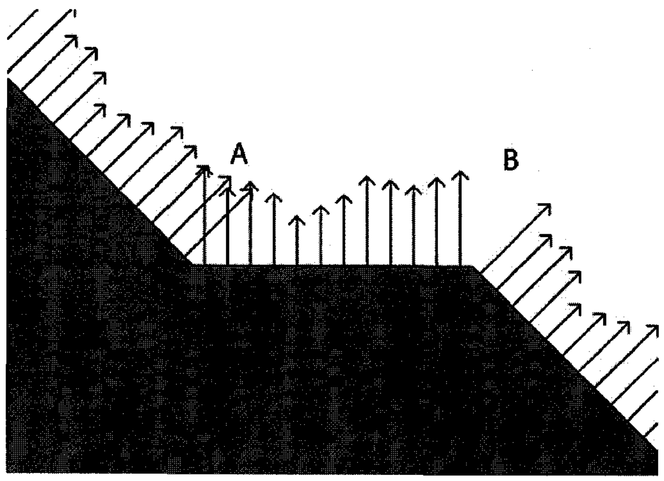


Figure 6-1: Self-intersecting surface (A)resulting from heightvectors crossing over. Diverging heightvectors (B) leave a space unsampled.

To avoid this problem, the worksurface model is based on the "slabs" data structure of [Jagnow 2002]. A triangular mesh is used for the base surface. A *slab* is a volume extruded upward and downward from a triangle on the base surface. Adjacent slabs share a common boundary, so the slabs neither overlap nor leave any space between them empty. (Figure 6-2) Each triangle on the base surface may be covered with many base points. The directions of heightvectors on a slab are interpolated smoothly between the slab boundaries so that they maintain even spacing and do not cross over each other. This allows the known connectivity of the base points to generate a triangular mesh of the worksurface that samples the surface with uniform density and is free from self-intersection.

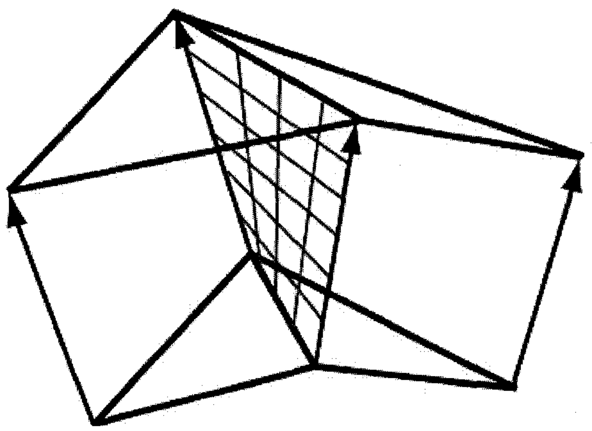


Figure 6-2: Adjacent slabs share a common boundary. (From [Jagnow 2002])

6.2 Updating the Model

With each Optotrak reading, the heightfield model is updated by calculating the position of the PFS blade with respect to the model, and removing all material that intersects the blade. To achieve this update, the intersection of each heightvector with the blade is calculated and the height of the heightvector is updated if necessary. For computational efficiency, this is implemented with a hard-coded function that finds the intersection between a vector and a capsule. (Figure 6-3). The function is adapted from [Cychosz 1994].

The vector-capsule intersection function determines whether the line described by the heightvector intersects the capsule, and if so, at what distance from the base of the heightvector. The function also determines the surface normal of the capsule at the intersection point. When the workpiece is updated by intersection with the capsule, the capsule's surface normal is recorded to represent the normal of the workpiece surface at that location of the heightvector. The surface normal is used by OpenGL to calculate shading on the worksurface.

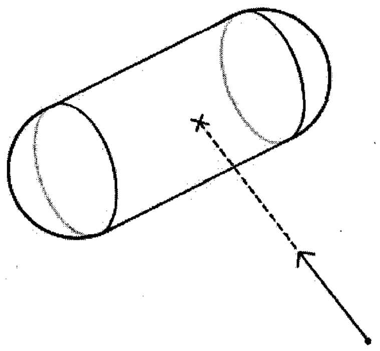


Figure 6-3: Algorithm calculates distance along each heightvector to where it intersects capsule. If distance is shorter than heightvector length, then heightvector length is updated. The algorithm is repeated for each heightvector in the model.

To update the heightfield model, the vector-capsule intersection test is run for each heightvector in the model.

For the vector-capsule intersection, let the capsule be described by:

- $\mathbf{x}_{\mathbf{cap}}$: Capsule position. Position of the center of one sphere on the capsule
- $\mathbf{d}_{\mathbf{cap}}$: Capsule axis. Unit vector from c_{pos} to the center of the other sphere
- l_{cap}: Capsule length. Length of the cylindrical section of the capsule
- r_{cap}: Capsule radius

Let the heightvector be described by:

- $\mathbf{x}_{\mathbf{vec}}$: Position of the base of the heightvector, which corresponds to zero height
- $\mathbf{d}_{\mathbf{vec}}$: Unit direction of the heightvector
- l_{vec}: The recorded length of the heightvector

The algorithm proceeds in two steps. First, the intersection of the vector with the infinite cylinder extending from the capsule is checked. Second, if necessary, the intersection of the vector with one of the cylinder end caps is calculated.

Intersection of the heightvector with infinite cylinder

A diagram drawn in the appropriate reference frame is tremendously useful in understanding the geometry. In Figure 6-4, the plane of the page is chosen normal to the cross product of $\mathbf{d}_{cyl}$ and $\mathbf{d}_{vec}$, so that both are parallel to the page, though not in the same plane. Assume here that $\mathbf{d}_{vec}$ lies in the plane and $\mathbf{d}_{cyl}$ does not. The algorithm proceeds as:

```
$\mathbf{n} = \mathbf{d}_{vec} \times \mathbf{d}_{cap} / \| \mathbf{d}_{vec} \times \mathbf{d}_{cap} \|$        (n is normal to the page in Figure 6-4)
$m = \| \mathbf{d}_{vec} \times \mathbf{d}_{cap} \|$        (component of $d_{vec}$ perpendicular to $d_{cap}$)
$\mathbf{v} = \mathbf{x}_{cap} - \mathbf{x}_{vec}$
$d = | \mathbf{v} \cdot \mathbf{n} |$        (distance of closest approach between heightvector and capsule axis)

if $d > r_{cap}$
    There is no intersection. Algorithm returns.
else
    $t = (\mathbf{v} \times \mathbf{d}_{cap}) \cdot \mathbf{n} / m$        (as seen in Figure 6-4)
    $s = \sqrt{r_{cap}^2 - d^2} \; / m$        (as seen in Figure 6-4. See footnote[1])
```

The heightvector intersects the infinite cylinder at a distance t±s from vbase. This algorithm uses the first intersection with the cylinder, at t-s. Next we check whether the intersection lies on the cylinder body of the capsule:

```
$a = -\mathbf{v} \cdot \mathbf{d}_{cap} + (t-s)(\mathbf{d}_{vec} \cdot \mathbf{d}_{cap})$        (position of intersection along capsule axis)
If $a<0$ or $a>l_{cap}$
    Intersection occurs beyond the cylinder body of the capsule.
    Proceed to next section to find intersection between heightvector and capsule end.
Otherwise, if $l_{vec} < t-s$
    Distance to intersection is less than $l_{vec}$. Update heightvector length and normal:
    $l_{vec} \leftarrow t-s$
    $heightvec\_normal \leftarrow \text{unit}(\mathbf{x}_{cap} + a\,\mathbf{d}_{cap} - (\mathbf{x}_{vec} + (t-s)\,\mathbf{d}_{vec}))$
```

[1] The formula given here for s is an improvement over [Cychosz 1994], who use
$s = \text{sqrt}(crad^2 - d^2) / | v_{dir} \cdot (n \times c_{axis}) |$
This is significant, because in their conclusion, Cychosz et al compare their algorithm to competitive algorithms by counting the number of cross-products. The elimination of one cross-product presented here means that [Cychosz 1994] is faster than competitive algorithms, rather than being equal to them.

The heightvector is updated if the distance to the blade is less than the current heightvector length. When the heightvector is updated, the heightvector normal is also updated. The heightvector normal represents the normal of the workpiece surface at the tip of the heightvector. It is used by OpenGL for rendering the workpiece surface. Here we assume that when the workpiece is cut at a point, the normal of the workpiece is equal (but opposite direction) to the surface normal of the blade that does the cutting. This is more computationally efficient than estimating the normal from the relative heights of surrounding points.

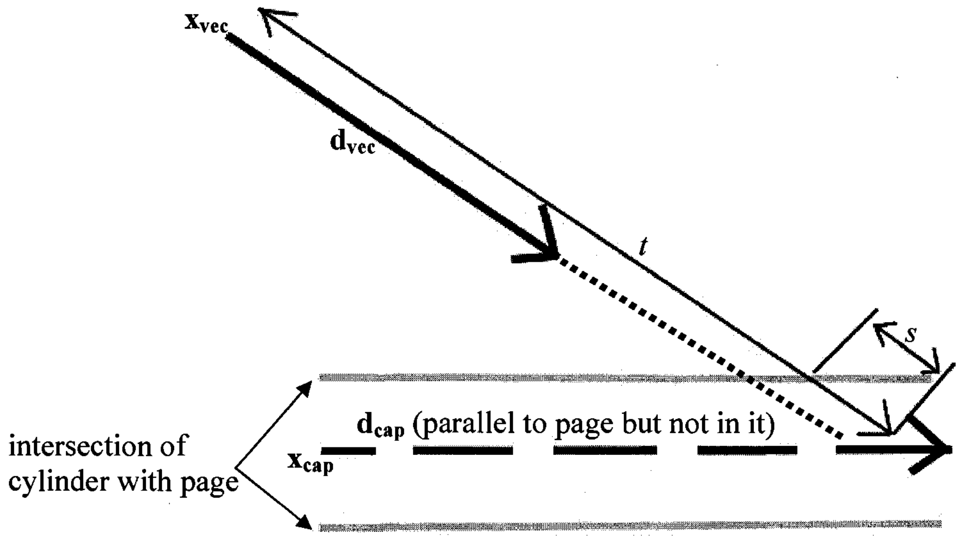

$\mathbf{x}_{vec}$
$\mathbf{d}_{vec}$
t
s
$\mathbf{d}_{cap}$ (parallel to page but not in it)
intersection of cylinder with page
$\mathbf{x}_{cap}$

Figure 6-4. Intersection of heightvector with infinite cylinder extending from capsule. The heightvector lies in the page. The cylinder axis c_{axis} is parallel to the page, but displaced from it. The grey lines indicate the intersection of the infinite cylinder with the page. Intersection occurs at distance t-s along heightvector.

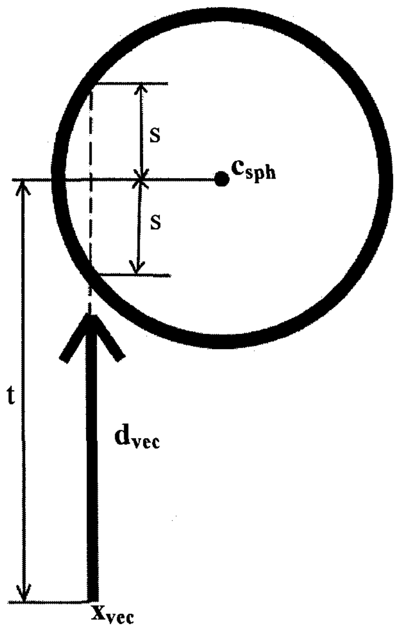


Figure 6-5. Intersection of heightvector with sphere. Intersection occurs at t-s along heightvector.

Intersection of Heightvector with Capsule End-cap

If the heightvector's intersection with the infinite cylinder does not fall on the body, then the heightvector potentially intersects the closer end-cap of the capsule. This requires finding the intersection of the heightvector with a sphere, as in Figure 6-5. The end cap is centered at:

$$\mathbf{c}_{sph} = \mathbf{x}_{cap} \quad \text{if } a<0$$
$$= \mathbf{x}_{cap} + l_{cap}\, \mathbf{d}_{cap} \quad \text{if } a>l_{cap}$$

The algorithm finds the distance along the heightvector to the sphere:

```
$t = (\mathbf{c}_{sph} - \mathbf{x}_{vec}) \cdot \mathbf{d}_{vec}$
if $\| \mathbf{x}_{vec} + t\, \mathbf{d}_{vec} - \mathbf{c}_{sph} \| > r_{cap}$
        There is no intersection. Algorithm returns.
else
```

$$s = \sqrt{r_{cap}^{2} - \| \mathbf{x}_{\mathbf{vec}} + t\mathbf{d}_{\mathbf{vec}} - \mathbf{c}_{\mathbf{sph}} \|^{2}}$$

The heightvector intersects the capsule at $t \pm s$. This algorithm uses the nearer intersection, t-s. Next we update the heightvector if necessary:

```
if $l_{vec} < t-s$
    Distance to intersection is less than $l_{vec}$. Update heightvector length and normal:
    $l_{vec} \leftarrow t-s$
    $heightvec\_normal \leftarrow \text{unit}(\mathbf{c}_{\mathbf{sph}} - (\mathbf{x}_{\mathbf{vec}} + (t\text{-s})\, \mathbf{d}_{\mathbf{vec}}))$
```

6.3 Graphical Display

Graphical display of the worksurface model gives the surgeon feedback on cutting progress, and helps the surgeon position the tool when visualization is limited. The graphical display shows 3D and cross-section views of the tool in relation to the bone. These views are updated as cutting progresses to display the current shape of the workpiece surface. The display features two 3D views and two cross-sectional views as seen in Figure 6-6. Note that the views differentiate the good bone from waste bone so that the surgeon can easily locate waste bone that must be removed.

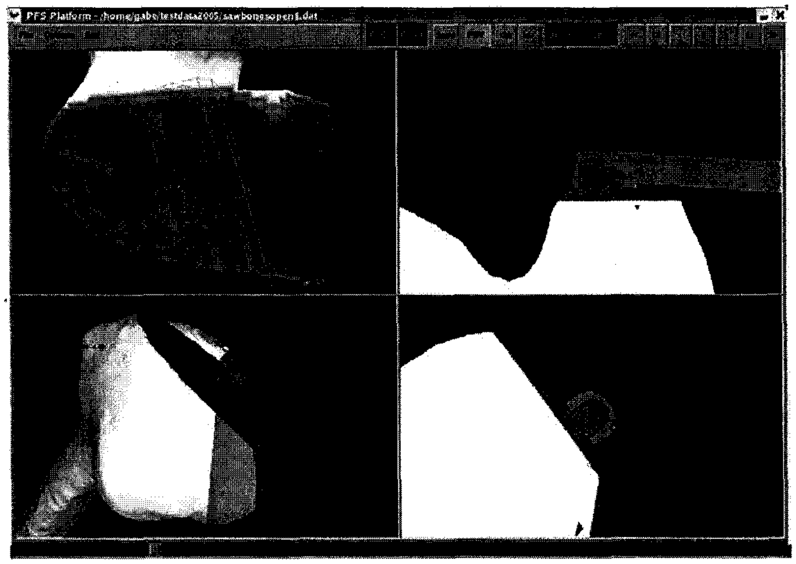


Figure 6-6: PFS Graphical Display.

The graphical display is important for LIS use of the PFS because it replaces the direct visualization lost because of reduced incision size. The display allows the surgeon to see where the tool is touching the bone and monitor cutting progress in areas that are not visible through the incision.

The display is also important in showing the surgeon waste material which must still be removed. Although the PFS ensures that good bone material is not removed, it is still the surgeon's responsibility to keep cutting until all waste is removed. Since there is no material difference between waste and good bone, without the display it would be very difficult for the surgeon to know that all waste had actually been removed, or to locate isolated patches that still need to be cut.

Usability of the display is very important because a more intuitive interface can reduce operating time. One principle that can be applied to the interface design is the "principle of the moving part" [Wickens 2000] which states that when two objects are shown relative to each other, the object that moves in the interface should be the object that moves in real life. In addition to this principle, we will in general try to make the viewpoint on the screen match the surgeon's view as best as possible.

6.3.1 3D View

In the 3D view, the 3D model of the bone is stationary and the tool moves around it. Waste material very close to the target shape is color coded yellow, waste material farther from the target shape is green, and areas that have been overcut are red. The noninvolved bone material is colored white.

We refer to the top left view in Figure 6-6 as a "map view". This straight-on view is ideal for displaying where waste bone must still be removed. The tool is drawn in wireframe on the map view so that the surgeon can see material being removed beneath the blade. The orientation of the bottom left 3D view is left to the surgeon's preference. Dr. Anthony DiGioia prefers a straight lateral perspective for this view. In addition to a preset view, the surgeon can rotate the bottom left view by stepping on a footpedal and rotating the tool. Since the tool's full 6D position is tracked, the surgeon can intuitively command any rotation.

The triangular mesh for the worksurface is derived from the heightfield model, and the surface normals used for lighting the surface are set based on the cutter normal each time a heightvector is updated by cutting. To properly color the model red, yellow, and green, a 1-dimensional texture map is used. This provides crisper results than simply coloring each vertex based on its height and smoothly interpolating the colors. (The texture map technique was developed by Jason Cipriani and Kort Eckman.)

The worksurface model provided by the heightfield model only includes the area of the bone which is waste and should be removed. The good bone, colored white, is provided by a separate surface model called the "background surface" because it provides a context for the waste area of the bone. Using a background surface is optional.

6.3.2 Cross Section Views

Each cross-section view shows the cross-section of a single plane through the tool and the bone. Once again the bone is color-coded: green for waste that must be removed, red for areas that have been cut too far, and white for good bone. The cross-section planes are fixed with respect to the tool, so that cross-section image of the tool is always the same. This means that the bone moves instead of the tool in the cross-section view, which is counterintuitive, but necessary because these simple cross-sections are much easier to understand and more informative than if the bone remained fixed and the tool were cross-sectioned arbitrarily. To make the view move more intuitively, the tool rotates in the cross-section view to match the rotation of the tool as best as possible without changing the plane through which the tool is cross-sectioned. Counter to intuition, however, the tool stays centered in the screen while the bone moves around it.

When operating with limited visibility, our experience was that it was often difficult to place the cutter at a desired location because the tool was blocked by some unknown obstacle of bone. The obstacle was not shown in the display because it did not intersect the cross-section plane. To remedy this, the bottom right cross-section view shows additional parts of the workpiece projected on the cross-section plane and rendered in grey. Only material in planes containing the tool is projected onto the cross-section. (Figure 6-7.) An area of this grey material can be seen to the right of the guard in Figure 6-6. This improvement aided handling of the tool in limited visibility situations.

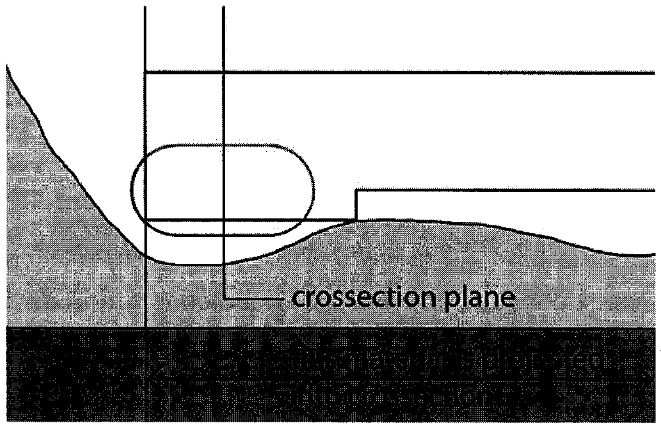


Figure 6-7: Only material in planes containing the tool is projected onto the cross-section.

To create the cross-sections, the OpenGL stencil buffer is used in a method similar to that described in [McReynolds 1997]. (Ben Hollis discovered this method for me.) At each pixel, the number of front-facing triangles in front of the cross-section plane is counted and then the number of back-facing triangles is subtracted from that. (Figure 6-8) If the pixel is on the interior of the object, there will be one more front-facing than back-facing triangle between the cross-section plane and the viewer, and the stencil buffer value will be 1. Otherwise, there are the same number of front-facing triangles as back-facing triangles between the viewer and the cross-section plane, the stencil buffer value will be zero.

This method of drawing cross-sections requires that the triangular mesh used to render the cross-section be completely closed. This provides a firm definition of "inside" and "outside", which allows the model to be defined as a solid object instead of the hollow shell defined by the mesh alone. To generate a completely closed object, the heightfield model provides triangular meshes for the base and sides of the model in addition to the worksurface mesh described by the height points. As in the 3D view, a "background surface" may be used to provide context, e.g. the shape of the remainder of bone which is not to be removed. The background model must also be closed.

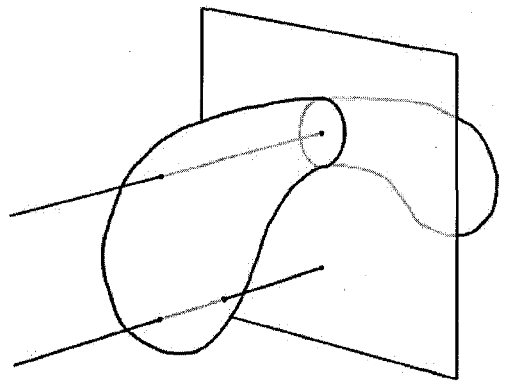


Figure 6-8: Calculating cross-sections from surface models in OpenGL. The top line shows one front-facing surface between the viewer and cross-section plane, so stencil buffer value is 1. The bottom line shows one front-facing and one back-facing surface, so the stencil buffer value is 1 – 1 = 0.

An interesting effect of using the base of the slab model to provide a closed shape for the worksurface is that when the workpiece is overcut this model becomes inverted, with the underside of worksurface protruding through the base of the slab model. Cross-sectioning through such an inversion will result in a stencil buffer value of -1. We use this fact to draw the overcut areas as red in the cross-section view.

Chapter 7. Blade Control for Accurate Cutting

The PFS blade control algorithm is responsible for extending and retracting the blade to ensure that the proper material is removed. The algorithm examines the optical tracking data to decide how far to extend the blade to remove only waste bone. The goals of the blade control algorithm are accuracy and efficiency. Accuracy can always be improved by cutting more cautiously, thus sacrificing efficiency. However, a better algorithm can improve accuracy without cost to efficiency.

To maximize cutting accuracy, the algorithm must predict how far the blade will be allowed to extend in the future, and begin the retraction motion early so that it is complete by the time of prediction. Prediction is necessary not only because of the limited blade retraction speed, but also because of the Optotrak position sensing period of 12ms. The required blade extension may change several tenths of a millimeter in a single Optotrak period, so if the algorithm sets blade extension based solely on the latest Optotrak data, it may differ significantly from the required extension by the time the next Optotrak data is available. Therefore, even with infinitely fast blade extension, the algorithm must at least predict one Optotrak reading ahead in order to avoid overcutting.

Figure 7-1 outlines the blade retraction algorithm developed for the PFS. The algorithm runs each time new Optotrak data is reported. First ("**Update Worksurface Model**" in Figure 7-1), the heightfield data structure is updated by removing all material that intersects the current position of the blade, as described in Chapter 6. The heightfield data structure will be used extensively by the blade control algorithm.

Next the allowable blade extensions are predicted. This is done in two steps. First, the algorithm predicts the future *positions* of the PFS tool. Then the allowable extension is calculated at each position.

Predicting tool positions is the first challenge. Simply extrapolating based on velocity and acceleration is inadequate, because the tool's interaction with the bone is very important. In use, the surgeon presses the tool against the bone so that it maintains contact with the bone surface. The result is that tool rides up and down on the contours of bone.

To approximate this motion of the tool over the bone, the algorithm uses a two step approach. First, the tool position is extrapolated based on velocity and acceleration ("**Extrapolate Position**" in Figure 7-1.) Then the extrapolated position is adjusted so that the tool rests on the bone surface ("**Snap to Surface**").

After the tool position is predicted, the algorithm calculates how far the blade may extend at that future position ("**Calculate Allowable Extensions**"). Since the blade can extend in more than one direction, a single number cannot fully describe the allowable blade extension. Instead, the allowable extension is calculated in five candidate directions distributed through the blade's range of motion.

The algorithm repeats these steps to calculate the allowable blade extensions at several future timesteps (The arrow labeled "**Repeat Several Timesteps**" in Figure 7-1). The timesteps used are the expected times for subsequent Optotrak reports, when the algorithm will run again. Currently, four future timesteps are predicted. In addition to predicted positions, the allowable blade extensions for the current tool position are also calculated.

Once the allowable blade extensions for every candidate direction have been calculated for the current and future timesteps, the predictions must be distilled down to a single direction and distance in which to extend the blade. This is done in two steps. First, a single allowable extension is calculated for each candidate direction ("**Find Extension Constraint**"). These allowable extensions take into account the tool's blade retraction speed to ensure that all future extension constraints can be met by the appropriate time. Next, a single direction is chosen from the five candidate directions as the direction in which to extend the blade ("**Choose Extension Direction**"). Here, the guiding principle is efficiency: the extension direction is chosen which will maximize the amount of material removed.

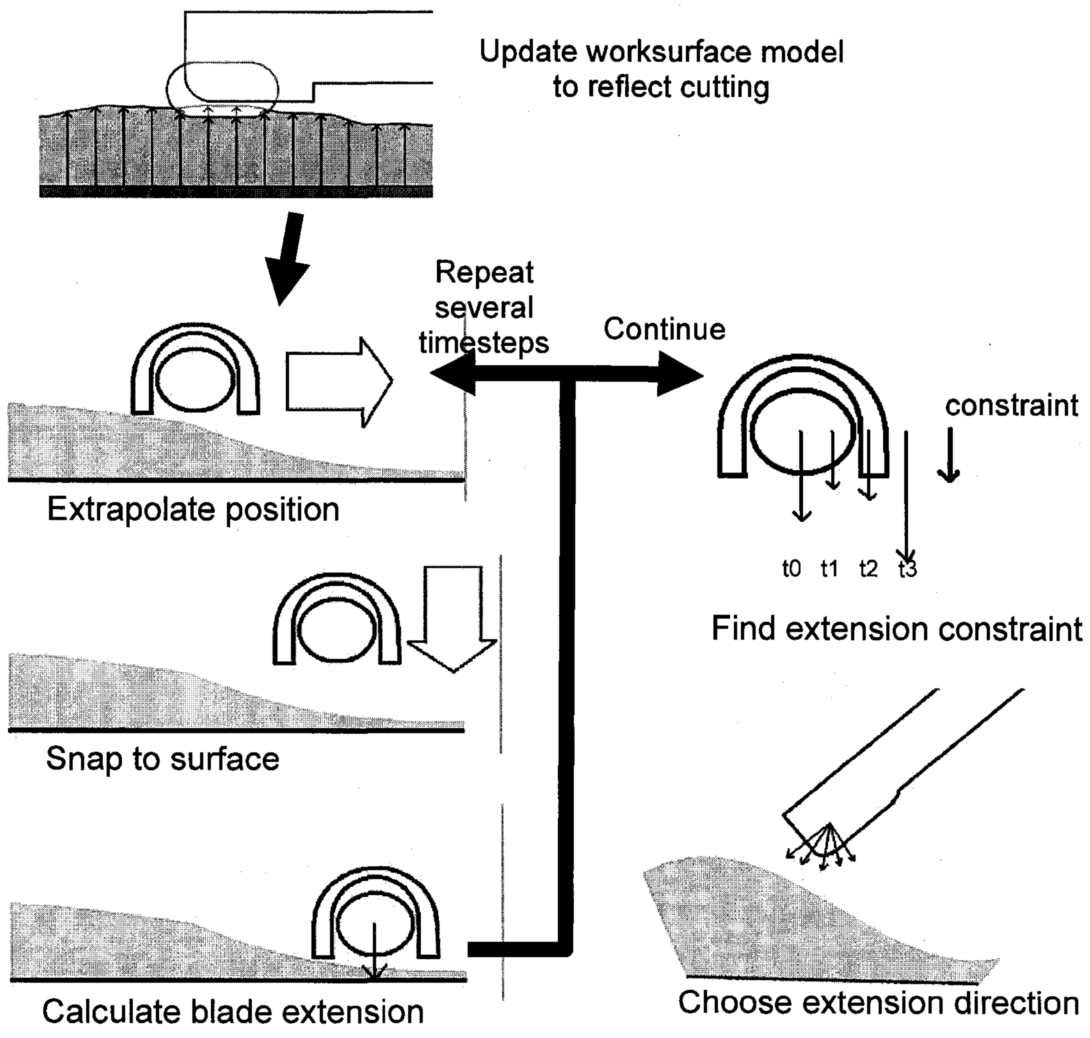

Update worksurface model to reflect cutting
Repeat several timesteps
Continue
Extrapolate position
constraint
t0 t1 t2 t3
Find extension constraint
Snap to surface
Calculate blade extension
Choose extension direction

Figure 7-1: Overall structure of blade control algorithm.

The algorithm has a significant amount of work to perform during each 12ms Optotrak period. The time intensive operations are those that operate on the heightfield model. These are:

- Update Worksurface Model: calculated once
- Snap-to-surface: calculated 5 times: for the current and four predicted positions
- Calculate blade extension: calculated 25 times: At each of 5 timesteps, the extension in 5 candidate directions is calculated.
- Choose extension direction: 5 calculations: The depth of cut is estimated for 5 candidate extension directions

To achieve these calculations within the allotted time, these operations were hard-coded for the shape of the PFS blade and guard.

The remainder of this chapter is devoted to a detailed description of each step in the algorithm. The final section of the chapter discusses calculation times and timing issues.

7.1 Predicting Future Tool Positions (Extrapolate Position and Snap to Surface)

The first step in calculating how far to extend the blade is to predict the future position of the tool. This is not as simple as just extrapolating based on velocity and acceleration: the user pushes the tool against the bone so that the tool rides up and down on the contours of the bone. It is important for the predicted positions to reflect this up and down motion, because any error in the tool's distance from the bone can cause the same amount of error in the predicted blade extension.

To estimate the motion of the tool over the bone, the algorithm first extrapolates tool position based on velocity and acceleration, and then adjusts the position to lie on the workpiece surface. We call the adjustment step "snap-to-surface". Effectively, the extrapolation step approximates motion tangent to the workpiece surface and the snap-to-surface step constrains the prediction to maintain contact with the workpiece surface. The snap-to-surface adjustment is based on the workpiece surface recorded by the heightfield model.

7.1.1 Tool Position Extrapolation

The tool position is extrapolated based on velocity, acceleration, and angular velocity about the coordinate frame of the tool, which is located at the tooltip. Position and velocity are estimated from the current and prior two position readings. With the current tool position represented by $\mathbf{x}_2$ in $\mathbf{R}^3$, and previous positions represented by $\mathbf{x}_1$ and $\mathbf{x}_0$ in $\mathbf{R}^3$, the velocity and acceleration estimates $\mathbf{v}_{est}$ and $\mathbf{a}_{est}$ are calculated by assuming

$$\mathbf{x}_0 = \mathbf{x}_2 - \mathbf{v}_{est}\, 2\Delta t + \tfrac{1}{2}\, \mathbf{a}_{est}\, (2\,\Delta t)^2 \quad \text{and} \quad \mathbf{x}_1 = \mathbf{x}_2 - \mathbf{v}_{est}\, \Delta t + \tfrac{1}{2}\, \mathbf{a}_{est}\, \Delta t^2$$

where Δt is the period of the tracking system (currently 12ms.) This yields

$\mathbf{v}_{\mathbf{est}} = (\mathbf{x}_0 - 4\mathbf{x}_1 + 3\mathbf{x}_2) / (2\ \Delta t)$ and $\mathbf{a}_{\mathbf{est}} = (\mathbf{x}_0 - 2\ \mathbf{x}_1 + \mathbf{x}_2) / \Delta t^2$

Using acceleration to extrapolate can be beneficial if true acceleration is relatively constant. However, for more erratic motion, extrapolating based on velocity may be more accurate, because velocity only requires one timestep to estimate, and hence relies on more recent data. We took a middle approach, taking a weighted average between extrapolation based on velocity and acceleration, and extrapolation based on a velocity-only estimate:

$\mathbf{v}_{\mathbf{only\_est}} = (\mathbf{x}_2 - \mathbf{x}_1) / \Delta t$

With γ in [0,1] as a weighting factor between velocity-acceleration and velocity-only estimates, our estimated position is:

$\mathbf{x}_{\mathbf{3\_est}} = \mathbf{x}_2 + (1 - \gamma)\ \mathbf{v}_{\mathbf{only\_est}}\ \Delta t + \gamma\ (\mathbf{v}_{\mathbf{est}}\ \Delta t + \frac{1}{2}\ \mathbf{a}_{\mathbf{est}}\ \Delta t^2)$

We now include a discussion on the choice of γ. This also serves as an interesting theoretical justification that interpolating between the velocity-only and the velocity/acceleration estimates is a reasonable policy.

First we simplify the expression for $\mathbf{x}_{\mathbf{3\_est}}$ to:

$\mathbf{x}_{\mathbf{3\_est}} = (2+\gamma)\ \mathbf{x}_2 - (1 + 2\gamma)\ \mathbf{x}_1 + \gamma\ \mathbf{x}_0$

Given the true $\mathbf{x}_3$, we can analyze the effect of γ on estimation error. The estimation error $\mathbf{x}_3 - \mathbf{x}_{\mathbf{3\_est}}$ simplifies to

$\mathbf{x}_{\mathbf{err}} = \mathbf{x}_3 - \mathbf{x}_{\mathbf{3\_est}} = (\mathbf{x}_3 - 2\ \mathbf{x}_2 + \mathbf{x}_1) - \gamma\ (\mathbf{x}_2 - 2\ \mathbf{x}_1 + \mathbf{x}_0)$

Note the symmetry of the first term $(\mathbf{x}_3 - 2\ \mathbf{x}_2 + \mathbf{x}_1)$ to the second $(\mathbf{x}_2 - 2\ \mathbf{x}_1 + \mathbf{x}_0)$. Define

$\mathbf{\Phi_n} = (\mathbf{x_n} - 2\,\mathbf{x_{n-1}} + \mathbf{x_{n-2}})$

Making

$\mathbf{x_{err}} = \mathbf{\Phi_3} - \gamma\,\mathbf{\Phi_2}$

Note that under constant acceleration, $\mathbf{\Phi_n} = \mathbf{a}\,\Delta t^2$, a constant. The closer the tool motion is to constant acceleration, the closer $\mathbf{\Phi_n}$ is to $\mathbf{\Phi_{n-1}}$. If $\mathbf{\Phi_n}$ and $\mathbf{\Phi_{n-1}}$ are close enough that they have similar directions, with $\mathbf{\Phi_n} \cdot \mathbf{\Phi_{n-1}} > 0$, then $\gamma=1$ will minimize $\mathbf{x_{err}}$. This is usually the case, so to minimize mean $\|\mathbf{x_{err}}\|$, $\gamma=1$, corresponding to acceleration/velocity estimation, is the best choice. However, the worst case for $\|\mathbf{x_{err}}\|$ is $\|\mathbf{\Phi_3}\| + \gamma\,\|\mathbf{\Phi_2}\|$. So to minimize worst case error, $\gamma=0$, corresponding to velocity-only estimation, is the proper choice.

Selecting γ is a tradeoff between minimizing average and worst-case extrapolation error. Although instances of worst-case error may occur very rarely, worst-case error is very important for the PFS. Every single cutting error removes bone material that cannot be put back. Larger error instances create larger gashes in the finished surface. This is important to remember when selecting γ.

We selected γ by trial and error. The PFS software system was used to record actual use of the tool. With this stored data, we could change γ and quickly replay the tool motion on the screen, overlaid with the extrapolated position and its effect on predicted extension. Using this method, a value of $\gamma = 0.5$ was chosen. Further investigation could yield a more optimal value. The value of γ could even be automatically customized per surgeon.

7.1.2 Tool position snap-to-surface

The snap-to-surface routine works along with the previous extrapolation step to reflect the fact that the tool stays in contact with the workpiece as it travels. The extrapolation step may result in a predicted tool position that lies above or below the surface. (Figure

7-2) Snap-to-surface adjusts the tool position so that it rests on the workpiece surface. (Figure 7-3)

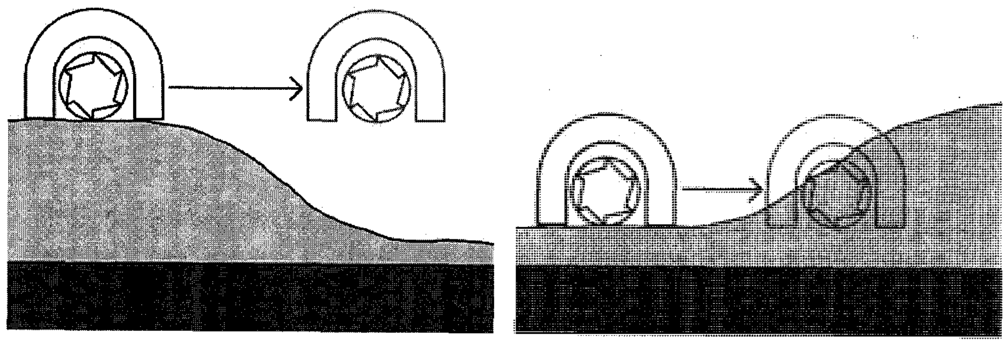


Figure 7-2: Extrapolated position may lie above or below workpiece surface.

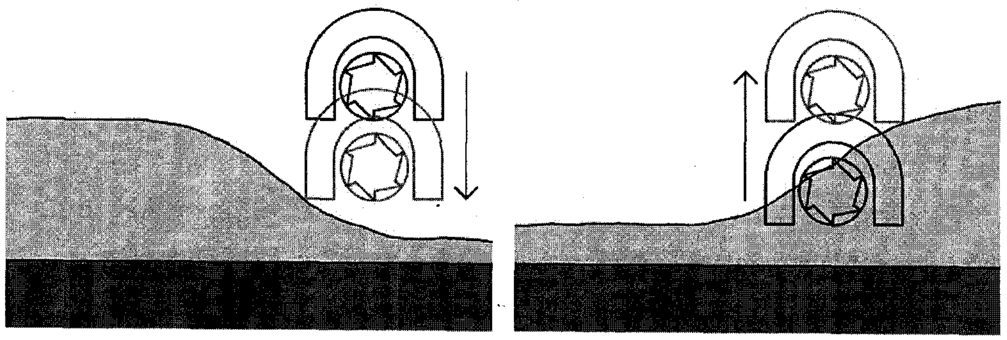


Figure 7-3. Tool position adjusted to rest on the workpiece surface.

Snap-to-surface is also applied to the current position of the tool, before it is used to calculate allowable extension. This ensures that the calculated extension is safe even if the user moves the tool towards the bone.

There is a lot of freedom in how snap-to-surface can be realized. Since the tool motion is human-controlled, it defies simple characterization, so no snap-to-surface implementation will always be right. We have attempted to develop a snap-to-surface that is the most accurate most of the time, but that is not the only way to judge snap-to-surface implementations. Another way to consider implementations is on a scale from conservative to aggressive. A more conservative method may predict tool positions that are less accurate than ours, yet may be desirable because it causes less overcutting and yields more accurate final surfaces.

Tool Orientation in Snap-to-Surface

The first question for implementing snap-to-surface is how the tool orientation is affected by the interaction with the workpiece. One might expect that by pressing the tool against the bone, the user will cause the guard to rest flat on the workpiece. To implement this, the computer needs to identify three points of contact between the workpiece and the guard, and match them up. Alternately, two points of contact, one on each side of the guard, might be used in a hybrid approach that assumes the tool rotates only about its long axis, causing both sides of the guard to rest on the workpiece.

We implemented both the former, then the latter of these strategies, but settled on a third model: that the tool does not naturally overturn to rest flat on the surface, but rather the user simply translates the tool across the surface. This was found to more accurately reflect the motion of the tool in practice. In fact, although it is desirable for efficient cutting to lay the guard flat to the workpiece, when visibility is limited it is often difficult to do so even intentionally. The cross-sectional view on the computer display is designed to help the user orient the tool flat to the workpiece, and I have often used it for this purpose.

Thus we assume that no overturning effects are present in the interaction between the guard and the workpiece, so the orientation of predicted tool positions is based entirely on the current tool orientation and angular velocity. The tool is snapped to a single point on the surface by pure translation.

Tool Translation for Snap-to-Surface

The next question is what point(s) on the workpiece surface to match the guard to. An obvious choice might be to find the point closest to the guard. We have chosen instead to translate the guard directly towards the target shape to the first point of contact. If the extrapolated guard rests inside the workpiece, we translate it directly away from the target shape until it clears the workpiece. Figure 7-4 illustrates the difference between translating toward the closest point and translating towards the target shape. In this case, translating toward the target shape puts the predicted position closer to the target shape, which would result in less blade extension.

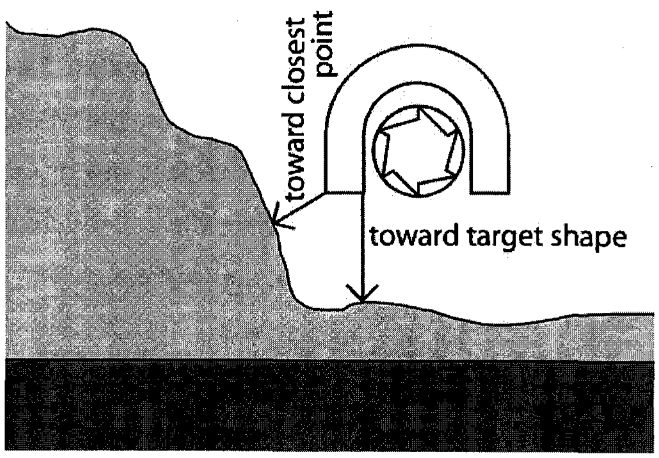


Figure 7-4 Translating to closest point vs translating towards target shape.

Adjustment of Current Position using Snap-to-Surface

In addition to predicting future positions, the algorithm examines how far the blade may extend based on the tool's current position. Adjusting the current tool position with snap-to-surface provides a safer estimate of allowable blade extension than using the current position directly would.

The naïve approach to using the current position would be to calculate how far the blade can extend directly from the tool's current position. However, if the tool is in the air instead of resting on the workpiece surface, extending the blade right to the target shape is dangerous. Because the guard is not resting on the workpiece surface, the user can quickly move the tool closer to the target surface and cause overcutting. With a moderate tool speed of 0.1m/s, and an Optotrak update rate of 0.012s, this overcut could be 1.2mm before the computer were even aware of it.

A safer way to incorporate the current position into the calculation of blade extension is to first modify the current position with snap-to-surface, so that the guard rests on the surface. This results in an allowable blade extension that is safe even if the user pushes the tool toward the workpiece surface, because once the guard rests on the workpiece surface, it will restrict how far the blade can penetrate.

The usual snap-to-surface routine adjusts both positions that are above the workpiece surface and those below it. For the current position, the tool should theoretically be on or above the workpiece surface, but tracking error, for the current frame or during earlier updates of the workpiece model, can cause the current tool position to appear to be inside the workpiece. In this case, the question is whether to believe the current position reading, or to trust the workpiece model and adjust the current position to rest on it. We have chosen to trust the current position, because it is a more direct measurement of position than inferring a position based on the model. Further, use of the current position instead of the workpiece surface results in a shorter blade extension, which is less likely to overcut.

Therefore, snap-to-surface should adjust the current position if it lies above the target shape, but not if it lies below.

Adjustment of Future Positions Based on Current Guard Penetration

The fact that in practice the current tool position reading can lie below the workpiece surface has another important consequence. The amount that the guard appears to penetrate the workpiece surface is usually similar from one timestep to the next. To improve the accuracy of predicted future tool positions, we adjust them to penetrate the workpiece surface by the same amount as the current position. We refer to the penetration distance as "guard penetration".

Evaluation of Snap-to-Surface Options as Conservative or Aggressive

Of course, the actual tool motion depends on complex human factors, and snap-to-surface is only an approximation. Although I expect the options chosen above to give the most accurate predictions most of the time, there will certainly be timesteps when other snap-to-surface policies described above are more accurate. Rather than evaluating snap-to-surface options in terms of accuracy of predicted tool position, it can be useful to consider the predicted blade extension, which directly determines cutting accuracy.

In this light, the policies can be seen on a spectrum from conservative to aggressive (Figure 7-5). Matching the surface with three points puts the guard closest to the target shape, which limits blade extension, cutting conservatively. At the other extreme, translating to the closest point on the target shape puts the guard furthest from the target shape and allows the blade to extend the furthest, cutting aggressively. In situations when a conservative approach is correct, it avoids overcutting where a more aggressive approach would overcut. When an aggressive approach is correct, it allows faster material removal where a more conservative approach would retract the blade more than necessary. Conservative approaches cut more accurately, but aggressive approaches cut more efficiently.

Tool orientation	Match 3 points	Match 2 points	Match 1 point
Tool translation	Translate to target shape		Translate to closest point
Adjustment of current position	Use snap-to-surface for current position		Don't use snap-to-surface for current position
Adjustment of future positions	Adjust for guard penetration		Don't adjust for guard penetration
	← conservative (accurate)		aggressive (efficient) →

Figure 7-5: Spectrum of snap-to-surface policies, from conservative to aggressive. The policies we chose are underlined.

A snap-to-surface policy can then be chosen based on which gives the desired mix of accuracy and efficiency, rather than which most frequently gives the best predictions of future position. For instance, if very high accuracy is desired, matching the surface with three points may be used even though it usually underestimates how far the blade may extend.

Implementation of Snap-to-Surface

The goal of snap-to-surface is to translate the tool position directly towards or away from the target shape to the point where the PFS guard contacts but does not interfere with the workpiece surface. We implemented snap-to-surface by adjusting the tool position along one of the heightvectors emanating from the target shape. This relies on the approximation that the heightvectors are roughly perpendicular to the target shape.

To adjust the position of the guard so that it rests on the surface, we must find the heightvector **v** along which the distance *d* from the PFS guard to the workpiece surface is the smallest. (Figure 7-6) If the guard violates the workpiece, we find the heightvector for which *d* is most negative. Translating the tool position along **v** by distance *d* makes the guard just touch the workpiece surface along **v**. Since the guard is further from the workpiece along all other heightvectors, this translated position should not violate the workpiece surface.

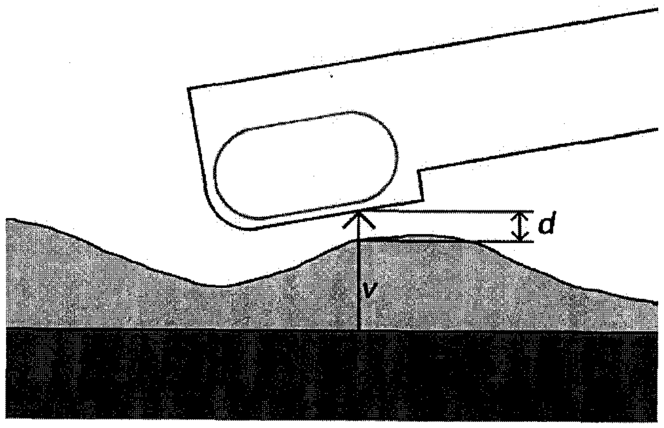


Figure 7-6: Heightvector *v* intersects guard with distance *d* from guard to workpiece. To implement snap-to-surface, the tool position will be adjusted along the heightvector for which *d* is minimum.

If the target shape is flat and the heightvectors are perpendicular to the target shape, this algorithm performs as expected. With a curved target shape, the above algorithm might result in a tool position that slightly violates the workpiece. However, the effect of this error is conservative: Since the tool position is inside the workpiece, the calculated blade extension may be slightly less than it should be.

Otherwise, the algorithm handles curved target shapes gracefully. For instance, it rounds the transitions over corners because the heightvector directions transition smoothly over corners.

All that remains here is to describe the calculation of the distance along a heightvector from the workpiece surface to the PFS guard. This distance is found by calculating the distance from the target shape to the guard, and subtracting the heightvecctor height, which is the distance from the target shape to the workpiece surface. We model the guard surface as simple regions in two planes joined by a quarter cylinder. (Figure 7-7) First we will find where the heightvector hits one plane, then check if that intersection point lies in the planar regions of the guard surface. We do the same for the other plane and then the quarter cylinder.

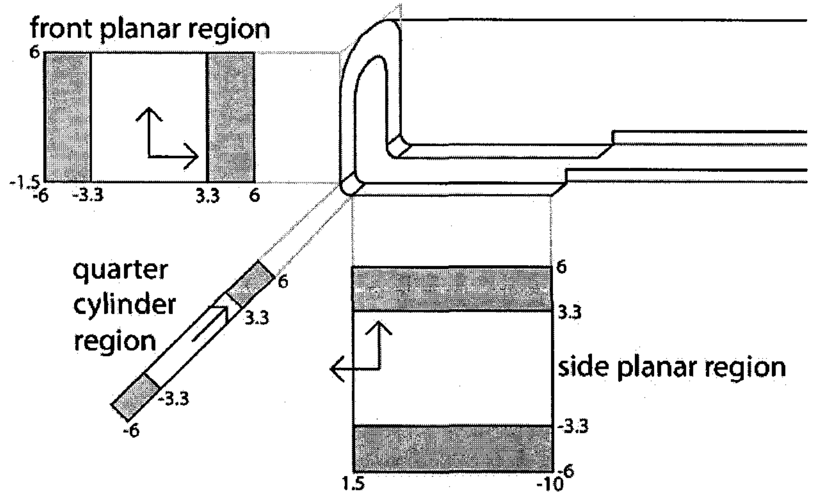


Figure 7-7: Guard is modeled as regions in two planes and a quarter cylinder.

Let a given heightvector be described by the symbols

- $\mathbf{b}_{\text{vec}}$: The position of the heightvector base on the target shape
- $\mathbf{d}_{\text{vec}}$: The unit direction of the heightvector

Let a given plane of the guard surface (i.e. side plane or tip plane), be defined by:

- $\mathbf{b}_{\text{pl}}$: The position of the plane origin
- $\mathbf{n}_{\text{pl}}$: The unit normal vector of the plane

- $\mathbf{p_1}$ and $\mathbf{p_2}$: orthonormal bases for the plane.

the heightvector intersects the plane at a distance

$$dist = ((\mathbf{b_{vec}} - \mathbf{b_{pl}}) \cdot \mathbf{n_{pl}}) / (\mathbf{d_{vec}} \cdot \mathbf{n_{pl}})$$

along the heightvector, and the position of the heightvector's intersection with the plane is given by:

$$\mathbf{x} = \mathbf{b_{vec}} + dist\ \mathbf{d_{vec}}$$

Next **x** is compared to the regions in the plane that represent the guard surface. The coordinates of **x** in the frame of the plane are computed by dot product with $\mathbf{p_1}$ and $\mathbf{p_2}$, and then simple comparisons can be made. For instance, for the radial plane, **x** intersects the guard surface if:

$$3.3 < | (\mathbf{x} - \mathbf{b_{pl}}) \cdot \mathbf{p_1} | < 6$$

and

$$-10 < | (\mathbf{x} - \mathbf{b_{pl}}) \cdot \mathbf{p_2} | < 1.5$$

Note that the area of the guard model must be wide enough, given the heightfield resolution, to ensure that it is hit by a representative sample of heightvectors. Otherwise the guard can "fall" into the spaces between the heightvectors.

The intersection of the heightvector with the quarter-cylinder section of the guard is similar to the heightvector-capsule intersection calculation described in Chapter 6. In brief, with the cylinder described by:

- $\mathbf{d_{cyl}}$: direction of cylinder axis
- $\mathbf{b_{cyl}}$: position of cylinder base
- r_{cyl}: radius of cylinder

then the distance *dist* along the heightvector to its intersection with the cylinder is found by:

$\mathbf{n_2} = \mathbf{d_{hv}} \times \mathbf{d_{cyl}} / \| \mathbf{d_{hv}} \times \mathbf{d_{cyl}} \|$

$m = \| \mathbf{d_{hv}} \times \mathbf{d_{cyl}} \|$

$\mathbf{v} = \mathbf{b_{cyl}} - \mathbf{b_{vec}}$

$d = | \mathbf{v} \cdot \mathbf{n_2} |$

$t = (\mathbf{v} \times \mathbf{d_{cyl}}) \cdot \mathbf{n_2} / m$

$s = \sqrt{{r_{cyl}}^2 - d^2} \; / m$

$dist = t - s$

additionally,

$a = -\mathbf{v} \cdot \mathbf{d_{cyl}} + (t\text{-}s)(\mathbf{d_{vec}} \cdot \mathbf{d_{cyl}})$

is the position of the intersection along the cylinder axis. The heightvector intersects the guard surface on the quarter-cylinder if $3.3<|a|<6$. If so, *dist* is the distance along the heightvector to the intersection.

The above calculations are repeated for all heightvectors on the target shape. The distance along each heightvector from the workpiece surface to the guard is computed as *dist* minus the heightvector height. The heightvector with the shortest distance from the workpiece surface to the guard is selected, and the snap-to-surface is achieved by translating the tool along that heightvector.

7.2 Allowable Blade Extension

At each predicted position of the tool, the algorithm must determine how far the blade can be extended. To determine this, the algorithm assumes that the blade moves freely through bone and that motion of the blade does not affect the tool position. The allowable blade extension in a given direction is determined by how far the blade can translate in that direction from the unextended "neutral" position before it intersects the target shape.

The allowable blade extension is not just a single number, it depends on the direction in which the blade extends. Allowable extension is also not linear – we cannot simply measure the allowable extension in the axial and radial directions and compose the result. Figure 7-8 illustrates a situation where composing the axial and radial allowable extensions would result in overcutting. To represent allowable extension throughout the blade range of motion, we discretize the range of motion into 5 "candidate directions" (Figure 7-9) and measure the allowable extension in each. Later we will choose one candidate direction in which to extend the blade.

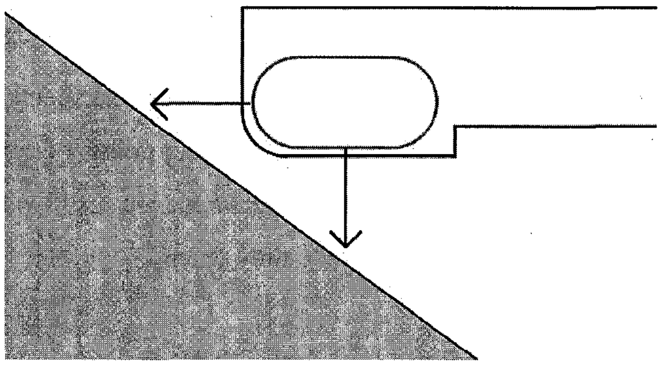


Figure 7-8: Allowable extension is nonlinear. In this example, the allowable extension in the diagonal direction is less than the sum of the allowable axial and radial extensions.

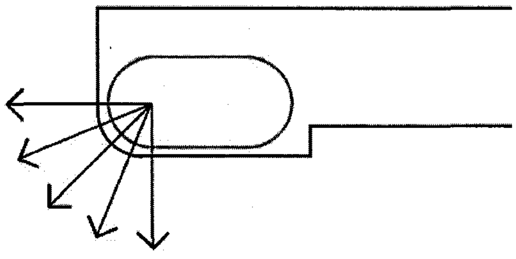


Figure 7-9: Candidate Extension Directions. The allowable extension is calculated in each of the five candidate directions.

Implementation: To calculate the allowable extension, a dedicated algorithm is used which calculates how far a capsule may be translated along a given direction before it intersects a triangle. As with the heightvector-capsule intersection code, the shape of the blade is implicitly hard-coded into this algorithm to achieve fast computation. This algorithm is used to test the blade against each triangle in the target surface, and the minimum allowable extension among all triangles on the target surface is found. Finally, the entire procedure is repeated for each candidate extension direction, so that an allowable blade extension that does not intersect the target shape is found for each candidate direction.

Note that this algorithm for comparing the capsule-shaped blade to a triangle on the target shape does not just calculate whether the capsule and triangle intersect. It calculates the first location where the capsule intersects the triangle as the capsule is translated along the extension direction. We refer to the intersecting point or region as the "first point of contact" between the capsule and the triangle, and the amount of translation as the "distance to first contact". If the capsule intersects the triangle in its initial position, then the distance to first contact will be negative. Note also that in the special case where the capsule axis is parallel to the plane, there may be many first points of contact. Any one of these points may be used for calculation, since the important result is the distance to first contact, which determines how far the blade may extend.

The algorithm to calculate the distance to first contact between a capsule and a triangle is composed of several more basic geometric computations. These are laid out in the flowchart in Figure 7-10.

First **(Step 1)**, the algorithm checks if the capsule first contacts the interior of the triangle. If not, it calculates the capsule's first contact with each triangle edge and reports the closest distance to first contact. To find the first contact with an edge, the first calculation **(Step 2)** compares the inifinite line extending from the edge with the infinite cylinder extending from the capsule. Subsequent calculations **(Step 3-5)** narrow down whether the first contact is on the finite extents of the capsule and edge.

Definitions for the Capsule / Triangle First-Point-of-Contact Algorithm.
We will use the following definitions for the capsule / triangle algorithm.

Let the capsule be defined by:

- $\mathbf{x_{cap}}$ (c_{pos}): Capsule position. Position of the center of one sphere on the capsule

- $\mathbf{d}_{\mathbf{cap}}$ (c_{axis}): Capsule axis. Unit vector from c_{pos} to the center of the other sphere.
- l_{cap} (c_{len}): Capsule length (scalar). Length between the two spheres of the capsule
- r_{cap} (c_{rad}): Capsule radius (scalar).

Let the direction that the capsule is translated be defined by:

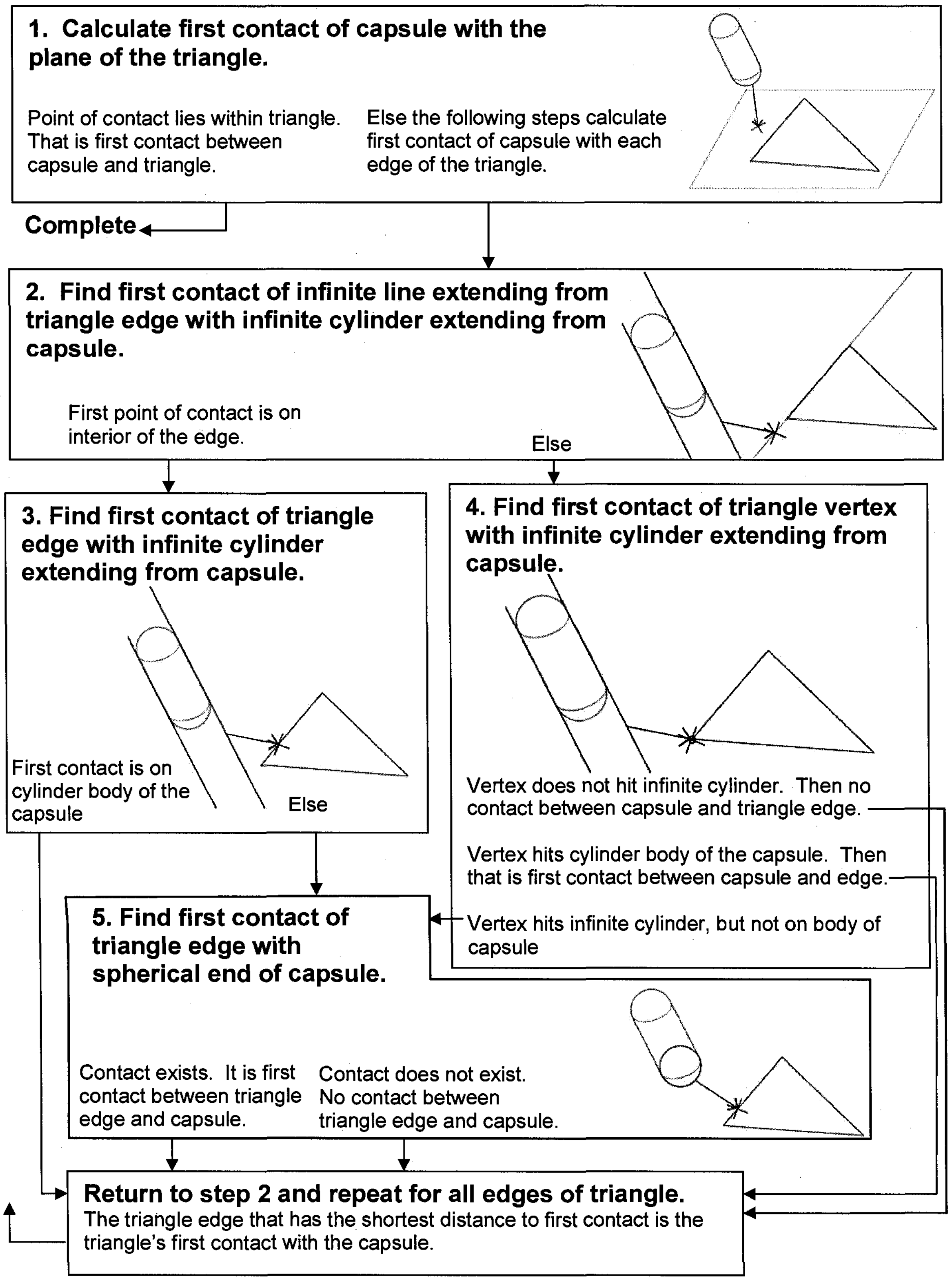
- **d**: a unit direction vector

Let the triangle be defined by:

- $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$: the vertices of the triangle

For convenience, we also define:

- **n**: let **n** be normal to the triangle, with a direction facing outside the target shape. Note that if $\mathbf{n} \cdot \mathbf{d} > 0$, the direction of extension points away from the target shape and we should not perform the test on this triangle. Two cases where this may happen are if the tool is pointed away from the workpiece, or if the triangle is on the far side of the workpiece from where the tool is operating.



1. Calculate first contact of capsule with the plane of the triangle.

Point of contact lies within triangle. That is first contact between capsule and triangle.

Else the following steps calculate first contact of capsule with each edge of the triangle.

Complete

2. Find first contact of infinite line extending from triangle edge with infinite cylinder extending from capsule.

First point of contact is on interior of the edge.

Else

3. Find first contact of triangle edge with infinite cylinder extending from capsule.

First contact is on cylinder body of the capsule

Else

4. Find first contact of triangle vertex with infinite cylinder extending from capsule.

Vertex does not hit infinite cylinder. Then no contact between capsule and triangle edge.

Vertex hits cylinder body of the capsule. Then that is first contact between capsule and edge.

Vertex hits infinite cylinder, but not on body of capsule

5. Find first contact of triangle edge with spherical end of capsule.

Contact exists. It is first contact between triangle edge and capsule.

Contact does not exist. No contact between triangle edge and capsule.

Return to step 2 and repeat for all edges of triangle.
The triangle edge that has the shortest distance to first contact is the triangle's first contact with the capsule.

Figure 7-10: Flowchart of algorithm to calculate how far capsule-shaped blade may extend before it intersects a triangle of the target shape.

Step 1. Calculate capsule's first point of contact with the plane of the triangle. The first step in calculating the capsule's first contact with the triangle is to calculate its first point of contact with the plane that the triangle lies in. If the first contact with the plane lies within the triangle, then it is the capsule's first contact with the triangle, since the triangle is a subset of the plane. In this case, the algorithm returns the first contact with the triangle. Otherwise, the capsule's first contact with the triangle must be on one of the triangle edges, because the capsule's intersection with the plane will grow continuously and hit an edge before it enters the triangle interior. In this case, the algorithm proceeds to steps 2 through 5, where it will calculate the capsule's first contact with each of the triangle edges, and then choose the shortest distance to contact.

The calculation of the capsule's first contact with the plane begins with two special cases.

```
special case: $\mathbf{d} \cdot \mathbf{n} = 0$                    (d is parallel to the plane)
Continue to Step 2
```

In this case, the extension direction **d** is parallel to the plane. If the capsule does make contact with the triangle, its first contact will be on an edge. We proceed to Step 2 to check each of the edges.

```
special case: $\mathbf{d}_{cap} \cdot \mathbf{n} = 0$              (capsule is parallel to the plane)
First contact of capsule with plane is a line.
If any part of line intersects triangle, return contact.
Else continue to Step 2
```

If the special cases do not apply, then we can assume that the capsule's first point of contact with the plane will therefore be on one of the spherical ends of the capsule. We set s to be the centerpoint of the spherical end which will hit the plane first (Figure 7-11), and find that sphere's first point of contact with the plane:

```
if $\mathbf{d}_{cap} \cdot \mathbf{n} < 0$
```

```
        $\mathbf{s} \leftarrow \mathbf{x}_{cap}$
else
        $\mathbf{s} \leftarrow \mathbf{x}_{cap} + l_{cap}\,\mathbf{d}_{cap}$

$\mathbf{p}_1 = \mathbf{s} - r_{cap}\,\mathbf{n}$                    (point on sphere which will first contact plane)
$dist = (\mathbf{p}_1 - \mathbf{t}_1) \cdot \mathbf{n} / \mathbf{d} \cdot \mathbf{n}$                    (distance to first contact)
$\mathbf{p}_2 = \mathbf{p}_1 + dist\,\mathbf{d}$                    (point of contact on plane)
```

$\mathbf{p}_2$ is the point of contact on the plane. Next we determine whether $\mathbf{p}_2$ lies in the triangle:

```
for i=1,2,3
        if $(\mathbf{p}_2 - \mathbf{t}_i) \cdot ((\mathbf{t}_{i+1} - \mathbf{t}_i) \times \mathbf{n}) > 0$          (Check which side of edge $\mathbf{p}_2$ is on)
                $p_2$ is not in triangle: continue to step 2
        end
end
$p_2$ is in triangle. Return $dist$ as distance to first contact.
```

To check which side of the edge $\mathbf{p}_2$ lies in, we have assumed that the triangle vertices are wound counterclockwise and the **n** points away from the workpiece. If the capsule's first contact does not lie in the interior of the triangle, the algorithm uses Steps 2-5 to calculate the distance to first contact of the capsule with each triangle edge.

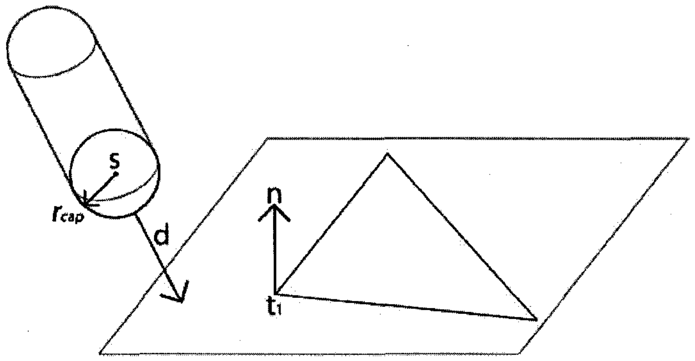


Figure 7-11: Calculating the capsule's first contact with the plane of the triangle is reduced to finding the first contact between the sphere centered at s with radius c_{rad} and the plane.

Step 2. Calculate first contact of infinite line extending from the triangle edge with the infinite cylinder extending from the capsule.

Step 2 is the first step in calculating the capsule's first contact with an edge of the triangle. In this step we find the first point of contact between the infinite line extending from the line segment (edge) and the infinite cylinder that extends from the capsule. In particular, we find whether that first point of contact is on the actual segment, or elsewhere on the line. If the point of first contact is on the actual segment, we continue to Step 3 to check where the first contact is on the cylinder. Otherwise, the cylinder's first contact with the segment will be on the closer endpoint of the segment, and we continue to Step 4 to compare the cylinder with the segment endpoint.

Let the line segment (triangle edge) be defined by:

- $\mathbf{x}_{\mathbf{seg}}$: an endpoint of the segment.
- $\mathbf{d}_{\mathbf{seg}}$: a unit direction vector.
- l_{seg}: the length of the segment.

In the following calculation, we assume that $\mathbf{d}$, $\mathbf{d}_{\mathbf{cap}}$, and $\mathbf{d}_{\mathbf{seg}}$ are linearly independent. There are three degenerate cases where this is not true:

If the cylinder is parallel to the extension direction, any potential first point of contact between the segment and the capsule will be on the spherical end of the capsule:

special case: $\mathbf{d}_{\mathbf{cap}} \parallel \mathbf{d}$
Go to step 5

If the segment is parallel to the extension direction, any potential first point of contact will always be on the endpoint of the segment:

special case: $\mathbf{d}_{\mathbf{seg}} \parallel \mathbf{d}$
Go to step 4.

The third special case is when the segment is parallel to the capsule. If the infinite cylinder does contact the infinite line, the entire line will contact the cylinder at once. Some calculations are needed then to determine whether the segment will contact the cylinder body of the capsule.

```
special case: $\mathbf{d}_{cap} \parallel \mathbf{d}_{seg}$
$\mathbf{d}_2 = \text{unit}(\mathbf{d} - \mathbf{d} \cdot \mathbf{d}_{cap})$                (projection of d perpendicular to $x_{cap}$)
$\mathbf{h} = (\mathbf{d}_2 \times \mathbf{d}_{cap}) / \| \mathbf{d}_2 \times \mathbf{d}_{cap} \|$
$dist = (\mathbf{x}_{cap} - \mathbf{x}_{seg}) \cdot \mathbf{d}_2 - \sqrt{r_{cap}^{\ 2} - ((\mathbf{x}_{cap} - \mathbf{x}_{seg}) \cdot \mathbf{h})^2} \ / \ \mathbf{d}_2 \cdot \mathbf{d}$
$a = ( (\mathbf{x}_{cap} + dist\ \mathbf{d}) - \mathbf{x}_{seg} ) \cdot \mathbf{d}_{seg}$
if $(a, a+l_{cap})$ intersects $(0, l_{seg})$
    Capsule's first contact lies on segment interior.
    Return dist as distance to first contact.
else
    Continue to Step 5.
```

If the special cases do not hold, then $\mathbf{d}_{cap}$, $\mathbf{d}$, and $\mathbf{d}_{seg}$ are all linearly independent. The viewpoint used in Figure 7-12 is useful in visualizing the first contact between the infinite line and infinite cylinder. Here, c_{axis} points into the page. d and s_{dir} are neither in the plane of the page nor perpendicular to it.

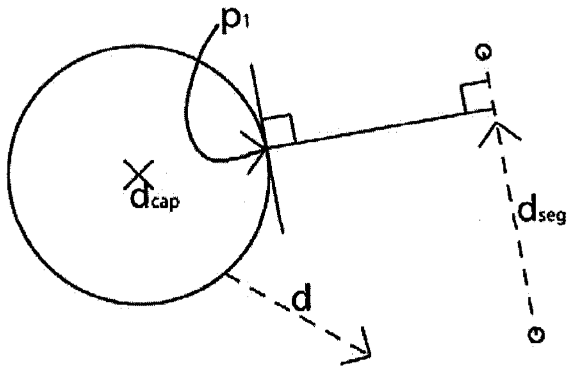


Figure 7-12: Finding the first contact between the capsule (normal to page) and the segment (arbitrary direction. Point of first contact p_1 lies on part of capsule that is tangent to the direction s_{dir}.

It can be seen in Figure 7-12 that $\mathbf{p}_1$, the line's first point of contact on the cylinder, will be located where the projection of $\mathbf{d}_{\mathbf{seg}}$ onto the page is tangent to the cylinder's projection onto the page.

To find $\mathbf{p}_1$, we let

$$\boxed{\mathbf{r} = \mathbf{d}_{\mathbf{cap}} \times \mathbf{d}_{\mathbf{seg}} / \| \mathbf{d}_{\mathbf{cap}} \times \mathbf{d}_{\mathbf{seg}} \|, \text{ with the sign chosen so that } \mathbf{r} \cdot \mathbf{d} > 0.}$$

Since $\mathbf{r}$ is perpendicular to $\mathbf{d}_{\mathbf{seg}}$, it is a radius vector of the cylinder that points toward $\mathbf{p}_1$.

Now that $\mathbf{r}$ tells us where to find the first point of contact, we can calculate its location on both the line extending from the segment, and the cylinder extending from the capsule. Let λ be the distance that the cylinder is translated along d. Let β be the distance along the line where the contact occurs, and let γ be the distance along the cylinder where the contact occurs. Thus the contact occurs on the line at $\mathbf{x}_{\mathbf{seg}} + \beta\, \mathbf{d}_{\mathbf{seg}}$. The contact occurs on the cylinder at $\mathbf{x}_{\mathbf{cap}} + \gamma\, \mathbf{d}_{\mathbf{cap}} + r_{cap}\, \mathbf{r}$, when the cylinder has been translated by $\lambda\, \mathbf{d}$. This gives:

$$\mathbf{x}_{\mathbf{cap}} + \gamma\, \mathbf{d}_{\mathbf{cap}} + r_{cap}\, \mathbf{r} + \lambda\, \mathbf{d} = \mathbf{x}_{\mathbf{seg}} + \beta\, \mathbf{d}_{\mathbf{seg}}$$

This can be rewritten as

$$[\mathbf{d} \quad -\mathbf{d}_{\mathbf{seg}} \quad \mathbf{d}_{\mathbf{cap}}] \begin{bmatrix} \lambda \\ \beta \\ \gamma \end{bmatrix} = \mathbf{x}_{\mathbf{seg}} - \mathbf{x}_{\mathbf{cap}} - r_{cap}\, \mathbf{r}$$

and solved by matrix inversion. The inverse exists, since $\mathbf{d}$, $\mathbf{d}_{\mathbf{seg}}$, and $\mathbf{d}_{\mathbf{cap}}$ are independent. The algorithm continues:

```
$\begin{bmatrix}\lambda\\ \beta\\ \gamma\end{bmatrix} = [\mathbf{d}\ \ -\mathbf{d}_{seg}\ \ \mathbf{d}_{cap}]^{-1}\,(\mathbf{x}_{seg} - \mathbf{x}_{cap} - r_{cap}\,\mathbf{r})$
if $0<\beta<l_{seg}$
        The first contact occurs on the interior of the segment.
        Continue to Step 3.
else
        If there is a first contact between segment and cylinder, it is on an edge vertex.
        Continue to Step 4.
```

Step 3. Find first contact of triangle edge with infinite cylinder extending from the capsule.

At this point, Step 2 has determined that the infinite cylinder makes first contact with the triangle edge on the edge's interior. Next we test whether that first contact occurs on the cylinder body of the capsule, or elsewhere on the infinite cylinder. If it occurs on the body of the capsule, then this represents the first contact between the capsule and triangle edge, and the algorithm returns to Step 2 to consider the remaining edges of the triangle. If the first contact with the triangle edge occurs elsewhere on the infinite cylinder, then the only possible contact between the capsule and the triangle edge is on the spherical capsule end that is nearest the edge's first contact with the infinite cylinder. (This is proven by a theorem in the discussion of Step 5.) The algorithm then goes to Step 5.

Where on the cylinder the first contact occurs is determined by γ, which was solved for in Step 2:

```
if $0 < \gamma < l_{cap}$
        The first contact occurs on the cylinder body of the capsule.
        The distance to first contact is $\lambda$.
        Return to Step 2 to check the other triangle edges.
if $\gamma > l_{cap}$
        The spherical end-cap centered at $\mathbf{s} = \mathbf{x}_{cap} + l_{cap}\,\mathbf{d}_{cap}$ may contact the edge.
        Continue to Step 5
if $\gamma < 0$
        The spherical end-cap centered at $\mathbf{s} = \mathbf{x}_{\mathbf{cap}}$ may contact the edge.
        Continue to Step 5
```

Step 4. Find first contact of triangle vertex with infinite cylinder extending from capsule.

At the start of this step, Step 2 has determined that the first contact of the infinite cylinder with the triangle edge is at one of the edge's endpoints. The goal of this step is to determine where on the infinite cylinder the first contact with the endpoint takes place.

Let **p** represent the endpoint position.

To perform the actual calculation necessary for Step 4, finding the first contact between the endpoint and the infinite cylinder, the problem can be rephrased to resemble the heightvector-capsule intersection discussed in Chapter 6. Rather than translating the capsule toward the point along the vector d, consider translating the point toward the capsule along -d. The ray made by the translating point is analogous to the heightvector. Substituting the point location **p** and the direction **–d** for the heightvector, we can rewrite the algorithm from Chapter 6 as:

```
$\mathbf{n} = \mathbf{-d} \times \mathbf{d_{cap}} / \| \mathbf{-d} \times \mathbf{d_{cap}} \|$
$m = \| \mathbf{-d} \times \mathbf{d_{cap}} \|$
$t = ((\mathbf{x_{cap}} - \mathbf{p}) \times \mathbf{d_{cap}}) \cdot \mathbf{n} / m$
if $((\mathbf{x_{cap}} - \mathbf{p}) \cdot \mathbf{n}) > r_{cap}$
        The endpoint does not contact the capsule.
        Return to Step 2 to test other triangle edges.

$s = \sqrt{r_{cap}^{2} - ((\mathbf{x_{cap}} - \mathbf{p}) \cdot \mathbf{n})^{2}} / m$
```

The first contact of the endpoint with the infinite cylinder is at *t-s*. Next we calculate where on the cylinder the contact occurs, to determine if the contact is on the cylinder body of the capsule:

```
$a = (\mathbf{p} - \mathbf{x_{cap}}) \cdot \mathbf{d_{cap}} + (t\text{-}s)(-\mathbf{d} \cdot \mathbf{d_{cap}})$
if $0<a<l_{cap}$
        The first contact of the edge with the capsule is on the endpoint.
        The distance to first contact is $t\text{-}s$.
        Return to Step 2 to test the other triangle edges.
if $a>l_{cap}$
        The spherical end-cap centered at $\mathbf{s} = \mathbf{x_{cap}} + l_{cap}\,\mathbf{d_{cap}}$ may contact the edge.
        Continue to Step 5
if $a < 0$
        The spherical end-cap centered at $\mathbf{s} = \mathbf{x_{cap}}$ may contact the edge.
        Continue to Step 5
```

Step 5. Find first contact of triangle edge with spherical end of capsule.
At the start of Step 5, the algorithm has deduced that triangle edge's first point of contact with the infinite cylinder is beyond the cylinder body of the capsule. The final test for contact between the capsule and the triangle is to check for contact between the edge and the spherical capsule end-cap which is nearest to the edge's first contact with the infinite cylinder. The following theorem shows that this end of the capsule is the only remaining possible location for first contact between the segment and capsule.

Theorem. Let the sets S and C represent the sets of points in the segment and capsule surface, respectively. Let C_∞ represent the infinite cylinder surface extending from C. Let C_{end} be a hemispherical end of C, including the circular rim which lies on C_∞. Note that the surface C_{end} divides C_∞ into two halves. Let C_{∞_end} be the half which does not include C.

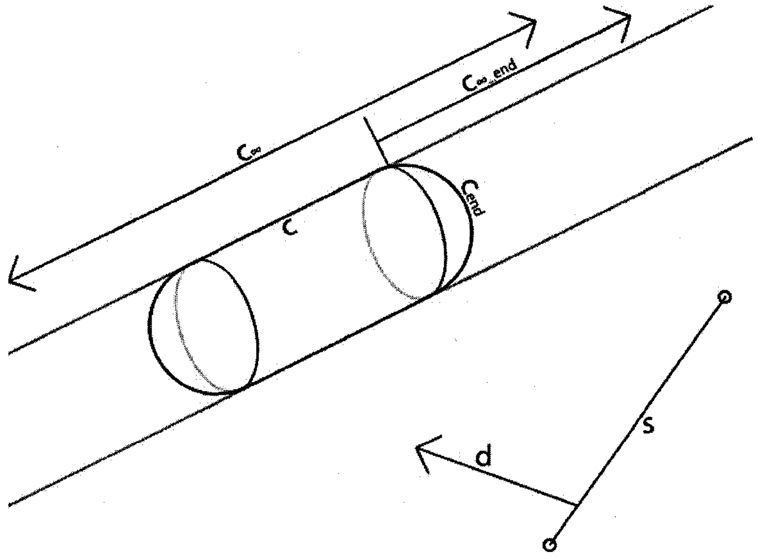


Figure 7-13: Sets c and s are the segment and capsule surface. If first contact of s with c_∞ lies on c_{∞_end}, then first contact of s with c lies on c_{end}.

Let **d** be a unit direction vector, and ρ be the distance along **d** to first contact between C and S. Assume the first contact of C_∞ with S along **d** occurs at least partially on C_{∞_end}, i.e.

$(S - \rho\, \mathbf{d}) \cap C_{\infty\_end} \neq \emptyset$ and $\{S - r\,\mathbf{d} \mid r < \rho\} \cap C_\infty = \emptyset$

Assume also that the first contact between S and C along **d** occurs at distance τ, i.e.

$(S - \tau\, \mathbf{d}) \cap C \neq \emptyset$ and $\{S - r\,\mathbf{d} \mid r < \tau\} \cap C = \emptyset$

Then at least part of the first contact of S with c lies on C_{end}.

Note that the use of set notation allow for the possibility of multiple first contacts if the segment is parallel to the cylinder axis.

Proof. Assume the theorem does not hold. Let $\mathbf{p_1}$ in $(S - \tau\, \mathbf{d}) \cap C$ but not_in C_{end}, and $\mathbf{p_2}$ in $(S - \rho\, \mathbf{d}) \cap C_{\infty\_end}$ be given. Consider the set $T = \{S - r\,\mathbf{d} \mid r < \tau\}$, which is convex. Consider the line Q between $\mathbf{p_1}$ and $\mathbf{p_2}$. Since T is convex and $\mathbf{p_1}$, $\mathbf{p_2}$ are in T, Q is a subset of T. Further, Q must contain a member of C_{end}: $\mathbf{p_1}$ and $\mathbf{p_2}$ are both in the infinite cylinder, and on opposite sides of C_{end}. However, this implies that some element of C_{end}

makes contact with s before or simultaneously with $\mathbf{p}_1$, and we reach a contradiction. QED

To calculate the first contact between the segment and the spherical end of the capsule, the problem can once again be manipulated to be equivalent to the heightvector-capsule intersection test developed in Chapter 6. The segment/sphere question asks how far the sphere center must be translated before it gets within the distance r_{cap} of the segment. By transferring the radius r_{cap} from the sphere to the segment, the problem is transformed into a point translating toward a capsule (Figure 7-14). In implementation, we calculated the segment/sphere first contact by calling the ray/capsule intersection function with the proper change of arguments.

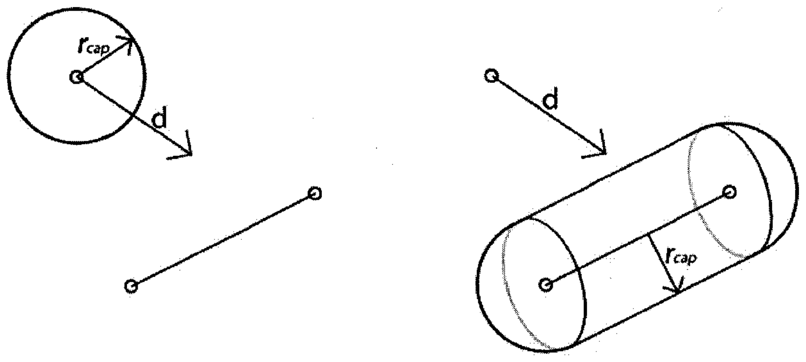


Figure 7-14: Calculating first contact of sphere with segment is equivalent to calculating vector/capsule intersection.

The algorithm is simply:

```
Call ray/capsule intersection code to calculate first contact between sphere and edge.
if intersection
        Record distance to contact
        Return to Step 2 to test the other triangle edges.
else
        This edge does not contact the capsule.
        Return to Step 2 to test the other triangle edges.
```

Once all three edges have been considered, the closest distance to first contact among all edges is reported as the distance to first contact between the capsule and the triangle.

7.3 Prediction of Multiple Timesteps

The above steps of predicting the tool position and calculating the allowable extensions are repeated for each timestep that the algorithm generates a prediction for. Presently the algorithm uses the current tool position and four future predictions. The prediction times correspond to future algorithm cycles, which are synchronized with the Optotrak and occur every 12ms.

The result of these calculations is an allowable blade extension for the current and several future timesteps, with five candidate directions considered at each timestep. The next task for the algorithm is to reduce this down to a single blade extension command to give the tool. To do so, we first apply the dynamic constraints of the blade extension motors to determine a maximum extension in each of the five candidate directions. Then we select the single extension direction that enables most efficient cutting.

7.4 Dynamic Constraint

The procedures described above result in a matrix of numbers describing allowable blade extensions: the current timestep and several future timesteps are considered, and for each timestep the allowable extension is calculated in each candidate extension direction. The algorithm must next condense this information down to a single extension direction and extension distance.

In this section we will apply the dynamic constraints of the blade extension motors to determine a single blade extension command for each candidate direction, based on present and future constraints. Then in Section 7.5, we will choose in which candidate direction to extend the blade.

In each candidate direction, the allowable extension predicted for each timestep places a constraint on how far the blade may be commanded to extend. For the prediction one step in the future this constraint is direct: the command must ensure that the blade will achieve the required position by the beginning of the coming timestep. For timesteps further out, it is indirect: the algorithm may command the blade to extend beyond predicted extensions, as long as the blade will be able to retract to the predicted amounts by the appropriate times.

The dynamic model used in this work was a constant velocity assumption. Certainly this is not ideal. In reality, there are separate axial and radial blade extension motors, each with its own dynamic characteristics. Constant velocity is not a bad approximation for the ultrasonic motor that drives axial extension, but for the DC gearmotor that drives radial extension, the acceleration phase is significant. Using a more realistic dynamic model may be a component of future work.

For a given candidate direction, the dynamic model is applied to the predicted and current allowable extensions to determine how far the blade may be commanded to extend. Let t_0, t_1, t_2, etc., spaced Δt apart, indicate the time of successive Optotrak reports, when the algorithm will run. For timestep t_0, let r indicate the allowable extension based on the current tool position, and let p_1, p_2, etc. indicate the predicted allowable extensions for timesteps t_1, t_2, etc., as computed at time t_0. The algorithm must derive an extension command c_0 which allows all extension constraints to be fulfilled. Because of computation time, the command c_0 will actually be issued some time after t_0 and before t_1, but we currently neglect that computation delay.

Prediction p_1 must be satisfied by time t_1. Since command c_0 is the only command to be issued before t_1, it must satisfy p_1 directly:

$$c_0 < p_1$$

Prediction p_2 must be satisfied by time t_2. This means that at time t_1, the blade must be positioned so that it can retract to p_2 by time t_2. With the constant velocity assumption, the distance the blade can retract in one timestep is $s\ \Delta t$, where s is the blade retraction speed. Therefore:

$c_0 < p_2 + s\,\Delta t$

And so on for the other predictions:

$c_0 < p_i + (i\text{-}1)\,s\,\Delta t$

The remaining constraint is r, the allowable extension for the current tool position. Theoretically, it shouldn't be necessary to retract based on the current position, since by the time the retraction is achieved, the tool will have moved on. However, predictions can be inaccurate, and the current tool position may provide the best estimate of how far we can safely extend the blade. Therefore, we do not allow the commanded extension to violate the allowable extension for the current position either:
$c_0 < r$

Combining these constraints lead to the formulation of c_0:
$c_0 = \min(r, p_1, p_2{+}s\,\Delta t, \ldots, p_i + (i\text{-}1)\,s\,\Delta t)$

A single extension command like c_0 is calculated for each candidate extension direction.

Application of Extension Multiplier α
The final step is to slightly reduce the calculated extension to compensate for prediction error. We multiply the calculated extensions by a factor α, which we call the *extension multiplier*. The value used in this work is $\alpha = 0.9$, and values in the range (0.7, 0.9) are typical. The extension multiplier ensures that even in the presence of minor prediction error, the tool will not overcut.

7.5 Choose Extension Direction

After the dynamically-constrained extension is computed for each of the 5 candidate directions, a single direction must be chosen in which to extend the blade. The obvious choice is to extend the blade directly toward the target shape. However, if the target shape is curved, more than one direction can simultaneously point toward it (Figure

7-15). To choose between these options, we attempt to maximize cutting efficiency. We will estimate how deeply the blade will penetrate the waste bone in each direction, and choose the maximum.

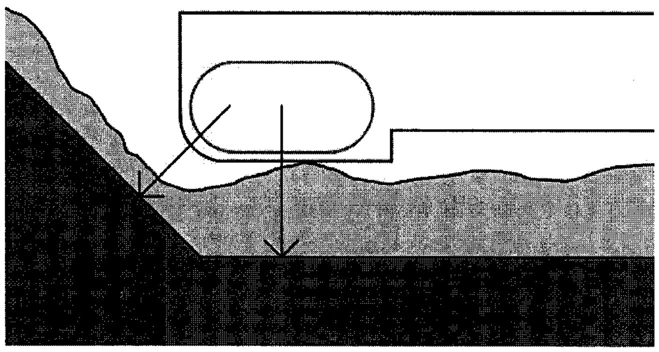


Figure 7-15: More than one direction points toward the target shape.

To estimate how much the blade will penetrate the surface with a given extension, the approximation shown in Figure 7-16 is effective and computationally efficient. As always, we assume that the guard position is unaffected by motion of the blade. Consider a candidate direction vector **dir** with extension distance *extn*. For a given point on the target shape, with normal **n**, we let *space* equal the size of the empty space between the workpiece surface and the neutral (unextended) configuration of the blade along the direction **n**. The estimated depth of blade penetration into the workpiece at that point is then given by projecting the extension onto **n** and subtracting space:

$$\textit{depth of cut} = -\textit{extn}\,(\mathbf{n} \cdot \mathbf{dir}) - \textit{space}$$

The depth of cut for a candidate direction **dir** is the maximum of this quantity over all points on the target shape. The candidate direction with the largest depth of cut is chosen as the direction to extend the blade in.

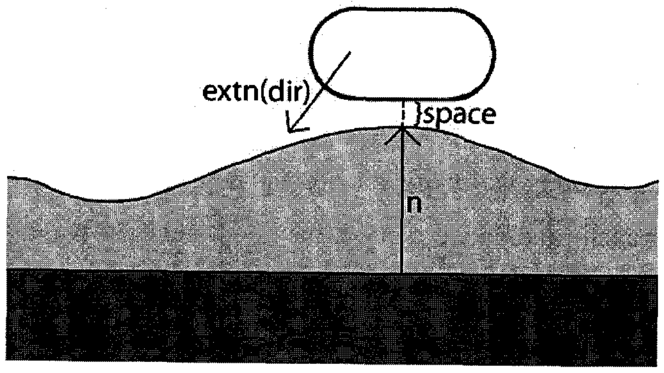


Figure 7-16: depth of cut = -extn (n · dir) – space

This calculation is efficient but only approximate. The problem is that *space* is calculated along the normal vector, but the blade extends to a different position. So *space* is not calculated beneath the position that a point of interest on the blade extends to. Figure 7-17 illustrates a worst case scenario. The blade could remove significantly more material by extending axially instead of radially, but this is not detected by the formula for *depth of cut*.

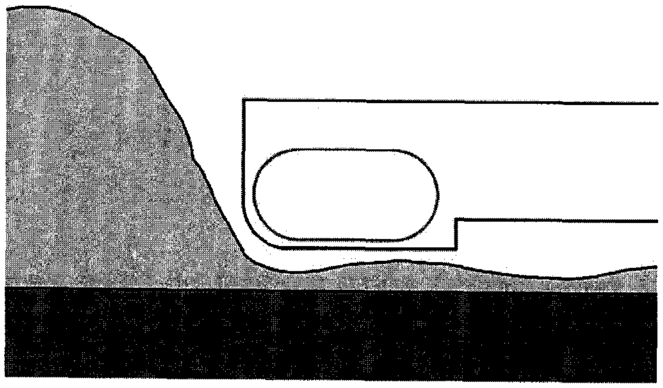


Figure 7-17: blade could remove more material by extending tangent to target surface.

Note that in general, if the target shape is locally planar, the depth of cut calculation essentially causes the blade to extend directly towards the target shape. In the formula for depth of cut, *space* is constant for all extension directions, and **n** · **dir** selects the direction closest that is most perpendicular to the target shape. Although this method may not choose optimally in situations such as Figure 7-17 illustrates, the problem is not debilitating as the user can easily remove the material by moving the tool over the thicker area.

Implementation: Calculating *depth of cut* using the heightfield model is straightforward. For each candidate direction, the depth of cut is evaluated over all heightvectors in the worksurface model. The heightvector direction is used in place of *n*, relying on the approximation that heightvectors are roughly normal to the target surface. The ray / capsule intersection code is used to determine the distance along each heightvector to the neutral-positioned blade, and the heightvector length is subtracted to determine *space*:

```
for each heightvector H with direction $\mathbf{h}$ and length $l_h$
    for each candidate extension direction $\mathbf{dir}$, with allowable extension $extn$
        $dist$ = distance along H to neutral-position capsule
        $space = dist - l_h$
        $depth\_of\_cut = -extn\ \mathbf{h} \cdot \mathbf{dir} - space$
    end
end
return the direction which achieved the largest $depth\_of\_cut$
```

The PFS blade is commanded to extend in the chosen direction.

7.6 Timing Considerations

Real-time performance of the PFS software is important. Fast software allows the tool to respond quickly so that cutting error is minimized. Writing custom functions hard-coded for the shape of the guard or capsule-shaped bur has helped to keep the software fast.

Ultimately, the software speed is limited by the Optotrak update rate. Optotrak rate depends on the number of LEDs being tracked, but for our application is 12ms. The software should be designed to execute fast enough to process every frame of Optotrak data as it becomes available. The Optotrak reads markers continuously and does not wait for the application, so the software should finish execution before the end of the Optotrak frame if synchronization is to be maintained. Software execution time is variable, because the geometric routines can take more or less time depending on what path is taken through their flowcharts. To maintain synchronization, the software checks execution time after calculating each prediction. If execution time passes a threshold, further predictions are abandoned, and blade extension is calculated with only the predictions that are complete.

Bounding Cylinder Test for Compuational Efficiency

Bounding-cylinder tests are used to quickly eliminate entire slabs of heightvectors before the more computationally intensive geometric routines run. For heightfield update, the bounding cylinder of each slab is tested against the bounding sphere of the capsule. If the two do not intersect, then none of the heightvectors in the slab needs to be tested for update. A bounding cylinder / sphere check is similarly used before calculating depth of cut for Choose Extension Direction, and for Snap-to-Surface, a bounding sphere which bounds the guard is used.

Execution Time

Algorithm execution time depends on the size of the workpiece model. Bounding sphere tests also affect execution time. The table below lists average execution time of individual algorithm components on a 1.5GHz Athlon XP 1800+. The table indicates that "snap to surface" and "calculate blade extensions" run multiple times each software cycle, due to multiple lookahead steps and candidate extension directions. The total time for the algorithm is 2.6μsec per heightvector plus 40μsec per slab. The UKR femoral model has 3187 points and 76slabs, giving about 11.3ms total if no heightvectors are eliminated.

	Operating unit	μsec per unit
Update Heightfield	Heightvectors	0.2
Snap to Surface	Heightvectors × 5 timesteps	2.2 per 5
Calculate Blade Extensions	Slabs × 5 timesteps × 5 candidate dirs	40 per 25
Choose Extension Direction	Heightvectors	0.19

For a mostly flat target shape such as the UKR femoral model, we have the option of using only a few slabs (6 for the UKR femur), or using a finer tessellation. Using more slabs may allow the bounding sphere test to throw out more heightvectors, but will increase the amount of time used for "Calculate Blade Extensions". In this case, we chose to use a finer tessellation, but in actual use the number of slabs that were thrown out by the bounding-sphere test was very small and did not justify the extra time taken by "Calculate Blade Extensions". In the future, using fewer, large slabs for this model is recommended.

Graphics Rendering Time

About every 9 algorithm cycles, the software performs the OpenGL rendering calls to update the 3D and cross-section displays. Unfortunately, this causes the algorithm to miss two frames of Optotrak data. This is surprising, because OpenGL should render asynchronously, rendering in the background while the cutting algorithm continues to run. However, OpenGL calls do block when transferring data from the host to the card, to avoid data corruption. This data transfer is significant, because the workpiece model needs to be transferred each time it is updated. This does not completely explain the delay, however: the call time was measured as 7.7ms, which only explains one missed Optotrak frame, since the Optotrak rate is 12ms. I cannot currently explain why the second Optotrak frame is missed.

This latency problem from OpenGL rendering may be eliminated entirely by making better use of the realtime scheduling facilities available in Linux.

Arrangement of Threads

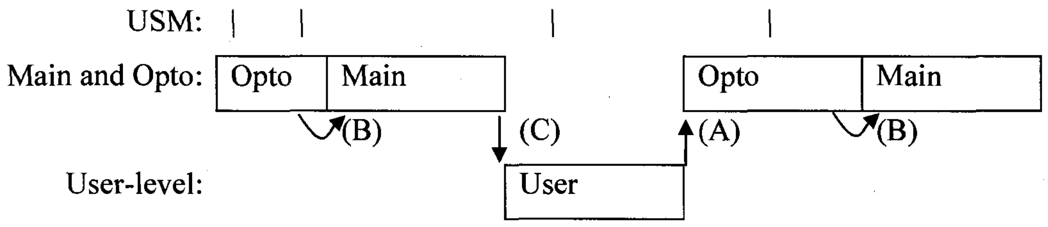
The PFS software runs several threads:

Main: Runs the blade control algorithm and user interface
Optotrak: Reads Optotrak data and sends it to main thread
USM control: Interrupt handler that stops ultrasonic motor when it reaches goal position
Recording: Spools recording data to disk. (Tracking data, etc. are recorded for debug.)
Sound: Plays the sound effects

In addition to standard scheduling, Linux provides 99 realtime priority levels for threads. These provide "strict" priority scheduling, i.e. the highest-priority runnable thread is always run. Currently, only the USM control thread uses realtime scheduling.

The arrangement of threads in the PFS software could be improved to achieve two goals: ensure that the cutting algorithm is not preempted by other user-level tasks, and eliminate the Optotrak frames ignored due to OpenGL calls. To achieve this goal, OpenGL rendering should be moved to a separate thread, and the Optotrak and algorithm threads should then be run at real-time priority, at a lower level than the USM thread. The proposed threads would then interact as shown in Figure 7-18 and summarized below:

USM: Highest priority thread. Runs occasionally and briefly.	
Optotrak:	woken by Optotrak data on SCSI port (A)
	passes data to Main thread
	signals Main thread and blocks waiting for Optotrak data on SCSI (B)
Main:	runs algorithm to completion.
	blocks waiting on Optotrak thread (C)
User-level:	Rendering, Sound, and Recording threads run when all realtime threads are blocked.



Chapter 8. Framework for Understanding PFS Cutting Process

The PFS cutting algorithm devotes significant effort to predicting required blade extensions in order to avoid cutting error. The problem with prediction is that it can be inaccurate. The algorithm employs the extension multiplier α so that the predicted extension can accommodate some prediction error without violating the target shape. However, up to this point, we have no tools to understand how the choice of α affects cutting accuracy.

In this chapter, we develop a model that predicts how α and prediction error affect cutting error. Given a limit ε_{max} on position prediction error, we can then apply the model to choose a value of α for which the PFS should never overcut. Of course, the model is an approximation, so in practice some overcutting will occur.

The larger importance of the cutting error model is that it provides some understanding of the cutting process and the sources of cutting error. This understanding can be applied to experimental results to identify where cutting error comes from in actual use. We will also describe a proposed improvement to the cutting algorithm based on the understanding gained by studying the cutting error model.

Prediction and prediction error drive many aspects of PFS design. The cutting error model can be used to estimate requirements for these design elements. We will apply the cutting error model to investigate how prediction error affects the requirements for blade retraction speed and tracking rate.

The model that we develop is an approximation, because it relies on several simplifying assumptions. However, it is a useful framework for understanding cutting error in the presence of prediction uncertainty, and it yields some important results.

8.1 A Model for Understanding Cutting Error

The central concept behind the error model is that the slope of the workpiece surface determines how errors in predicted tool position correspond to errors in predicted maximum blade extension (Figure 8-1). Given bounds on the error in predicted tool position and on the slope of the workpiece surface, we can generate a bound on error in blade extension, which is cutting error. First, we need a bound on the slope of the workpiece surface.

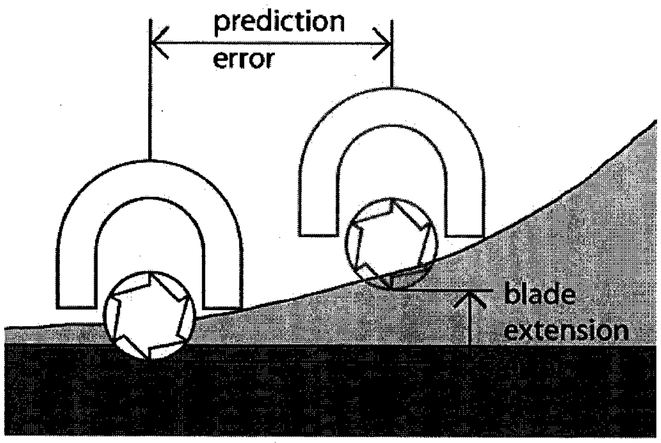


Figure 8-1: Prediction error (known) and workpiece slope (known) yield extension error.

8.1.1 A Limit for Worksurface Slope Based on the Extension Multiplier α

The first step to understanding the cutting process is to understand how the extension multiplier α affects the shape of the workpiece as it is being cut. How far the blade may extend roughly depends on the thickness of waste material that the guard rests on. This maximum allowable extension is then multiplied by α, so that some waste material is preserved, in rough proportion to the thickness of the waste material under the guard. The result is that sharp slopes are avoided, and the waste material thickness slopes gradually toward zero.

To examine the effect of the extension multiplier, we will consider the two-dimensional system shown in Figure 8-2, consisting of an axial cross-section of the PFS blade and guard, and a two-dimensional workpiece. We will assume that the guard remains oriented horizontally and that it always rests on the surface, so that the allowable blade extension is equal to the height of the tool above the surface. Let x in **R** represent the position of the tool center along the horizontal axis, and let $y(x)$ be the height of the workpiece as a function of the tool position. Let w equal half the width of the guard, so that the corner of the guard is at position $x+w$, with height $y(x+w)$ above the target surface. Since we assume the guard is oriented horizontally, this means the allowable blade extension depicted in Figure 8-2 is also $y(x+w)$.

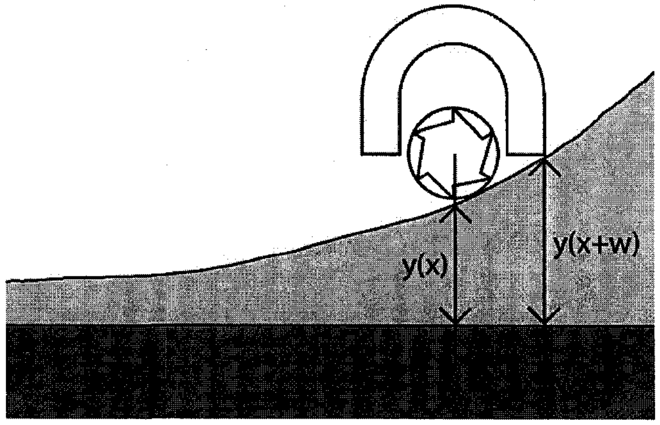


Figure 8-2: Simple two-dimensional PFS system. y(x) is the height of waste material at position x.

Assume now that the PFS always extends the blade the correct amount. With the extension multiplier α, that means the PFS will extend $\alpha\, y(x+w)$ in Figure 8-2, leaving $(1-\alpha)\, y(x+w)$ thickness of material remaining. This means that after cutting,

$$y(x) = (1-\alpha)\, y(x+w)$$

This result applies everywhere the tool cuts. Therefore, the height of every point on the surface is constrained by the height of points around it. The neighboring points enforce a lower bound on the height of each point on the workpiece. We can rewrite this equation as an inequality for all points on the workpiece:

$y(x) \geq (1-\alpha)\, y(x+w)$ for all x. Inequality 8-1

This applies at every point on the workpiece, so we can also say

$y(x+w) \geq (1-\alpha)\, y(x+w+w)$ for all x.

and substitute back into Inequality 8-1, giving

$y(x) \geq (1-\alpha)^2\, y(x+2w)$ for all x.

we can continue this by induction as:

$y(x) \geq (1-\alpha)^{|n|}\, y(x + n\, w)$ for all n in **Z**. Inequality 8-2

Note the absolute value symbol around n in the exponential.

Ideally, we would like to extend inequality 2 to hold for n in **R** instead of n in **Z**. However, this change cannot be formally made even if y is assumed to be continuous or differentiable. For instance, $y(x) = \sin(2\pi\, x/w)\, (1-\alpha)^{-x/w}$ satisfies inequalities 8-1 and 8-2, but does not satisfy inequality 2 if n is assumed to be in **R**: For x in $[0,w)$, each series $\{y(x+nw) \mid n \text{ in } \mathbf{Z}\}$ is independent of the others. Although no degree of "mathematical" smoothness is sufficient to generalize inequality 8-2 to the real numbers, the "practical" smoothness seen in real life, because of factors such as limited material properties and the non-negligible width of the guard, limits the disparity between the independent sequences $x+nw$, so that the surface approximately fulfils inequality 8-2 for all real n. For the purposes of this analysis, we *assume* that this is true, and we rewrite inequality 8-2 as

$y(x) \geq (1-\alpha)^{|\varepsilon|/w}\, y(x+\varepsilon)$ for ε in **R**. (Worksurface Slope Assumption) Inequality 8-3

We will refer to this inequality as the *worksurface slope assumption* because it constitutes a limit on the slope of the workpiece surface at a given point. This limit can be seen as an exponential curve rising from any point on the workpiece, which defines an upper bound on the height of the surrounding workpiece surface (Figure 8-3). Note that the maximum slope of the workpiece increases as the thickness of waste increases.

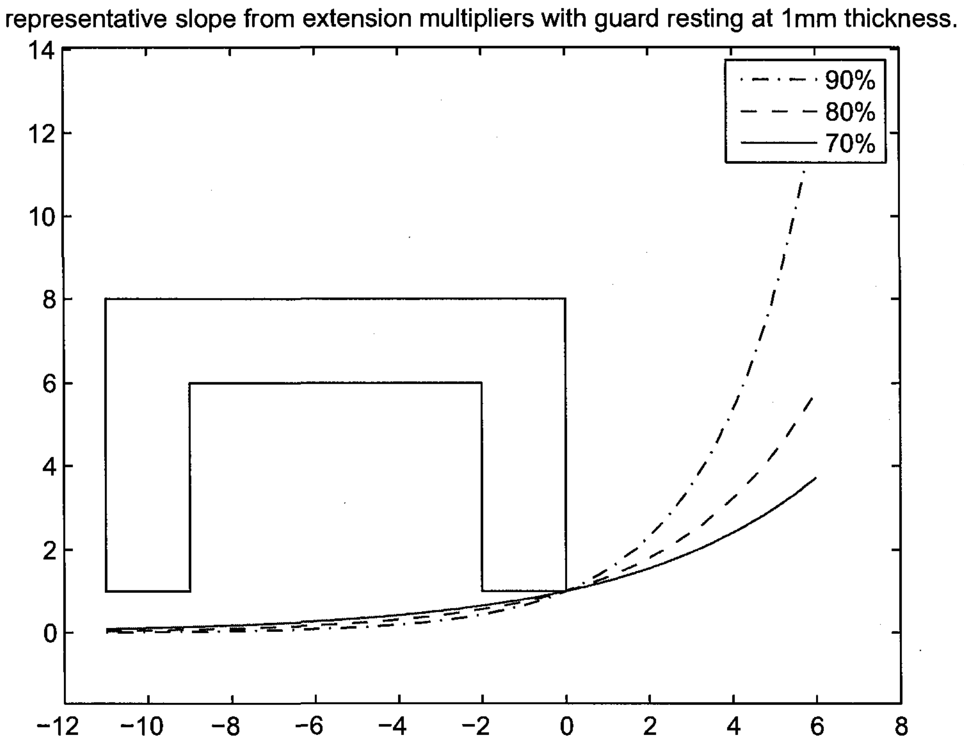


Figure 8-3: The maximum slope allowed by the worksurface slope assumption, for various values of w. The axial cross-section of the guard (drawn crudely in MATLAB) is shown for comparison.

Limitations of Worksurface Slope Assumption

The accuracy of the worksurface slope assumption is limited by the validity of the assumptions it relies on. Some of these assumptions are inherent in the 2D model of Figure 8-2. First, the model assumes that the guard always stays parallel to the target shape. Second, the 2D model considers only the axial cross-section of the tool. The final assumption is that inequality 8-2 can be extrapolated to inequality 8-3.

The assumption that the guard remains parallel to the target shape allows us to equate $y(x+w)$ with the maximum blade extension. If the guard is not parallel, the relationship between $y(x+w)$ and the extension distance changes, increasing or decreasing the worksurface slope.

Consideration of the axial cross-section of the tool means that how far the blade cuts is limited by the waste material thickness on both sides of the blade. In contrast, in the sagittal cross-section (Figure 8-4), the guard does not surround the blade on both sides, so blade extension is not constrained by the thickness of the material beyond the tip of the tool. The relationship between $y(x)$ and $y(x+w)$ is not enforced, so the worksurface slope assumption fails. Fortunately, the PFS is usually moved primarily in the axial plane, so this problem is minimized.

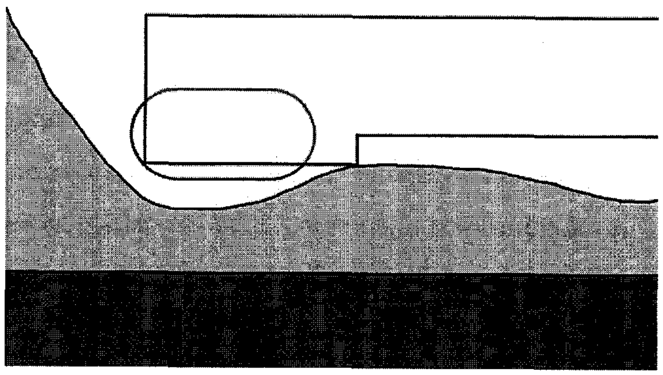


Figure 8-4: sagittal cross-section. The guard does not rest on material beyond the tip of the guard, so that material does not limit cutting depth.

The fact that inequality 8-3 does not follow from inequality 8-2 is not merely a theoretical distinction. This assumption can fail in practice, although the magnitude of the error is limited. Figure 8-5 illustrates a worst-case scenario. The dotted line illustrates the limit defined by the worksurface slope assumption. The shaded area has not been cut and does not satisfy inequality 8-3, even though every point does satisfy inequality 8-2. Typical results are much better than this worst-case scenario. Typically, the user slides the tool smoothly side-to-side, which results in a smoother surface that better approximates the worksurface slope assumption.

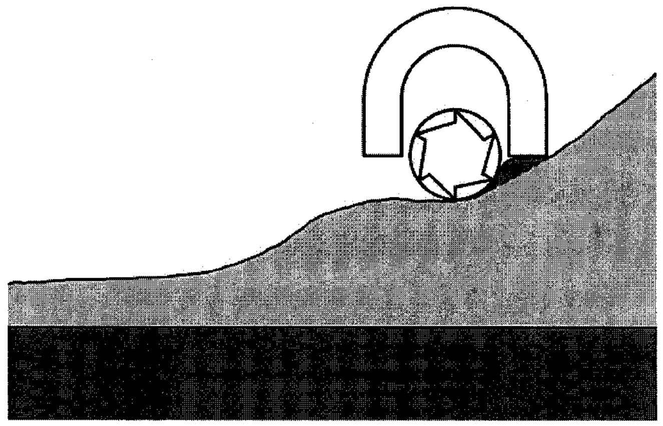


Figure 8-5: Worst-case scenario for failure of workesurface slope assumption. The dark area above the dotted line satisfies inequality 2, but not inequality 3.

8.1.2 A Limit on Cutting Error Based on Limited Worksurface Slope

In this section, we will demonstrate that the worksurface slope assumption determines how errors in position prediction correspond to errors in blade extension, and thus to cutting error. Once this relationship is understood, it can be turned around to limit cutting error: given a limit ε_{max} on position prediction error, an extension multiplier α can be chosen which ensures that the tool will not overcut.

The idea behind this analysis is simple. The slope of the workpiece determines how much an error in predicted tool position results in error in the tool's predicted distance from the surface. If we assume distance from the surface equates to allowable extension, then the slope of the workpiece relates error in position prediction to error in allowable extension. Since the algorithm only extends α times the predicted allowable extension, small amounts of prediction error can be tolerated without overcutting.

Consider the 2D model from Figure 8-2, with the tool traveling over a workpiece that satisfies the worksurface slope assumption. Assume the algorithm predicts the tool position only one step in advance. Let x be the position of the tool at a given timestep t, and $x+\varepsilon$ be a prediction, from the prior timestep, of the tool position at timestep t. (Figure 8-6)

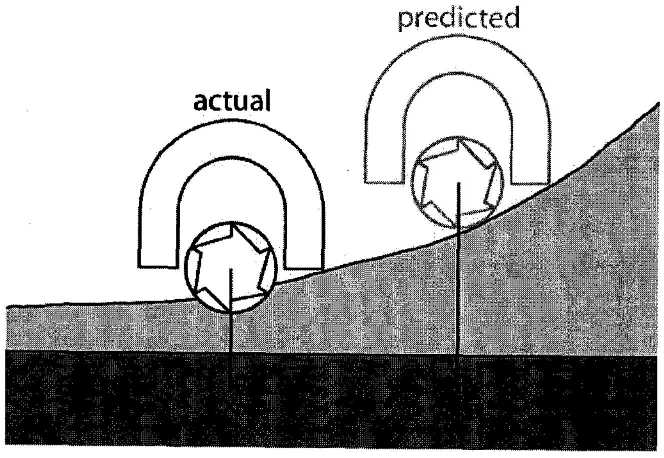


Figure 8-6: Actual tool at position x, predicted tool at position x+ε.

Based on the predicted position x+ε, the algorithm will have extended the blade a distance $\alpha\, y(x+w+\varepsilon)$. However, the actual height of the tool over the target shape is only $y(x+w)$. The cutting error is given by the commanded blade extension minus the allowable blade extension:

$err = \alpha\, y(x+w+\varepsilon) - y(x+w)$ Equation 8-4

A positive value for this quantity indicates that the blade has cut deeper than the target shape. Although this equation identifies cutting error in the positive and negative directions, only positive cutting error actually produces error in the final result. Negative error just means that the tool does not cut as efficiently as it could have.

We can simplify the expression for *err*, because the worksurface slope assumption relates y(x+w+ε) to y(x+w) as:

$$y(x+w+\varepsilon) \leq (1-\alpha)^{-|\varepsilon|/w}\, y(x+w)$$

so equation 8-4 becomes

$$err \leq \alpha\, (1-\alpha)^{-|\varepsilon|/w}\, y(x+w) - y(x+w)$$

Given ε and w, we can now choose an α such that $err<0$ for all x. More to the point, given a bound ε_{max} on the position prediction error, and assuming the worksurface slope assumption, we can choose an extension multiplier α such that the tool will never cut past the target shape. We do so by requiring

$$err \leq \alpha\ (1-\alpha)^{-|\varepsilon\_max|/w}\ y(x+w) - y(x+w) \leq 0$$

since $y(x+w)$ is positive, it can be factored out, leaving

$$\alpha\ (1-\alpha)^{-|\varepsilon\_max|/w} - 1 \leq 0 \ \ => err \leq 0 \quad \text{Equation 8-5}$$

If we choose α so that the inequality holds, then the tool should never overcut. The solution is transcendental for α. For intuition, the following table lists the allowable values for ε for several values of α. For our PFS surgical tool, w=5.5mm.

	α=0.90	α=0.80	α=0.70	α=0.60
ε_{max}/w	0.046	0.139	0.296	0.557
ε_{max} for w=5.5mm	0.253	0.7645	1.628	3.0635

Table: Representative values of position prediction error ε and corresponding values of necessary extension multiplier α.

Of course, this analysis is an approximation, and in practice some overcutting will always occur. It may be necessary to tune α to achieve the desired results. In fact, there is not really one "correct" value of α: smaller α will cut more accurately, but larger α will cut more efficiently. The tradeoff between accuracy and efficiency must be chosen to suit the application.

If this were the only result of this error analysis, it might seem pointless, since α could easily be tuned by hand anyway. Section 8.2 discusses three applications where this error analysis can provide insight into the nature of the cutting process and suggest improvements.

Limitations of the Cutting Error Model

This analysis has several important limitations. First, although it relies on the worksurface slope assumption, it does not guarantee that the worksurface slope assumption continues to hold. The analysis only says that in the presence of prediction error, the blade will not violate the target shape. However, it can violate the limit defined by the worksurface slope assumption. Violation of the worksurface slope assumption can then lead to cutting error in the future.

Another limitation is that like the worksurface slope assumption, the error analysis depends on the 2D model of Figure 8-2. This model has two assumptions: that the guard is parallel to the target shape, and consideration only of the axial cross-section of the tool. This section of the analysis does not actually depend on the axial cross-section. It simply assumes that some part of the guard was predicted to touch down at some point, but touched down at a different point, and the height difference between those points is given by the worksurface slope assumption.

The analysis does rely on the guard being parallel to the workpiece, because it equates worksurface height with allowable extension. However, under the usual conditions of tool use, equation 8-4 still provides a good approximation to cutting error. These conditions are that the guard's angle of tilt is small, that the angle of tilt remains relatively constant, and that the target shape is relatively flat.

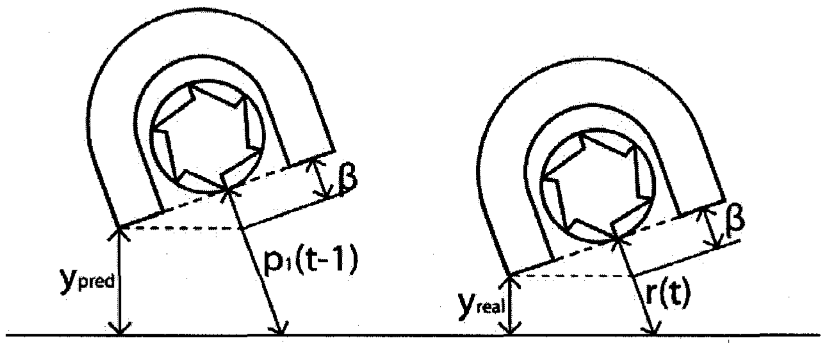


Figure 8-7: The effect of relatively constant tool tilt on cutting error analysis. If the tool orientation is close to constant and close to flat, we can estimate $y_{real}(t) \approx (r(t) - \beta_t) \cos \theta$ and $y_{pred}(t) \approx (p_1(t\text{-}1) - \beta_t) \cos \theta$ for some small β_t.

Under these conditions, we have the situation shown in Figure 8-7. The familiar cross-sectional view is maintained to keep the illustration readable, but the following discussion is valid for an arbitrary orientation of the tool in 3D. Let $p_1(t\text{-}1)$ be the predicted allowable blade extension for time t as calculated at time t-1. Let $r(t)$ be the real allowable blade extension at time t. Let $y_{pred}(t\text{-}1)$ be the predicted thickness of the waste material for time t as calculated at time t-1, and $y_{real}(t)$ be the actual thickness at time t. These $y_{pred}(t\text{-}1)$ and $y_{real}(t)$ are the thickness of the waste material at the point of contact of the guard with the workpiece, as determined by Snap-to-Surface. Under our assumptions about typical tool use, we can estimate

$y_{real}(t) \approx (r(t) - \beta_t) \cos\theta$ Equation 8-6

$y_{pred}(t) \approx (p_1(t\text{-}1) - \beta_t) \cos\theta$ Equation 8-7

For some β_t, with θ being the angle between the guard and target surface. For the planar case illustrated, $\beta_t = w \tan\theta$, but it will be different for other 3D orientations. Then equation 8-4 can be rewritten

$$err_{model}(t) = \alpha\, y_{pred}(t\text{-}1) - y_{real}(t)$$

Substituting equations 8-6 and 8-7 into the above, we get

$err_{model}(t) \approx \alpha\, (p_1(t\text{-}1) - \beta_t) \cos\theta \text{ - } (r(t) - \beta_t) \cos\theta$

$err_{model}(t) \approx (\alpha\, p_1(t\text{-}1) - r(t) + (1\text{-}\alpha)\, \beta_t) \cos\theta$ Equation 8-8

Since $p_1(t\text{-}1)$ and $r(t)$ are the actual predicted and real blade extensions, we will call

$err_{actual}(\mathrm{t}) = \alpha\, p_1(t\text{-}1) - r(t)$ Equation 8-9

Note that err_{model} is an approximation of err_{actual}. Substituting equation 8-9 into equation 8-8, we get:

$err_{model}(t) \approx (err_{actual} + (1\text{-}\alpha)\,\beta_t) \cos\theta$

Based on the assumption that the guard's angle of tilt is small, β_t and θ are small. In addition, $1\text{-}\alpha$ is typically 0.3 or smaller. Under these conditions, err_{model} is a good approximation of err_{actual}. More importantly, since $(1\text{-}\alpha)\beta_t$ is positive, $err_{model} \leq 0$ implies $err_{actual} \leq 0$. This means that where the error model predicts the tool will not overcut, err_{actual} also predicts the tool will not overcut. Therefore, if our assumptions about typical tool use hold, then the effect of tool tilt on the analysis in this section is minimal.

8.2 Applications of the Cutting Error Model

The model we have developed for cutting error is an approximation because it relies on several simplifying assumptions. However, it is still useful as a framework for understanding the cutting process and the causes of cutting error.

One application of the cutting error model is to analyze experimental results to determine the causes of cutting error. The model states that if the worksurface slope limit and the prediction error limit are satisfied, then the tool should not overcut. In cases where the tool did overcut, we can ask whether the error was due to exceeding the slope limit or the prediction limit. If, for instance, the slope limit fails, we can ask which simplifying assumption was primarily responsible for that failure. In this way, we can classify the largest contributors to cutting error. This understanding is valuable for improving performance. We take this approach to classifying error sources in Chapter 10.

In this section, we present two further applications of the cutting error model. First, we use the model to discuss the relationship between major design parameters for the PFS. With this understanding, we describe a simple procedure that could be used to estimate optimal design parameters for PFS mechanisms and systems that might be built for other applications.

The second application of the cutting error model is a proposed improvement to the current PFS blade control algorithm that has been illuminated by work on the model.

These changes alleviate the weaknesses of the worksurface slope inequality by removing some of the assumptions that it relies on.

8.2.1 Application of the Model to PFS Design Parameter Selection

When development of the PFS began, there were many design parameters we had to guess at. For instance, we wanted to know how fast the blade must retract. We also weighed using a cheaper, slower tracking system instead of the Optotrak, and wanted to know what tracking rate was necessary. In both cases, we lacked the tools to determine how the design parameter would affect tracking accuracy. One application of the cutting error model is that it provides a way to reason about questions like what blade retraction speed is necessary. In fact, the cutting error model relates several important questions about tool design:

How fast must the blade retract?
How fast must the optotrak sample?
How fast may the user move?

The answers are intimately related and one can be traded off against another. For instance, if the user doesn't move as fast, then the blade doesn't need to retract as quickly. Figure 8-8 illustrates the relationship between the three desired parameters, plus α, the extension multiplier; and ε_{max}, the prediction error limit. Many of the connections have a "natural" direction that we think of the dependency going, but often these connections can be driven the other way as well. For instance, normally α is chosen based on ε_{max}, but if we want a particular α, we can calculate what ε_{max} is necessary.

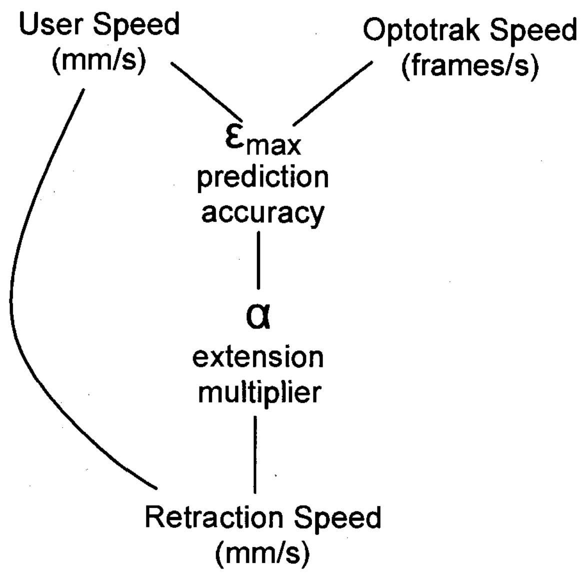

User Speed (mm/s)
Optotrak Speed (frames/s)
ε_{max} prediction accuracy
α extension multiplier
Retraction Speed (mm/s)

Figure 8-8: Relationship between user speed, optotrak speed, and retraction speed.

User motion and Optotrak rate determine prediction accuracy ε_{max}. Slow Optotrak rate will cause poor prediction accuracy because the prediction is over a longer time. User motion is complex, and connot be easily modeled or predicted. The best way to find ε_{max} is to measure it from actual user motion while using the PFS. User motion is hard to control or limit. However, if it is necessary to reduce the contribution of user motion to ε_{max}, enforcing a speed limit may reduce prediction error in the user motion. A speed limit could be enforced by sounding an alarm or retracting the blade.

The next connection in Figure 8-8 is between ε_{max} and α. α can be chosen based on ε_{max} according to the cutting error analysis in the previous section. Note that we can also derive ε_{max} based on α if necessary.

One way of estimating necessary retraction time is with the user speed and the worksurface slope assumption. User speed determines how fast the tool moves across the workpiece, and worksurface slope determines how motion across the surface translates to change in waste material thickness. The worst case is at $y(x) = m/\alpha$, where m is the maximum possible blade extension. For lower points than m/α, the slope is more gradual, and for higher points, the tool is far enough away that it doesn't need to immediately retract.

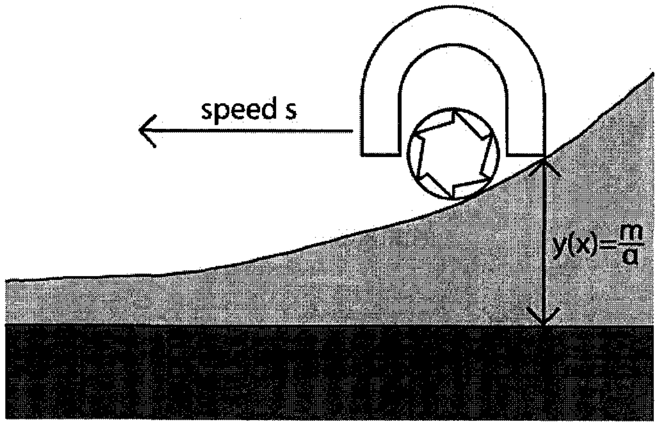


Figure 8-9: Tool on surface of maximum slope moving at speed s, with height of waste material $y(x) = m / \alpha$. Worksurface slope and tool speed determine how fast blade must retract.

Consider the situation in Figure 8-9. Assume that the worksurface slopes at the maximum allowable by the worksurface slope inequality. The tool is moving at speed s across the surface, the waste material thickness is at the worst-case $y(x) = m / \alpha$. The horizontal distance that the tool travels in one timestep is $s\ \Delta t$. Assume that the algorithm has correctly predicted that at the start of the next timestep, the blade extension must not exceed $\alpha\ y(x + s\ \Delta t)$. To achieve this extension, the blade must move at speed

speed to meet next required extension = $(\ \alpha\ y(x) - \alpha\ y(x + s\ \Delta t)\ ) / \Delta t$ Equation 8-10

An alternate formulation is to consider the speed required to meet not only the next predicted retraction requirement, but every intermediate value as well. Consider again Figure 8-9, with the tool moving at a constant rate s, and at time $t_0 = 0$, let $y(x) = m / \alpha$. The maximum allowable extension is graphed in Figure 8-10 as a function of time. To satisfy the predicted extension requirement for time $t_1 = 0.012$s, the tool only needs enough speed to travel along the line labeled "trajectory 1". But to satisfy the allowable extension at all times between t_0 and t_1, a faster initial retraction speed is required, as illustrated by the line labeled "trajectory 2". For trajectory 2, the required retraction speed is the limit of equation 8-10 as Δt goes to zero:

speed to meet intermediate required extensions = $s\,\alpha\,y'(x)$

The algorithm cannot explicitly detect these intermediate required extensions, but if the blade retracts fast enough they can be satisfied implicitly.

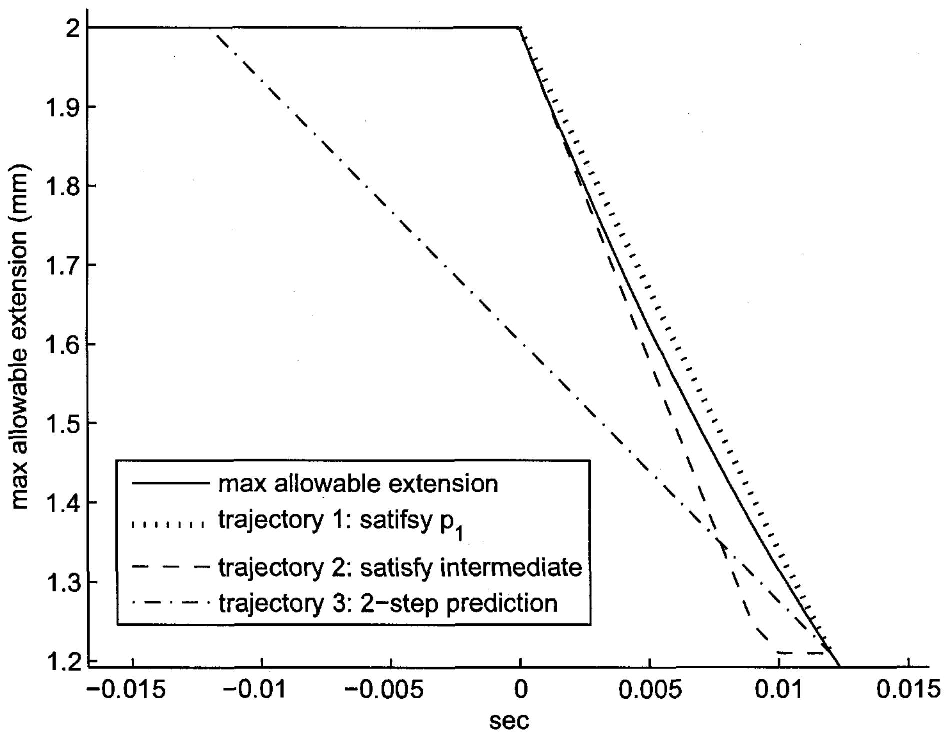


Figure 8-10: Required retraction speed as a function of time, for the situation depicted in Figure 8-9. The solid line is the maximum allowable extension as a function of time. Trajectory 1 shows the motor trajectory necessary to satisfy prediction p_1 by time t_1=0.012s. Trajectory 2 shows a motor trajectory where the motor is fast enough to satisfy the allowable extensions for the times between t_1 and t_2 as well. Trajectory 3 shows the motor trajectory necessary if two-step prediction is used.

For the PFS surgical tool, the maximum user speed was about 0.1m/s and α was 0.9. For the worst case $y(x) = m/\alpha$ = 2mm/0.9, this gives

speed to meet next required extension = $(\,\alpha\, y(x) - \alpha\, y(x + s\,\Delta t)\,) / \Delta t$ = 65mm/s speed to meet intermediate required extensions = $s\, \alpha\, y'(x)$ = 83mm/s

For comparison, the PFS surgical tool was designed to retract 4mm in 100ms, which averages to 40mm/s.

Predicting more than one step into the future can reduce the required blade retraction speed. Figure 8-10 depicts a third line, labeled "trajectory 3", that depicts the required blade retraction speed if allowable extensions can be correctly predicted two steps in advance. The minimum retraction speed required to meet this extension is less than required if only one step ahead is predicted. Note that the prediction allows slower retraction speed by averaging the slope over a longer time period. Importantly, 2-step prediction only presents an advantage because the occasions of high slope are averaged together with periods of lower slope. If the slope were uniform during the prediction period, 2-step prediction would not offer any advantage for required retraction speed.

This analysis of blade retraction speed assumes that the worksurface slope assumption holds and that predictions are perfect – that the PFS actually knows how much it will be required to retract. When those assumptions fail, the required retraction speed is bounded only by the overall tool speed. In this case, that would require blade retraction of 100mm/sec. On the other hand, a blade retraction speed less than those estimated here may actually give acceptable results, because in practice, worksurface slope rarely reaches the maximum at all points.

The cutting error model has thus illustrated the connections between user motion, tracking rate, and required blade speed. However, these connections are not strict equalities. Blade retraction speed and ε_{max} both place upper bounds on α, but a smaller value of α will also satisfy the constraints. For instance, user motion and tracking rate can dictate a required blade retraction speed, but if that is infeasible, a lower value of α can be used to allow a slower blade retraction speed.

A Procedure for Determining Design Parameters for a PFS Application

The derivations above suggest how design parameters might be chosen when designing a PFS system for a new application. First, we would determine ε_{max} based on tracking rate and user motion. Then α is determined by ε_{max}. Finally, the necessary blade retraction speed is calculated based on α and the tool speed.

The challenge here is to estimate tool speed and ε_{max} before a prototype is actually built for the new application. I suggest that a simple unactuated mockup can be used to measure tool speed and ε_{max}. The mockup should be tracked by the tracking system. The user should use the mockup to cut a sample workpiece, so that the computer can measure tool speed and ε_{max}. Although the best prediction of tool positions requires an accurate worksurface model, simple extrapolation may be acceptable for the purpose of estimating ε_{max}.

Of course, this analysis is only an estimate. Once the new PFS system is built and tested, it may be necessary to tune α or adjust other parameters to achieve the desired performance and desired tradeoff between accuracy and efficiency. However, this method provides a quantitative way to reason about some of the important factors that influence PFS performance.

8.2.2 An Improvement to the Cutting Algorithm

Development and study of the cutting error model have suggested an improvement to the cutting algorithm which may improve accuracy. The change has not yet been implemented. The proposed improvement addresses the limitations of the worksurface slope assumption discussed above: the assumption that the guard is parallel to the target shape, the consideration only of the axial cross-section, and the assumption that inequality 8-2 generalizes to inequality 8-3.

Presently, the worksurface slope assumption is enforced implicitly through the action of the guard and the extension multiplier α. I propose to enforce a slope limit explicitly by calculating how far to extend based on the height of the surrounding heightvectors. The PFS should use a "virtual" guard to identify the surrounding heightvectors. How far the tool should cut can then be constrained by the neighboring heightvectors to achieve the desired slope. The allowable extensions can then be calculated to leave the desired margin rather than cutting all the way to the target shape.

Implementation of the virtual guard can be similar to the guard modeling used in the snap-to-surface calculation. Each heightvector on the target shape is intersected with the planes of the guard. The point where the heightvector intersects the plane is compared to the shape of the virtual guard. Unlike the actual guard, the virtual guard should completely surround the blade so that the heightvectors surrounding the blade in all directions are considered.

Once the heightvectors that intersect the virtual guard are identified, we must decide how far the blade may cut based on the surrounding area. The simplest approach is just to use the extension multiplier: if the maximum height of a heightvector that hits the virtual guard is h, then the blade may only cut to within α h of the target shape.

This will address the problems that stem from the use of the simple 2D model. The sagittal cross-section is now not a problem, because unlike the real guard, the virtual guard surrounds the blade in all directions. It is also not a problem if the guard is not parallel to the workpiece, because now the algorithm explicitly computes how much waste material must not be cut, and the algorithm computes the appropriate extension independent of tool orientation.

The remaining limitation of the worksurface slope assumption is the extension of inequality 8-2 to inequality 8-3. This can be improved somewhat by using a more complex method to determine the cutting distance from the heightvectors that hit the virtual guard. Rather than simply using an extension multiplier over all heightvectors, we can consider where each heightvector hits the guard. To enforce a smooth slope, heightvectors close to the center of the guard should allow the blade to extend less, while heightvectors that hit toward the outside of the guard can allow the blade to extend further relative to their own heights. For instance, to enforce the worksurface slope assumption, each heightvector should require that the thickness of material not cut is $(1-\alpha)^{\varepsilon/w}$ h, where ε is the distance from the heightvector to the center of the blade in the plane of the virtual guard, and h is the height of the heightvector. This is illustrated in Figure 8-11. Heightvector A is 2mm long and 6mm from the center of the blade, so it requires the blade leaves $(1-\alpha)^{6mm/5.5mm}$ 2mm = 0.16mm waste material remaining. Meanwhile, heightvector B is 1.5mm long, but only 4mm from the center of the blade, so it requires that the blade leaves $(1-\alpha)^{4mm/5.5mm}$ 1.5mm = 0.28mm of material remaining, which is more restrictive. This avoids the situation pictured in Figure 8-5, and results in a smoother slope which will better approximate that described by inequality 8-3.

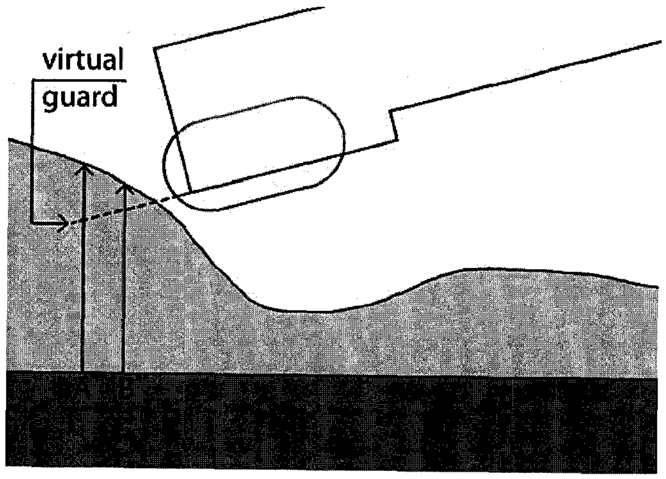


Figure 8-11: Heightvectors A and B intersect the virtual guard, and therefore constrain how far the blade may cut.

Note the flexibility that this approach gives. The virtual guard doesn't have to be the same size as the actual guard, so it can consider a wider neighborhood around the blade. Also, instead of the exponential surface defined by the worksurface slope assumption, an arbitrary monotonic profile can be used.

Based on the heights of the heightvectors that hit the guard, the algorithm determines how much waste material the tool should preserve. The final step is to calculate how far the blade should extend to meet this constraint. The standard routine can be used, which calculates how far the blade can extend until it hits each triangle in the target shape. The radius of the blade model is simply increased by the desired waste thickness, so that the routine will calculate how far the blade should extend to preserve the desired material thickness.

Chapter 9. Experimental Validation of Cutting Accuracy

The PFS was used in several cutting experiments to measure cutting accuracy, and gather data about conditions during use. Foam blocks and Sawbones femurs were cut with the UKR target shape. Each cut sample was scanned with a Minolta Vivid 910 laser scanner to measure cutting accuracy. (Thanks to Daniel Huber and Martial Hebert for use of the scanner.)

Extensive data were collected during tool use as well. The PFS software records and timestamps every frame of Optotrak and encoder data, along with program state and diagnostic messages. This allows an entire trial to be replayed for analysis. Interactive playback was used extensively during development and testing. Data can also be extracted for automated analysis. Of particular note, the heightfield model resulting at the end of the case was used for comparison to the laser scan of the actual workpiece.

Two series of tests were run. The first was used to evaluate cutting accuracy in a controlled setting on foam blocks. The second examined cutting accuracy on foam blocks and Sawbones femurs through small incisions.

9.1 Foam Block Experiments

The foam block tests were designed to mimic the UKR procedure but in a more controlled setting. Figure 9-9-1 illustrates the experimental setup. Identical foam blocks were used with a jig made from MDF on which was mounted an optical marker. For each trial one block was screwed to the jig and registered by pressing against three flat, orthogonal surfaces before screwing it down. This repeatable positioning allows the jig to be registered to the optical marker just once instead of for each trial.

The target shape was the UKR femoral shape, which consists of three facets. The orientation of the target shape corresponded to its orientation during the UKR procedure, ensuring tracker viewing angles were comparable to those used with sawbones UKR.

Users were instructed to use a side approach angle for the distal surface cut and a front approach for the chamfer and posterior cuts, as is done in the sawbones UKR. The side approach and front approach use different optical markers on the tool.

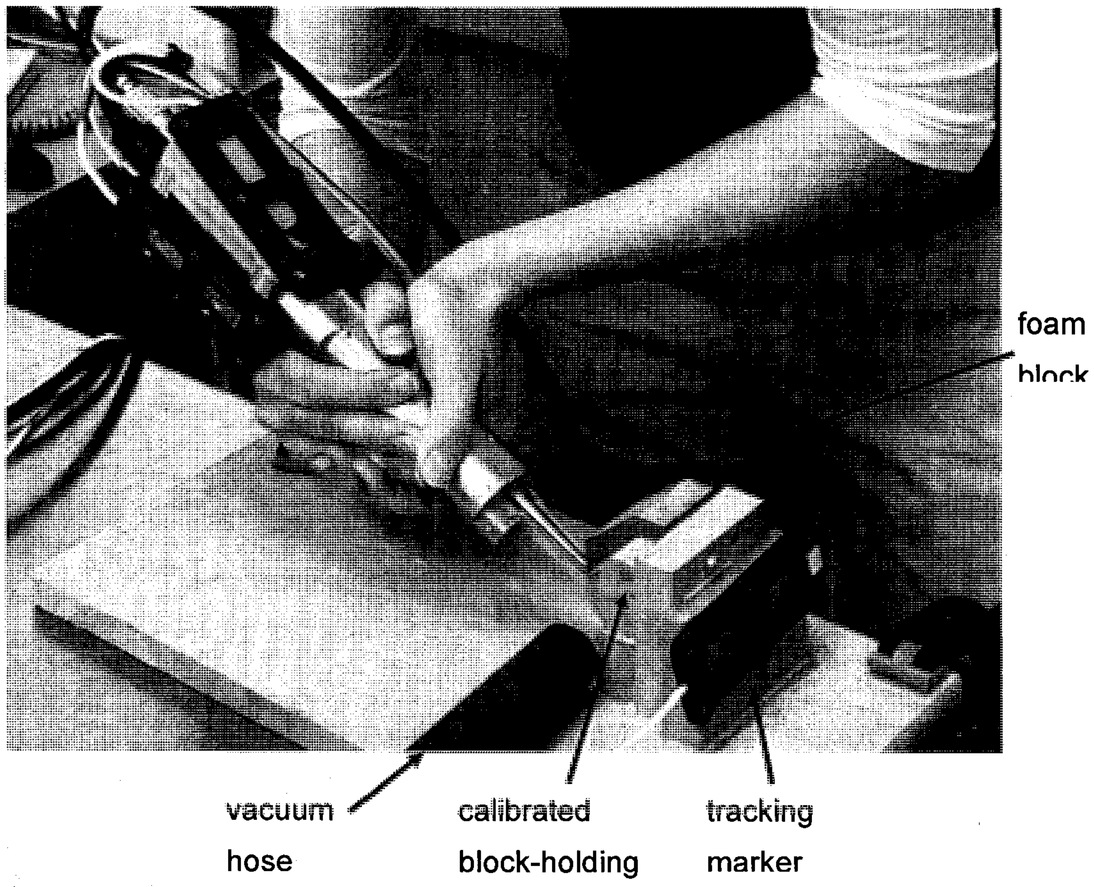


Figure 9-9-1: Foam block experimental setup. Foam block is mounted in jig and cut by user. Target shape is UKR femoral shape.

Three users cut three blocks each. User 1 was the author, who has used the PFS perhaps 50 times. User 2 had used the PFS perhaps 5 times previously, and user 3 had never used the PFS. All users were familiar with the PFS concept and were given a brief orientation.

9.1.1 Measured Accuracy

Each cut block was laser scanned to determine cutting accuracy. To extract from the scan the points on the surface that was cut by the PFS, a combination of manual and automatic techniques were used. Extraneous points were removed, and points within a small margin of the edge of the cut surface were also removed. Each scan had about 15,000 points on the cut surface.

Our primary interest in accuracy results was fit accuracy, because it is a more challenging constraint for the PFS than implant position accuracy. Since the tool cuts a small patch at a time, with little correspondence between neighboring patches, most error sources in the system contribute equally to fit and position accuracy. Meanwhile, fit accuracy requires much tighter tolerances than position accuracy. For instance, 0.5mm is significant error for a gap between the bone and the implant (fit accuracy), but a small error as a change in leg length (position accuracy). Therefore the challenge with the PFS will be ensuring sufficient fit accuracy.

To measure fit accuracy, the least squares alignment between the scan points and the target shape was found. The residual distance from each scan point to the target shape was termed error, with the sign set to positive for points below the target shape and negative for points above the target shape. Positive error indicates that the tool cut too far.

The distribution of fit error over all points on all samples is shown in Figure 9-2. The residual RMS error to the target shape was 0.16mm.

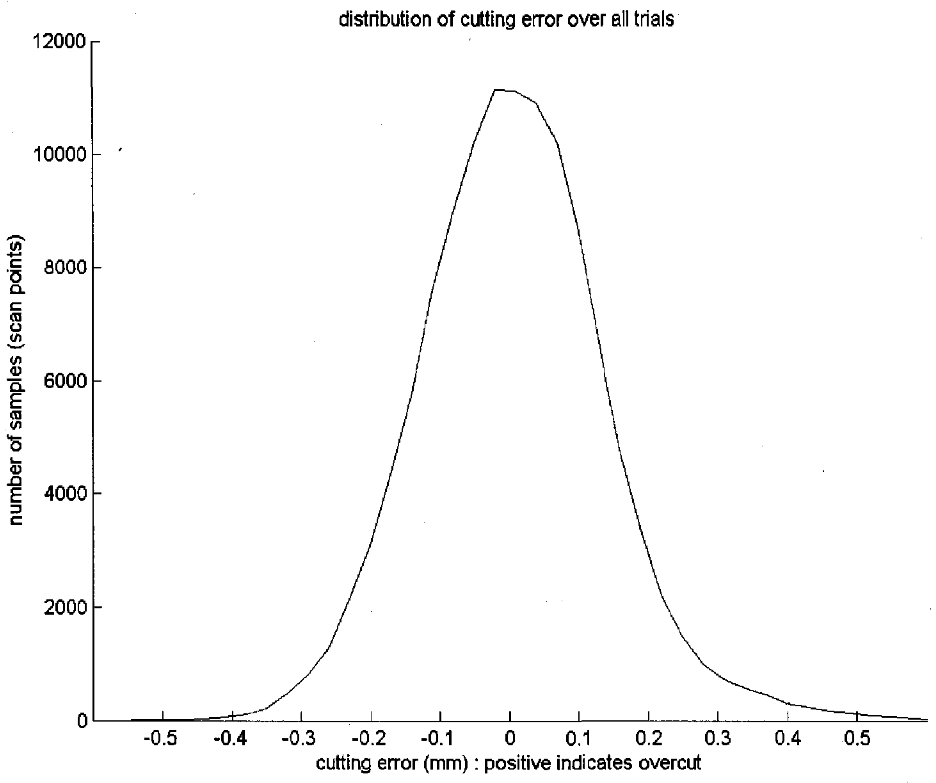


Fig 9-2: Distribution of cutting error in foam block trials.

The Effect of Experience on Accuracy

One important question is whether the PFS was less accurate for less experienced users. The distribution of cutting error is broken down by user in Figure 9-3. No strong effect of experience is seen. In fact, my own results (User 1) are the worst. It may be that I cut most aggressively because I placed the most trust in the PFS, while the others were more cautious.

In Figure 9-4, we examine the effect of experience again by plotting the cutting error for each trial of the inexperienced user, User 3. Again, no significant accuracy difference is seen between User 3's very first use of the PFS, and the third use. Thus we conclude that user experience has no important effect on cutting accuracy.

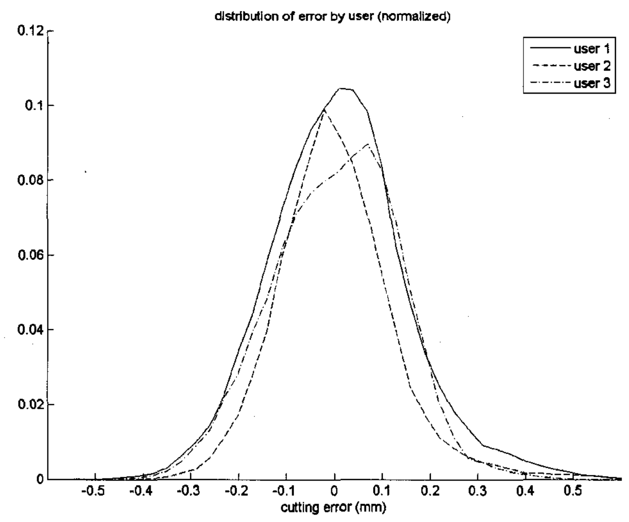


Figure 9-3. Distribution of error for each user. Lack of experience does not noticeably impact achieved cutting accuracy.

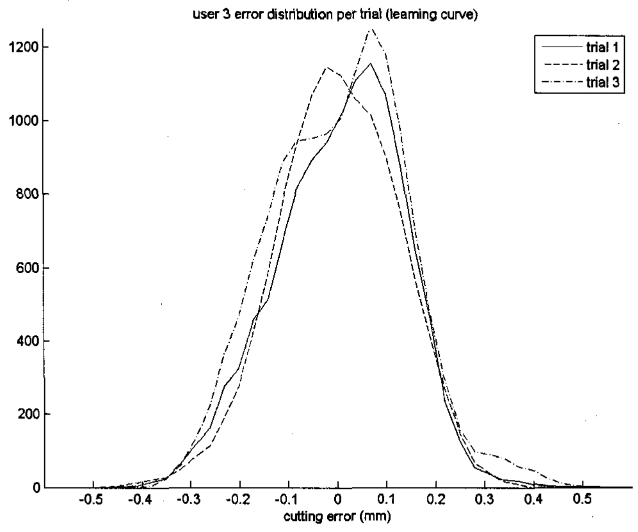


Figure 9-4. Distribution of error for first, second and third trial for user 3. Again, level of experience did not affect cutting accuracy.

Spatial Distribution of Cutting Error

Another question of interest is how the error is distributed spatially on the target shape. Figure 9-5 maps the surface generated by user 1 in trial 2. Scan points that are overcut by more than 0.2mm are marked with an "x". Scan points that are undercut by more than

0.2mm are marked with a "+". Other scan points are marked with a dot. The overcut seen along the edges between faces is typical for the results in these trials. Note that the bottom facet, which is least accessible by the user, shows the most cutting error.

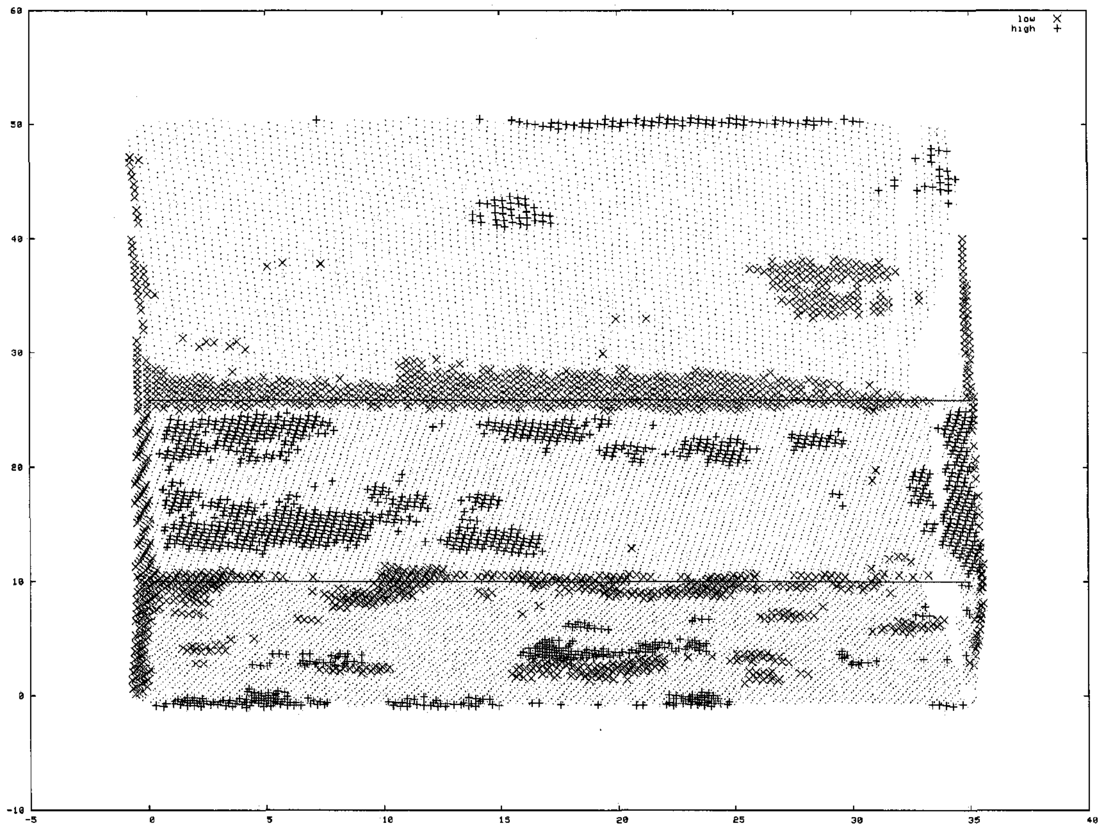


Figure 9-5: Map of error for User 2, Trial 1. "x" indicates overcut of 0.2mm or more. "+" indicates undercut of 0.2mm or more.

Comparison to Accuracy Requirements

Chapter 4 cited several studies on the accuracy of conventional instrumentation, and other studies on the accuracy required for bone ingrowth in cementless implants. For accuracy of conventional saw-based knee instrumentation, the best estimate was provided by [Toksvig-Larsen 1994], who cited 0.26mm RMS error on a single, flat tibial cut. The PFS measured accuracy of 0.16mm RMS certainly compares favorably with this. Although this is not a head-to-head comparison, and very possibly conventional technology has improved since 1994, this strongly suggests that the PFS will be able to achieve accuracies competitive with conventional instrumentation.

For the accuracy requirements of cementless implants, the most popular requirement seemed to be a maximum gap of 0.5mm, from [Sandborn 1988]. Exactly how the PFS measured accuracy translates into bone gap depends on how much the high points on the bone compress or subside. To a lesser extent, it also depends on the geometry of the implant. We can imagine compression and subsidence of the high points as cutting off the left-hand tail of the distribution in Figure 9-2. If the distribution is cut off at -0.3mm, we can see that a large majority of the bone material will be within 0.5mm of the implant. Although much more extensive study is needed to determine the suitability of the PFS for cementless knee replacement, this initial result is promising.

9.1.2 Cutting Time

Execution time for each trial is listed below. The test block was wider and contained more waste material than typical for UKR, so actual cutting time in surgery may be less. Cutting times for the Sawbones experiments will be more representative of actual surgery.

	Execution Time For Each Trial (min:sec)		
User 1	n/a	8:18	8:40
User 2	11:05	11:38	9:42
User 3	13:34	9:25	7:36

Note that in the first trial user 3 was much slower than the other users, but cutting time improved quickly with experience. Even user 2, who had used the tool several times over the previous months, derived same benefit from cutting 3 blocks in the same day. Compaing to Figure 9-4, note that the decrease in cutting time did not affect user 3's cutting accuracy.

Remember that since the user decides when to stop cutting, cutting time is somewhat subjective. Some users spend more time than others "polishing" the workpiece to remove the last bits of waste bone.

9.1.3 User Velocity and Acceleration

The distributions of velocity and acceleration with which users moved the tool during the trials are shown in Figures 9-6 and 9-7. Many time samples are removed from this distribution. For the purpose of understanding cutting error and tuning the PFS algorithm, the only time samples that are important are those where some blade retraction action is necessary. Therefore, only time samples when the PFS was close enough to require retraction, i.e. 2mm from the target surface, are included in the distributions. At other times the tool may travel faster, particularly when it is lifted off the surface and moved to another position.

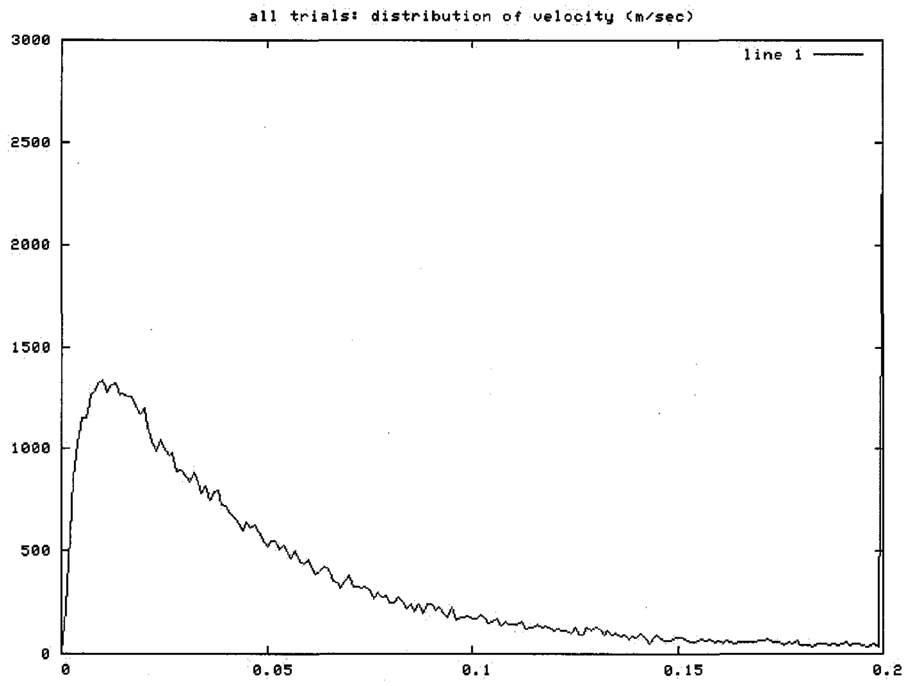


Figure 9-6. Distribution of velocity (m/s) in trials

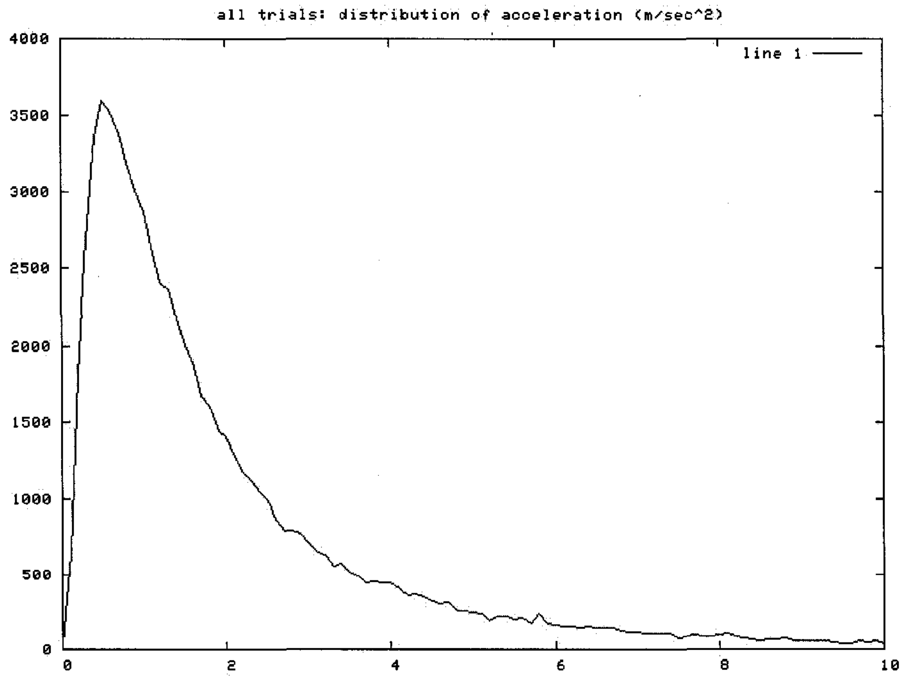


Figure 9-7. Distribution of acceleration (m/s^2) in trials

9.1.4 Position Prediction Error

Measuring position prediction error ε is important for preventing overcutting. The theoretical framework described in Chapter 8 demonstrated how, given a limit ε_{max} on ε, the extension multiplier α can be chosen to avoid overcutting. By measuring ε from the recorded data, we can inform the future choice of α.

To calculate ε, the tool position $x(t)$ was extracted from the recorded data and the one-step position prediction $x_{p1}(t)$ reconstructed based on the recorded Optotrak readings. $x_{p1}(t)$ is the prediction at time t for $x(t+1)$. ε was found as $| x_{p1}(t) - x(t+1) |$. Because ε is defined as distance parallel to the target shape, the component of $x_{p1}(t) - x(t+1)$ normal to the target surface was removed from this. The distribution of ε over all trials is shown in Figure 9-8. Additionally, the error in two-step position prediction, $\varepsilon_2 = | x_{p2}(t) - x(t+2) |$ over all trials was calculated and is shown in Figure 9-8 as well. As with the velocity readings, this distribution only reflects samples where the PFS was within 2mm of the target shape.

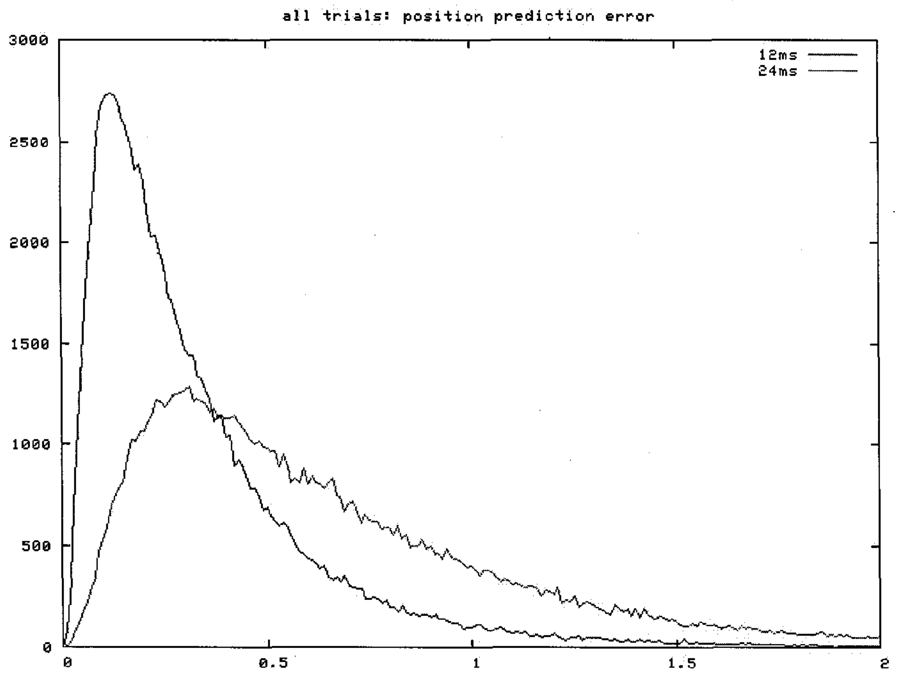


Figure 9-8: Position prediction error (mm) for one step (12ms) and two steps (24ms) into the future.

These ε measurements can suggest an appropriate α value to use in the future. (For the present experiments, the value of α=0.90 was chosen by rough intuition and direct observation of cutting results.) Choosing α requires determining ε_{max}, but the distribution in Figure 9-8 tapers off instead of having a clean maximum. ε_{max} cannot be chosen to exceed *all* observed ε, because that would cause the tool to be overly conservative and cut very inefficiently. On the other hand, every misprediction is a potential gash in the final surface, so ε_{max} should exceed most of the observed ε readings. A 95th or 98th percentile limit is not unreasonable. To select a value for the present application, ε_{max}=1.2 exceeds almost all ε readings, and corresponds to α=0.75, which will give reasonable cutting efficiency.

9.1.5 Typical Retraction Distances

In Chapter 8 we described how to estimate the required blade retraction speed based on the expected worksurface slope and user tool speed. Another way to estimate required blade retraction is by looking at how much the commanded blade retraction changed for each timestep in the trials. The advantage of using actual recorded data is that it accounts for the failure of assumptions used in Chapter 8.

From the recorded data we extracted $c(t)$, the commanded radial blade extension at each timestep. The distance that the blade must travel in one timestep is $c(t) - c(t\text{-}1)$. The distribution of $c(t) - c(t\text{-}1)$ over all timesteps in the trials is shown in Figure 9-9. Positive indicates that the blade was commanded to retract. For thoroughness, we also extracted $r(t)$, the amount that the blade is *allowed* to extend at each timestep, given the tool position with respect to the target shape. The distribution of $r(t) - r(t\text{-}1)$ is also shown in Figure 9-9, and is almost identical to the distribution of $c(t) - c(t\text{-}1)$.

Because a single instance of cutting error can leave a permanent gash in the bone, it is important that blade retraction be able to meet almost every required retraction in Figure 9-9. A good limit that encompasses almost all samples in Figure 9-9 is 0.2mm. To achieve this retraction in a single 12ms timestep requires a constant velocity of 16mm/s. This is significantly slower than the 65mm/s – 85mm/s requirement given in Chapter 8, which is something of a worst-case estimate given maximum workpiece slope and maximum tool speed.

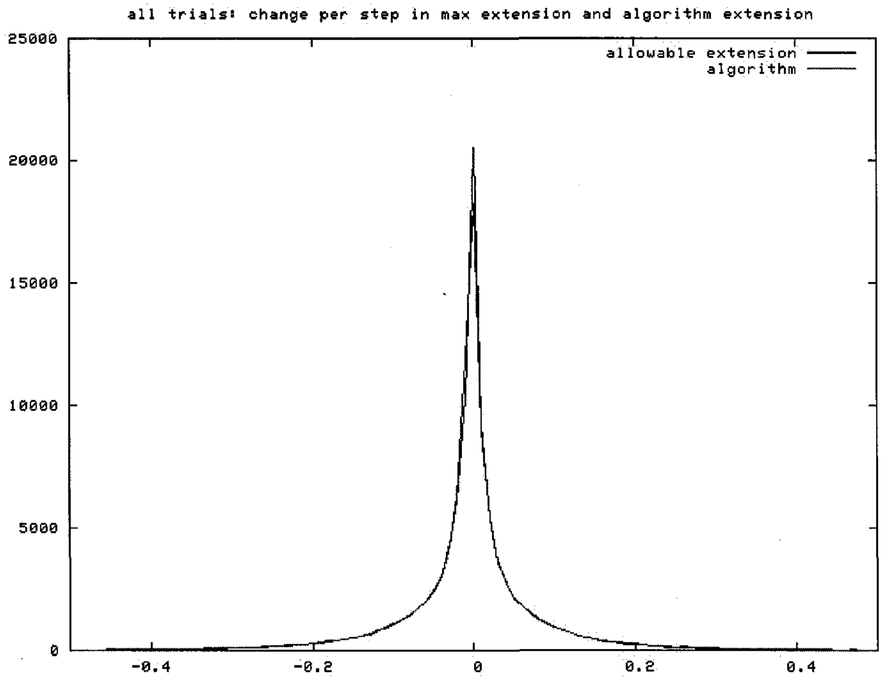


Figure 9-9: distribution of c(t)-c(t-1) ("algorithm") and r(t)-r(t-1) ("allowable extension"). They are practically identical.

9.1.6 Blade Response and Latency

The data recorded in these experiments offers a good opportunity to verify and refine the dynamic model for the retraction motor measured in Section 5.4. We will focus on radial retraction because most of the cutting in these experiments was done in the radial direction. The commanded radial blade extension c(t) and the measured blade extension b(t) for each timestep t were extracted from the recordings. Based on c(t), and using the acceleration and top speed measured in Section 5.4, a simulated blade position $b_{sim}(t)$ was computed. When b(t) was plotted along with $b_{sim}(t)$, b(t) lagged. This is expected, since the command c(t) is not issued at the start of the software frame, but only after the algorithm has computed it based on the Optotrak data.

To estimate the delay between the start of the frame and when the retraction command is executed, a delay was added to the dynamic model, and adjusted by hand so that $b_{sim}(t)$ lined up with b(t). Figure 9-10 shows a sample time interval from one trial. The tuned latency was 16ms, which is surprising because the software frame only lasts 12ms, and the command is certainly issued before the start of the next software timestep. This suggests some communication lag between the PC and the motion controller, or a delay within the motion controller itself. The exact cause has not been investigated. The effect of this delay on cutting accuracy will be examined in Chapter 10.

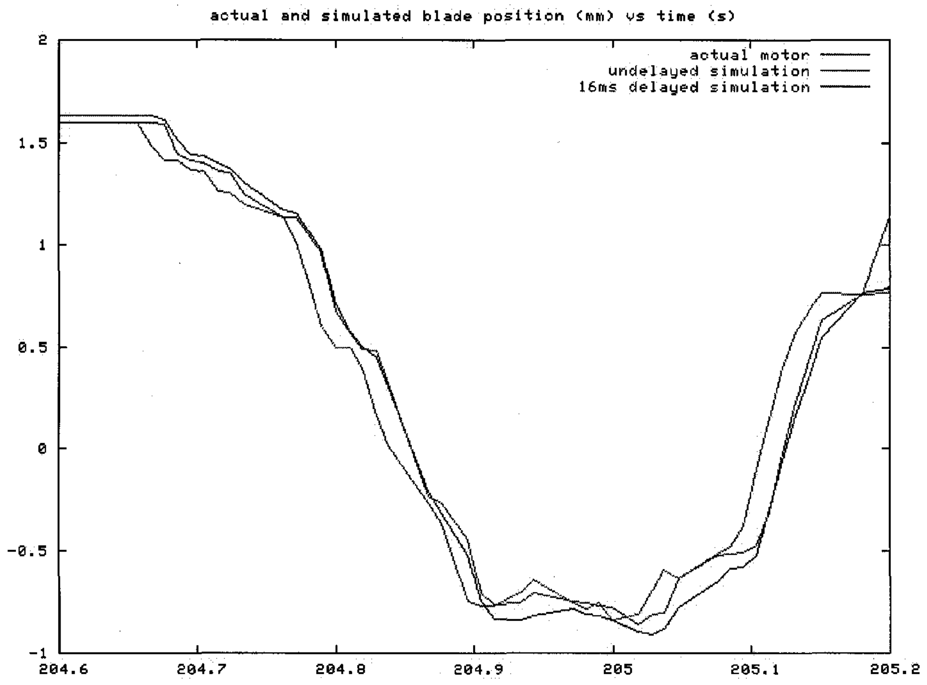


Figure 9-10: Simulated motor position matches actual only with 16ms delay from start of frame. Example time interval from trials. x-axis is

9.2 Small Incision Experiments

The second series of tests was designed to measure cutting performance when the tool must operate through a small incision. Sawbones femurs were wrapped in a sheet of flexible foam with a small incision through it, and the UKR shape was cut with the PFS. To isolate variables, Sawbones were also cut without the flexible foam sheet, and foam blocks were cut through an incision in the flexible foam sheet. In summary, the follwing conditions were used:

Sawbones Open: The UKR shape was cut on Sawbones femurs.

Sawbones Covered: The UKR shape was cut on Sawbones femurs that were wrapped in a sheet of flexible foam with a small incision in it.

Foam Blocks Covered: The foam blocks setup from the first round of experiments was wrapped in a sheet of flexible foam with a small incision in it, and the UKR shape was cut.

I (User 1) performed 3 cuts under each condition. We will compare these results with the User 1 results from the original foam block experiments, which we will refer to as "**Foam Blocks Open**".

The experimental setup was as described in Chapter 4, but for compatibility with the foam block experiments, only the femur was cut. A sawbones femur and tibia joined by elastic ligaments were used. A tracking marker was attached to the femur and point-based registration was used to register it to the original CT scan. For covered tests, the bones were then covered with a sheet of flexible foam and an incision was cut throough the foam.

For the covered block tests, the foam sheet was stapled to the jig that held the blocks.

All cut blocks and femurs were laserscanned and accuracy calculated as before.

9.2.1 Measured Accuracy

The distribution of cutting error for each condition is shown in Figure 9-11. The residual RMS errors are:

Block open:	0.15mm
Block covered:	0.19mm
Sawbones open:	0.16mm
Sawbones covered:	0.20mm

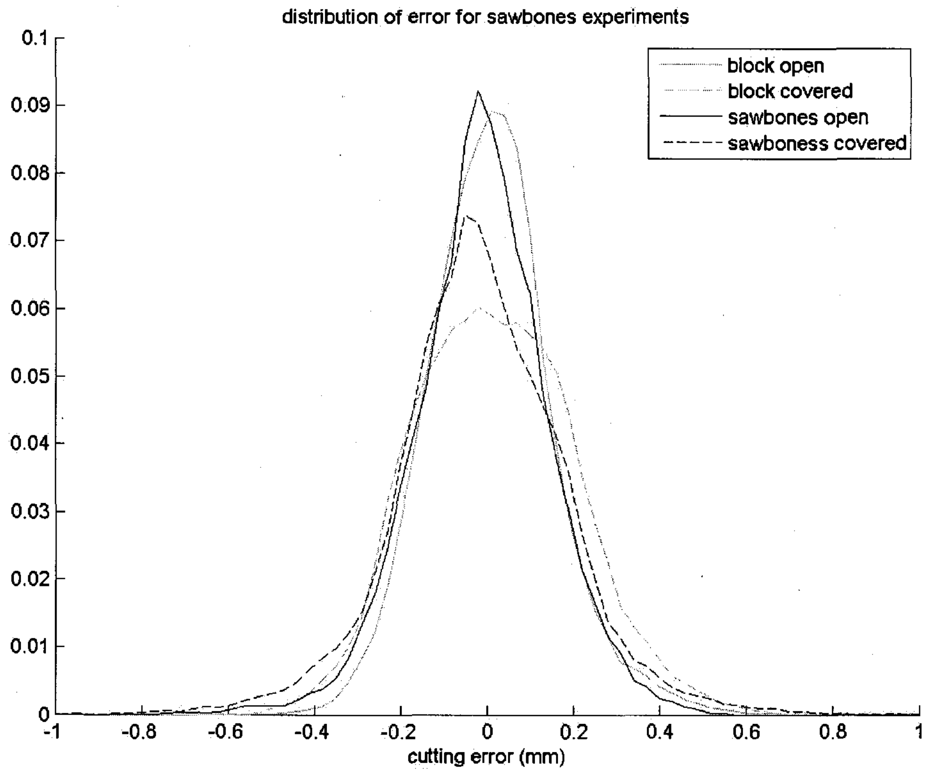


Figure 9-11: Distribution of error for Sawbones experiments.

9.2.2 Cutting Time

Cutting time for the sawbones trials was:

Sawbones open:	4:45	2:55	2:53
Sawbones closed:	6:16	4:13	7:01

9.2.3 User Velocity and Acceleration

The distribution of tool velocity (measured at the cutter) seen in the trials is shown in Figure 9-12. The velocities in the covered experiments were slightly slower than those in the open experiments.

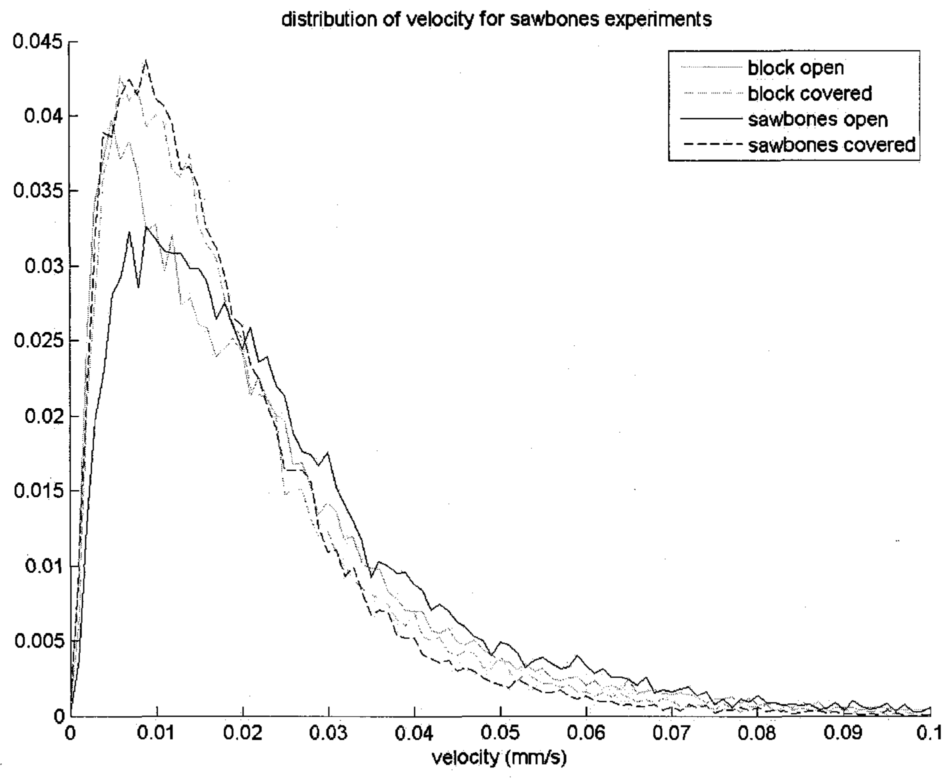


Figure 9-12: Distribution of tool velocity for Sawbones experiments.

The distribution of tool acceleration seen in the trials is shown in Figure 9-13.

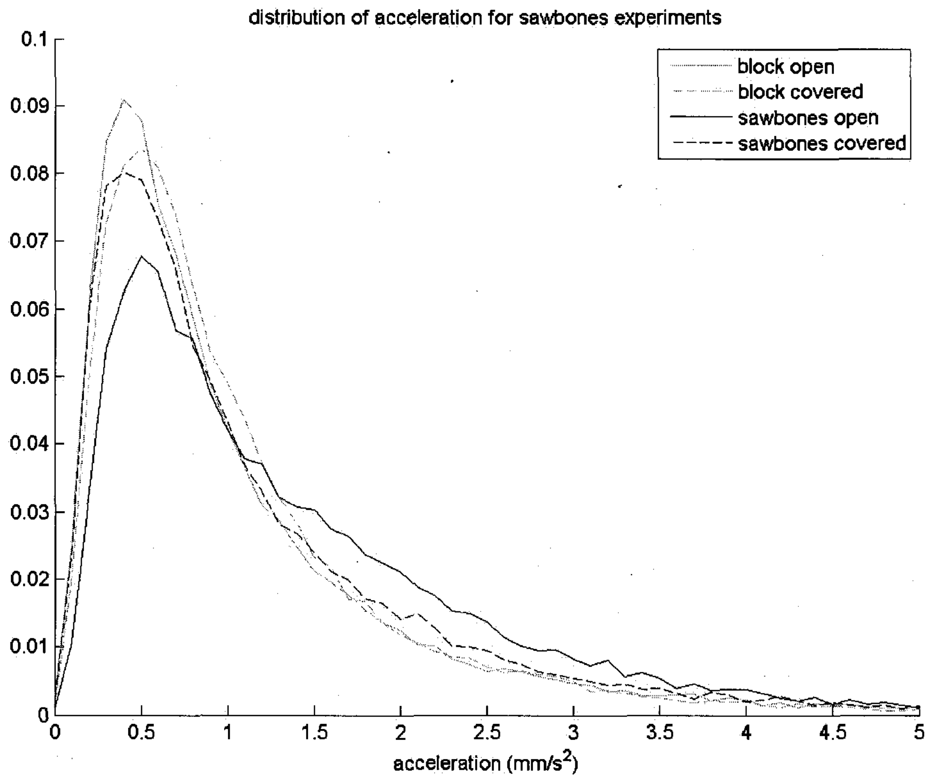


Figure 9-13: Distribution of tool acceleration for Sawbones experiments.

9.2.4 Position Prediction Error

The distribution of ε, the error in predicting tool positions one timestep ahead, is shown in Figure 9-14. ε was worse for both open and covered Sawbones than it was for open or covered blocks.

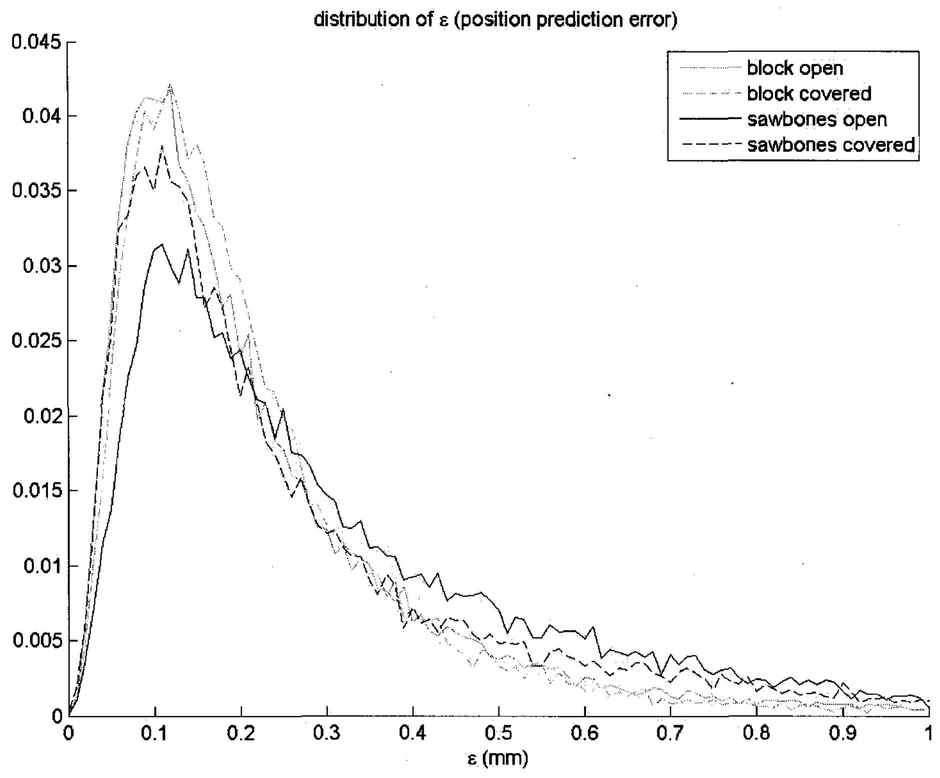


Figure 9-14: Distribution of ε (position prediction error) for Sawbones experiments.

9.2.5 Typical Retraction Distances

The distribution of change in allowable blade extension for single timesteps is shown in Figure 9-15. Slightly larger changes in extension were required for the Sawbones experiments compared to the original block trials.

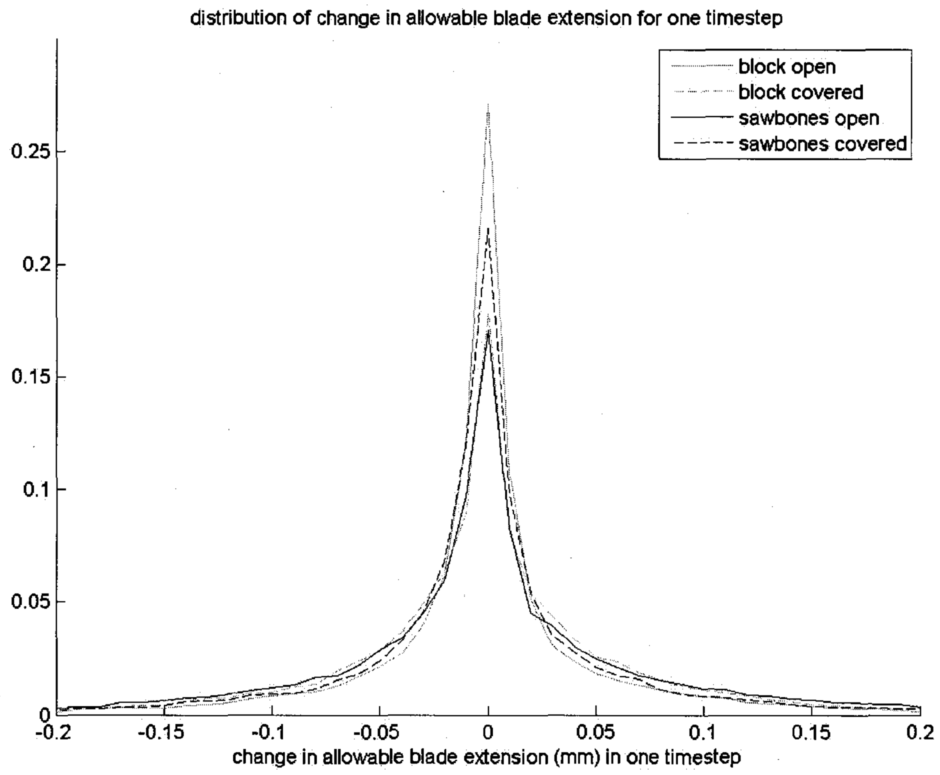


Figure 9-15: Distribution of change in allowable blade extension for one timestep for Sawbones experiments.

9.3 Practical Experience Gained

The use of the tool in these experiments, and earlier experience, has resulted in some practical recommendations about how to use the tool. These are listed here.

Slow and Controlled: Orthopedic surgeons tend to use conventional burs in a quickly sweeping, side-to-side motion. This prevents the bur from digging in and helps to produce a level surface. The PFS, on the other hand, should be used in a slow, controlled fashion, typically in the conventional-cutting direction. First of all, this gives the PFS more time to retract and extend the blade properly. Secondly, it allows for the most aggressive material removal rate. The guard prevents the tool from digging in, allowing the surgeon to make a very aggressive but controlled cut.

Start at the Front: The PFS should be used starting at the area of the workpiece closest to the user and working back from there. This prevents the rear sections of the guard from resting on the workpiece and preventing the tip from cutting.

Cutting the Tibia: The tibia can be difficult to cut because of the interior corner. By experience, we found that an effective technique is to rotate the guard to expose the blade to the corner, even during the beginning cuts. If the surgeon tries to flatten the main area of the tibia without rotating the guard, a large "uncuttable" area builds up extending from the side wall of the tibial cut. If the user does not rotate the guard, the guard rests on the uncut side wall and prevents the user from cutting the area next to it. This effect compounds to create quite a large "uncuttable" area extending from the corner. This was a cause of some frustration in practice. Rotating the guard appropriately alleviated this problem.

Keep the Tip Down: Rather than trying to keep the guard flat to the working surface, it may be helpful sometimes to tip the guard forward slightly so that the tip of the PFS definitely makes contact. By replaying cases from recorded data, we noticed some occasions when the user appeared to be attempting to remove some small remaining material, but the rear of the guard was on the surface, preventing the tip from reaching the waste material. This was especially prevalent on the posterior femoral cut. Intentionally tipping the tool so that the tip definitely makes contact could eliminate this problem.

9.4 Bug Discovered

After the trials were complete a problem was discovered with how the snapped-to-surface positions are adjusted by the prediction algorithm. Normally, if the current position of the guard is inside the surface, then the predicted positions will be snapped to also rest that far inside the surface. During the trials, this same logic was applied when the guard was perceived to rest above the surface. That was wrong, because adjusting the predicted positions away from the bone surface will allow the calculated extensions to cause overcutting if the tool actually moves into contact with the bone surface. This bug may have caused additional cutting error during the trials.

To estimate the effect of this mistake on cutting error, we extracted the radial blade position b(t) from the recorded data. We also extracted r(t), the maximum amount that the blade may extend in the radial direction without overcutting. Cutting error is b(t)-r(t). If this quantity is positive, the tool overcut at timestep t. We then recalculated b(t) with the bug fixed, based on the recorded Optotrak data, and called it $b_{fix}(t)$. Figure 9-16 shows the distributions over all timesteps of b(t)-r(t) and $b_{fix}(t)$-r(t). Positive samples indicate that the tool overcut. Looking at these two distributions in the region greater than zero, it seems that bug caused only a very minor increase in the frequency with which the tool overcut.

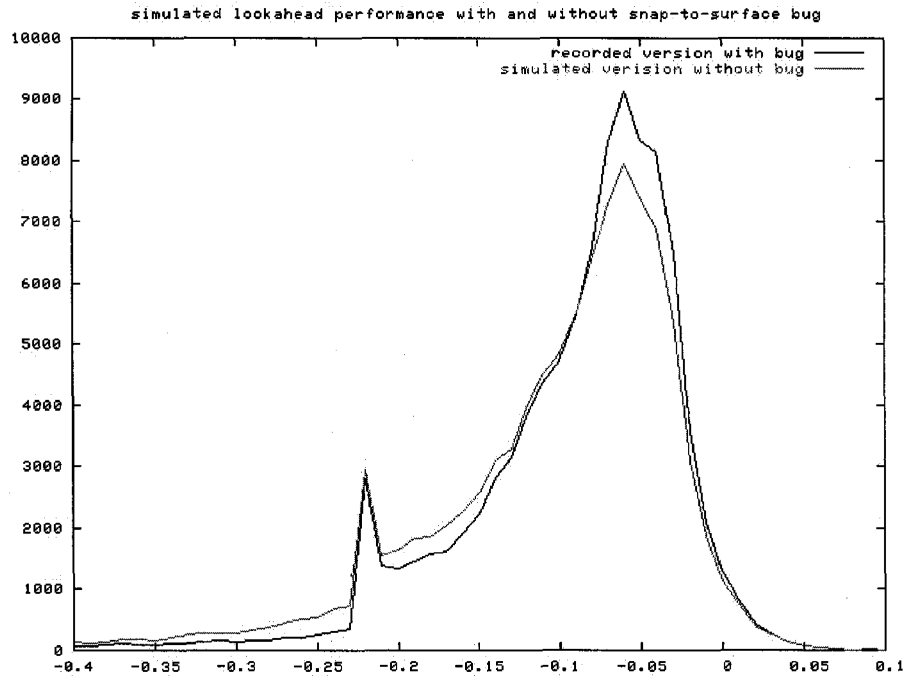


Figure 9-16: Simulated distribution of cutting error with and without bug. Positive indicates overcut.

Chapter 10. Analysis of Cutting Error

Cutting accurately enough is the biggest technical challenge the PFS faces. Although current accuracy is promising, the chances of successful patient outcomes can certainly be increased by further improvements. To improve accuracy, we enumerated the potential sources of cutting error, and attempted to identify the contribution of each to total cutting error. By estimating the contribution of each error source, we can identify the largest sources of error so that future development work can be focused most efficiently.

Enumerating and studying the sources of cutting error can improve our fundamental understanding of the PFS. This understanding may illuminate new paths to improvement and can help with future design of PFS tools. It can indicate which design factors require special attention to achieve the desired accuracies, and which are more forgiving.

Figure 10-1 illustrates the list of potential error sources we enumerated. Some error sources are hierarchally broken down into individual error components.

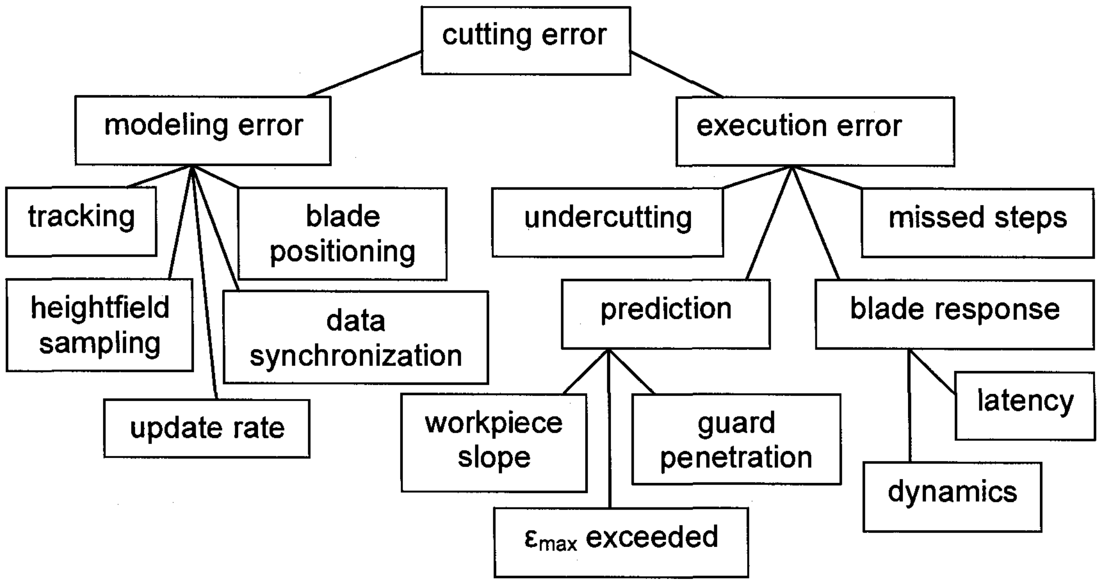


Figure 10-1. Taxonomy of Error

Cutting error can first be broken down into two categories: the difference between the target shape and the computer model, and the difference between the computer model and reality. We refer to the former as *execution error* and the latter as *modeling error*. Effectively, execution error is error the computer is aware of but was unable to prevent, and modeling error is error the computer is not aware of. (Figure 10-2) Note that total cutting error is the difference between the target shape and the actual surface, so that

$err_{total} = err_{modeling} + err_{execution}$

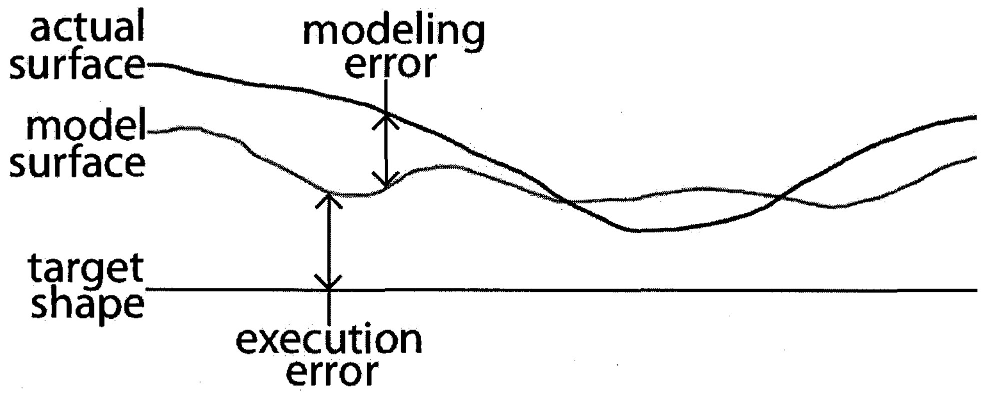


Figure 10-2. Cutting error is broken down into modeling error and execution error.

Execution error for a trial can be read directly from the final worksurface model at the end of the trial. The execution error at each point in the model is simply the height of the heightvector at that point. The distribution of execution error from the block-cutting trials is shown in Figure 10-3.

Modeling error is the difference between the final worksurface model and the laser scanned actual surface. Just as when total error was extracted from the scans in Chapter 9, extracting modeling error from the scans requires a registration of the scan to the workpiece coordinate frame. To measure modeling error, we did not use the registration calculated in Chapter 9, because it is not a ground-truth registration, and could artificially increase the measured modeling error. Rather, we used a least-squares fit between the scan points and the heightfield model. The distribution of modeling error for the block-cutting trials is shown in Figure 10-3.

Modeling error and execution error were calculated for each foam block trial. The heightfield contained 3187 points, and the scans contained about 15,000 points. The distribution of modeling, execution, and total error over all points on those trials is plotted in Figure 10-3. Positive total error indicates that the tool cut too far. Positive modeling error indicates that the workpiece was cut further than the model reflected. Positive execution error indicates that the model showed the tool had cut too far. (This may confuse, because positive execution corresponds to a negative height for the heightvector, but it is more consistent with our definitions of modeling and total error, and allows us to write $err_{total} = err_{modeling} + err_{execution}$.)

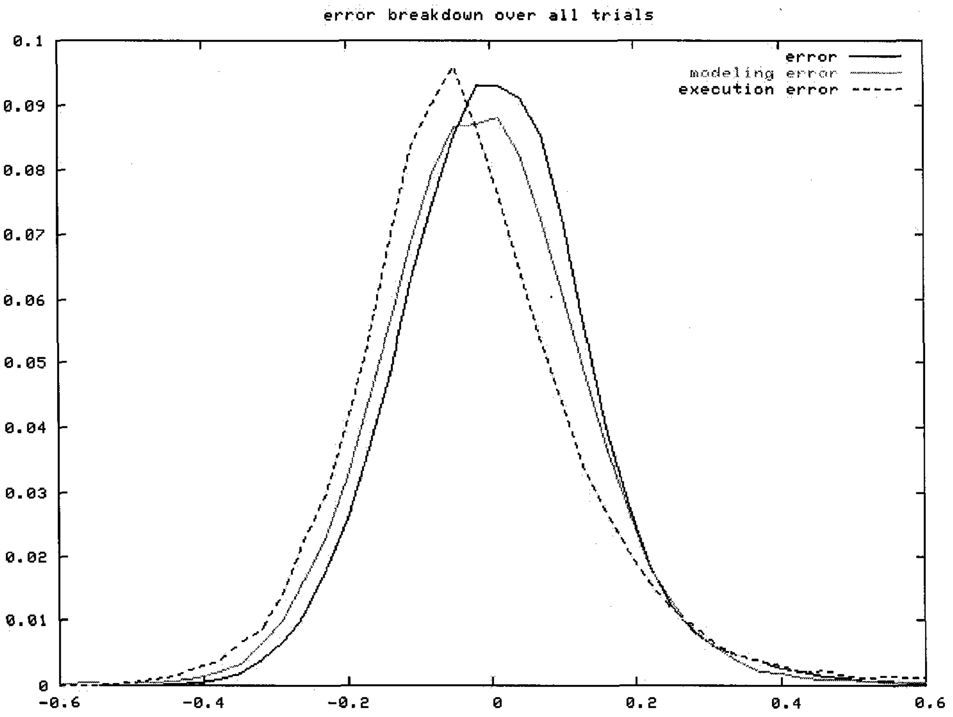


Figure 10-3. Distribution of modeling, execution, and total error, in mm.

Since error is the sum of modeling and execution error, it is very surprising that the distribution of total error is not much wider that the distributions of execution and modeling error. Two effects are at work here: First, because separate least-squares fits are used for modeling error and total error, the scan can make a closer fit with the model than if total error were explicitly measured as modeling error plus execution error. Second, there are characteristics of the computer model that tend to accentuate errors, while characteristics of the actual tool tend to smooth out the actual surface.

The phenomena which accentuate errors in the model and which smooth the actual surface are separate effects. The actual tool smooths the surface because the guard very accurately references cutting to the surrounding areas. In the final stages of cutting, the PFS blade extends only a tiny bit beyond the guard. Since the guard cannot penetrate the surface, the blade cuts every point only to the level of the surrounding surface. This tends to flatten the surface outside of any active control by the computer. Further, this allows the tool to reference the existing flat surface much more accurately than Optotrak readings can inform the computer.

At the same time, the computer model tends to exaggerate roughness in the surface. If a single erroneous Optotrak reading shows the blade inside the workpiece, the result is permanently recorded in the heightfield model.

Figure 10-4 illustrates clearly the circumstances by which total error can be less than the sum of modeling and execution error. The figure maps total, execution, and modeling error for user 1, trial 2. Points where the surface was cut more than 0.2mm too far are indicated by an 'x', and points where the surface was undercut more than 0.2mm are indicated by a '+'. Figure 10-4a shows total error, based on the fit between the target shape and scan surface. Figure 10-4b shows execution error, based on the heights of the vectors in the heightfield. The sample density is lower because the heightfield has many fewer points than the scan. Execution error shows considerably more variation than total error, indicating that the model surface is rougher than the actual workpiece surface. The oval-shaped indentations in the model match the shape of the cutter. Such an indentation could be caused by a single erroneous Optotrak reading that showed the blade inside the target surface. Figure 10-4c shows modeling error. Even though the actual workpiece surface was smooth, modeling error is rough because the fit between the model and the actual surface is rough.

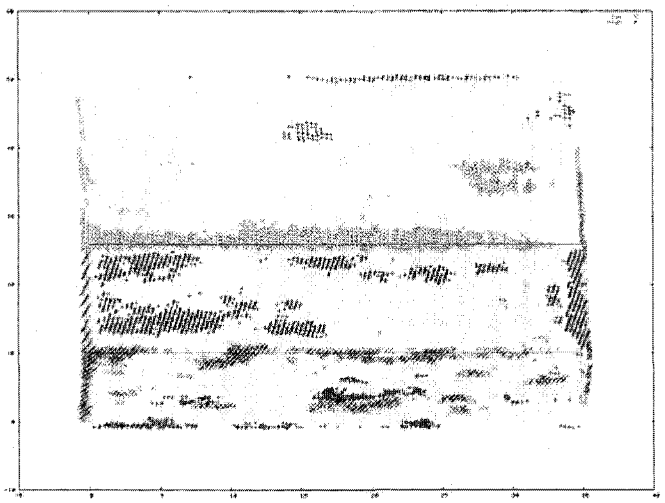


Figure 10-4a. Total cutting error for user 1, trial 2.

'x' indicates overcut by > 0.2mm

'+' indicates undercut by > 0.2mm

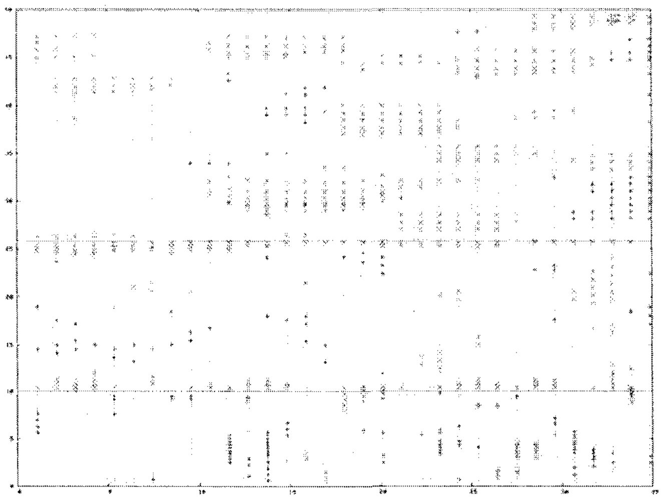


Figure 10-4b. Execution error for user 1, trial 2.

'x' indicates overcut by > 0.2mm

'+' indicates undercut by > 0.2mm

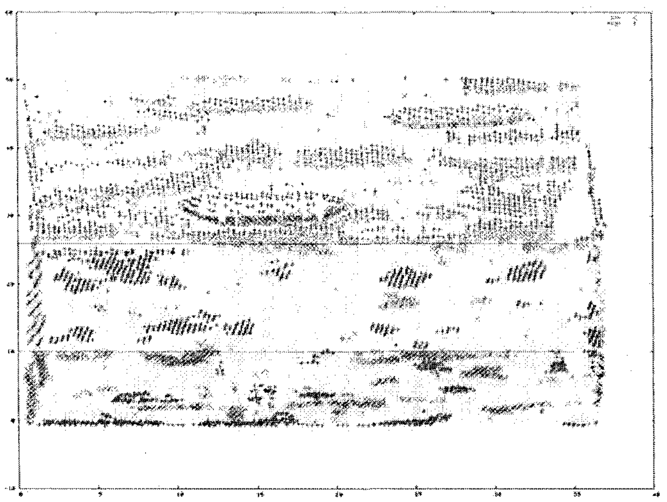


Figure 10-4c. Modeling error for user 1, trial 2.

'x' indicates actual cut further than model by > 0.2mm

'+' indicates actual cut less than model by > 0.2mm

Modeling error shows gashes that are apparent in execution error, but not total cutting error.

10.1 Analysis of Execution Error

Execution error is error that is reflected in the computer's sensor readings, but that the computer was unable to prevent. Figure 10-3 depicted execution error in aggregate. In this section, we will attempt to identify the component sources of execution error, and to measure the influence of each individually. In Figure 10-5, we identify several potential causes of execution error. Execution error can be primarily broken down into the failure of the software to predict proper blade retraction (prediction error), and failure of the blade to achieve the predicted blade retraction within the allotted time (blade response error). In this PFS implementation, errors can also be caused when the software skips an Optotrak data frame because of timing constraints. Finally, any waste material not removed when the user stops cutting is in error. Although it is the user's responsibility to remove this material, the PFS system needs to make waste removal as easy as possible.

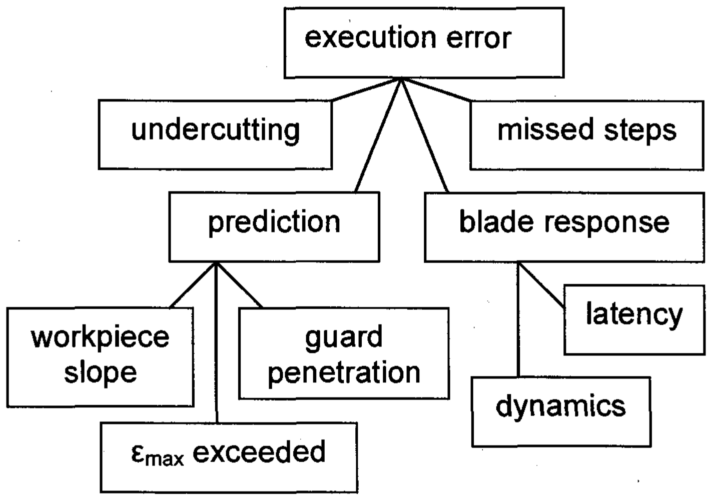


Figure 10-5. Potential sources of execution error.

The boundary between prediction error and blade response error is fuzzy, because better prediction can make up for slower retraction speed, and vice versa. The minimum requirement for prediction is to predict one step ahead, to compensate for limited Optotrak update rate. Our analysis in this chapter will concentrate on this basic requirement for prediction, and define failure to meet the one-step prediction as an error in blade response.

Since execution error is reflected in the computer model, much can be learned about its causes by studying the data recordings from the trials. A recording can reveal whether the tool overcut at each timestep, and can provide clues as to why. The following variables can be extracted from the data recording and will be useful for our analysis:

$b(t)$ is the actual blade extension at the start of timestep t.
$r(t)$ is the required maximum blade extension at t.
$p_i(t)$ is the predicted allowable blade extension for i steps ahead, i.e. predicted $r(t+i)$.
$c(t)$ is the commanded blade extension issued at t. This is based on $r(t)$ and $p_i(t)$.

At the beginning of each software cycle t, the computer samples the blade position $b(t)$ and Optotrak data. From the Optotrak data the computer calculates $r(t)$ and $p_i(t)$, and then issues blade retraction command $c(t)$.

The execution error at any timestep can now be expressed simply as

$$err_{exec}(t) = b(t) - r(t)$$

However, some clarifications are required first. In reality, $b(t)$, $r(t)$, and $p_i(t)$ cannot be given only a single value – each has a different value for each candidate extension direction. For the sake of analysis, we will consider only one extension direction. The analysis could easily be repeated for other extension directions.

We have chosen to consider the radial direction only, because it is the primary cutting direction for the recorded cutting experiments. For the radial direction, the values for $r(t)$, $p_i(t)$, and $c(t)$ can be extracted from the data recordings. $r(t)$ and $p_i(t)$ are the allowable extensions in the radial direction for the current and future timesteps. $c(t)$ is the allowable extension in the radial direction based on dynamic constraints. On the other hand, $b(t)$ cannot be extracted directly from the recorded data, because the actual blade was not moving under the radial position command *c*(*t*), but under a position command influenced by all the candidate extension directions. To remedy this, *b*(*t*) was simulated using the dynamic model developed in the previous chapter.

Another issue is the choice of *r*(*t*), the maximum allowable blade extension which is used for calculating execution error. Recall that to calculate the maximum extension allowed by the current position, the algorithm first adjusts the tool position to rest on the workpiece surface using snap-to-surface. This suggests two possible definitions for *r*(*t*), as in Figure 10-6: the allowable extension based on the actual position of the tool, $r_{nosnap}(t)$; or the allowable extension based on the snapped position, $r_{snap}(t)$. For safety, we want to ensure that $r_{snap}(t)$ is always satisfied. Hence we will use $r_{snap}(t)$, the more restrictive definition.

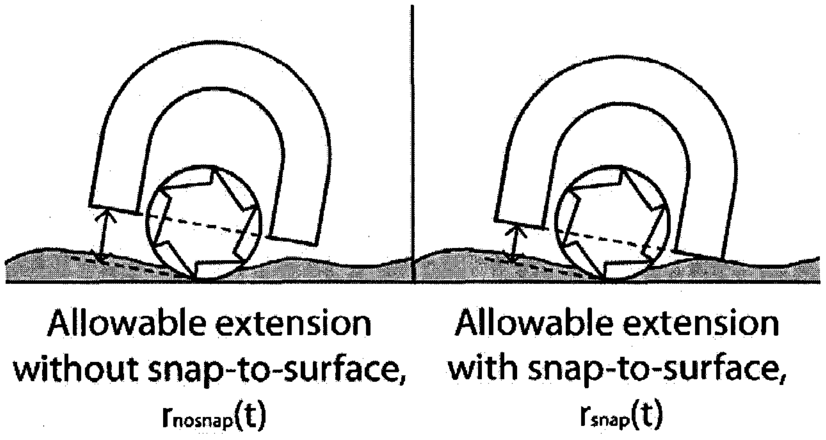

Allowable extension without snap-to-surface, $r_{nosnap}(t)$
Allowable extension with snap-to-surface, $r_{snap}(t)$

Figure 10-6: $r_{nosnap}(t)$ versus $r_{snap}(t)$. Cross-section of blade and guard is pictured. Waste material shown in grey.

The variables *b*(*t*), *r*(*t*), $p_i(t)$, and *c*(*t*) can now be used to examine execution error. The execution error at any timestep is

$$err_{exec}(t) = b(t) - r(t) \qquad \text{Equation 10-1}$$

which is cutting error that the computer is aware of: the difference between the allowable blade extension based on the Optotrak, and the sensed blade extension.

Figure 10-7 shows the distribution of $err_{exec}(t)$ over all timesteps in all trials. Positive indicates that the PFS overcut.

Certain timesteps were discarded in Figure 10-7, to isolate only timesteps where some response from the PFS was necessary to avoid overcutting. All timesteps where the allowable extension was 2mm or more were discarded. Also, timesteps when the distance between the guard and the workpiece surface was over 0.5mm were discarded, to remove cases when the tool is not resting on the bone, which is usually because the user is moving the tool from one pass to the next, rather than cutting. These removed samples far outnumber the included samples, and would overwhelm the distribution, if included. The peak in Figure 10-7 at -0.2mm is an artifact of some samples that should have been removed, but narrowly missed the thresholds for removal.

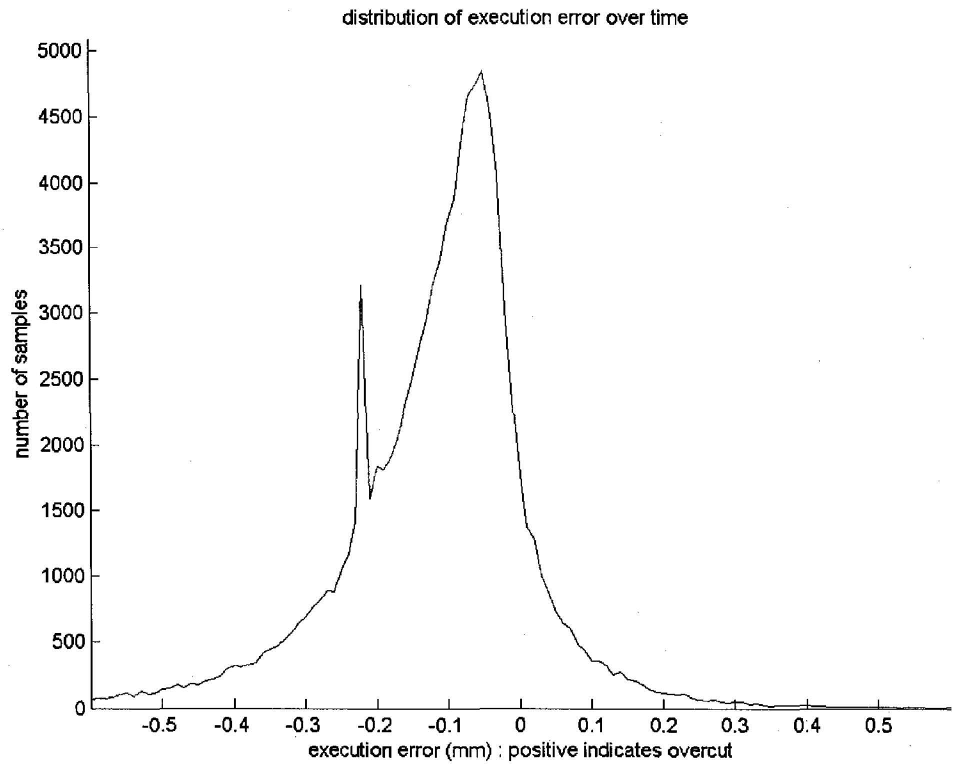


Figure 10-7. Distribution of execution error over time.

Figure 10-7 shows a distribution over time, as opposed to the plot of aggregate execution error (Figure 10-3), which showed the distribution over the area of the target shape. Unfortunately, no direct correspondence can be drawn between these two distributions: Multiple time samples featuring overcut may all occur at the same position. Further, time samples where the tool undercuts (execution error < 0) do not indicate cutting error at all. Undercutting in the spatial distribution is caused by waste material that remains when the user decides to stop cutting.

Although we cannot attribute an actual amount of final cutting error to a given error in time, the distribution of cutting error over time is still a useful tool for evaluating the relative magnitude of error sources. By comparing the distribution over time of each component of execution error, we can estimate each component's contribution to execution error. We know from the spatial distribution that execution error makes up about half of total cutting error, so we know that the largest sources of execution error are also significant contributors to total error.

When comparing the time distribution of error sources, the real effect on the final spatial distribution of error comes not the vast number of timesteps when the tool overcut slightly or not at all; but the small number of times when the tool overcut significantly. The magnitude of execution error is the amount that the blade violates the target shape, so many small overcuts will never do as much damage as one deep overcut. Each instance when the tool overcuts significantly leaves a large, permanent gash in the target shape. The regular histogram is not a good tool for locating these rare instances – they are spread throughout the tail of the distribution.

A better tool to highlight instances in the time distribution of large overcuts is the reverse cumulative distribution, which we will use for comparing sources of execution error. The reverse cumulative distribution of total execution error is shown in Figure 10-8. The height of the curve is the number of samples where execution error was at least as large as the value on the x-axis. Only the positive execution error part of the plot is shown. To best represent execution error sources for comparison, in the following sections we will use the reverse cumulative distribution instead of the plain histogram when examining the time distribution of error.

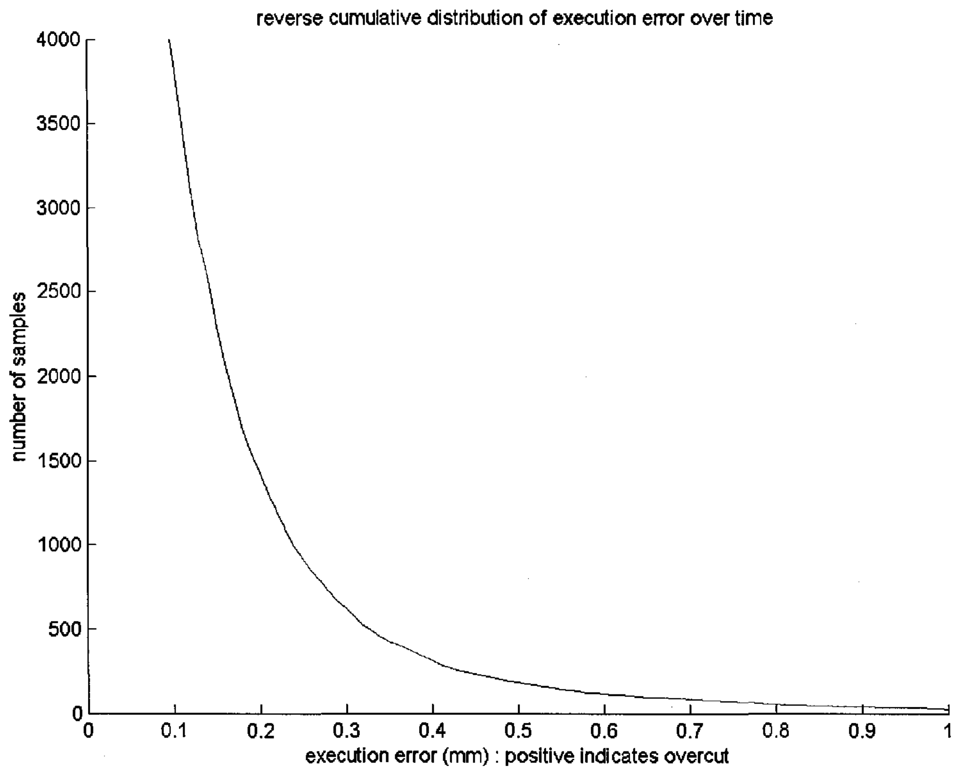


Figure 10-8. The reverse cumulative distribution of $err_{exec}(t)$ highlights the rare instances of large amounts of overcut.

10.1.1 Error due to Missed Optotrak Frames

Missed software frames are one potential source of error in this PFS implementation. The blade control algorithm runs once each frame of Optotrak data. If a frame is missed, the blade position will not be updated until the next frame of Optotrak data is reported. Naturally, this doubles the PFS response time and can introduce execution error.

There are two primary causes for missing data frames in the PFS implementation. First, the algorithm can simply run over time. In the trials, about 10% of frames took too much computation time and missed the next Optotrak frame. Second, OpenGL rendering causes the algorithm to skip two consecutive frames out of every 9. Therefore, about 11% of frames were preceded by two missed frames.

To test the effect of missed software frames on execution error, each sample in the recording was sorted based on whether it was preceeded by a gap of zero, one, or two timesteps. We refer to these classes of frames as singles (no missed steps), doubles (one missed step), and triples (two missed steps), respectively. The reverse cumulative distribution of execution error that occurred for each class is plotted in Figure 10-9. Since there are a different number of samples in each class, the curves are normalized, i.e. the vertical axis shows proportion of samples out of 1 total.

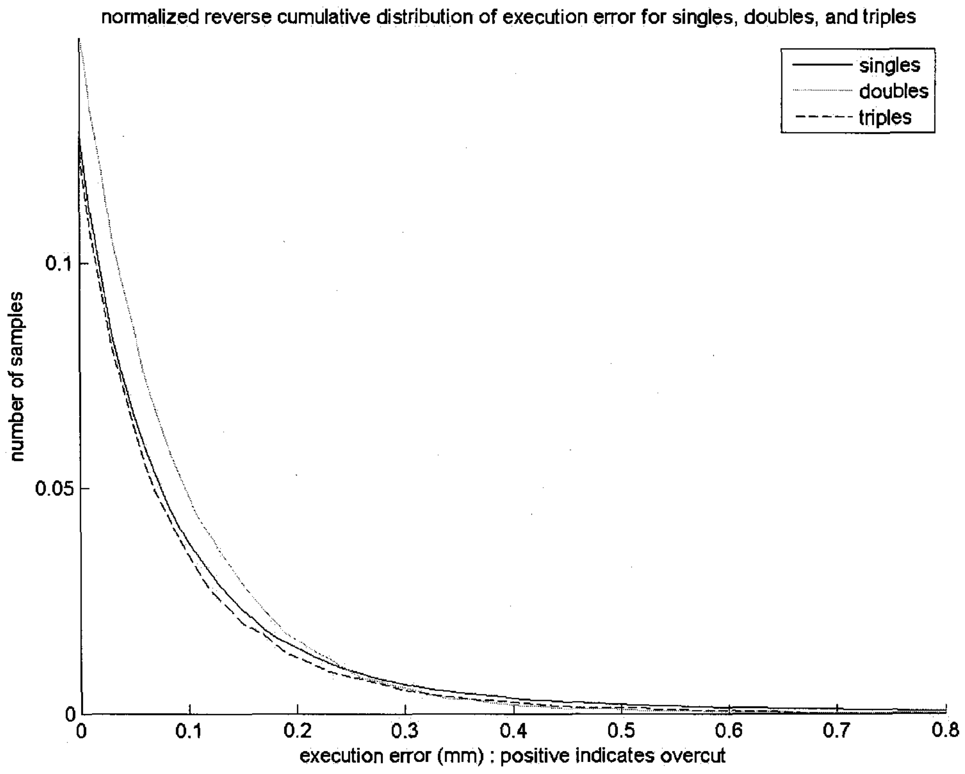


Figure 10-9. Distribution of execution error for singles (no missed software frames), doubles (preceded by 1 missed software frame), and triples (preceded by two missed software frames). The difference between the curves is minor.

This plot indicates that the error profiles for doubles and triples were almost identical to the error profile for singles. Therefore we conclude that missed software frames were not a significant source of execution error.

However, since doubles and triples may have an effect in correlation with certain error sources, we will exclude them from the analysis below. We expect that in the future, doubles and triples will be eliminated. Below we will examine the contribution to error of other potential execution error sources, and the distributions we plot will include only the samples that were not preceded by missed Optotrak frames.

10.1.2 Error due to Prediction

Accurate prediction is the first step towards avoiding execution error. The algorithm must *predict* how far the blade will need to retract, so that the retraction motors can be given adequate time to meet the retraction requirement. At minimum, the algorithm should predict how far the blade must be retracted by the beginning of the upcoming timestep. In this section we will focus on that minimum requirement. To further compensate for blade retraction speed, prediction can be extended further into the future: The analysis is similar.

Our goal for prediction, then, is for the command $c(t)$ to accurately predict the extension requirement $r(t+1)$ at the beginning of the following timestep. If $c(t)$ exceeds $r(t+1)$, then the command has overestimated how far the blade may extend, and may result in cutting error. We label this difference as the error due to prediction:

$$err_{pred}(t) = c(t\text{-}1) - r(t) \qquad \text{Equation 10-2}$$

Figure 10-10 illustrates the reverse cumulative distribution over time of $err_{predict}(t)$ as compared to $err_{exec}(t)$.

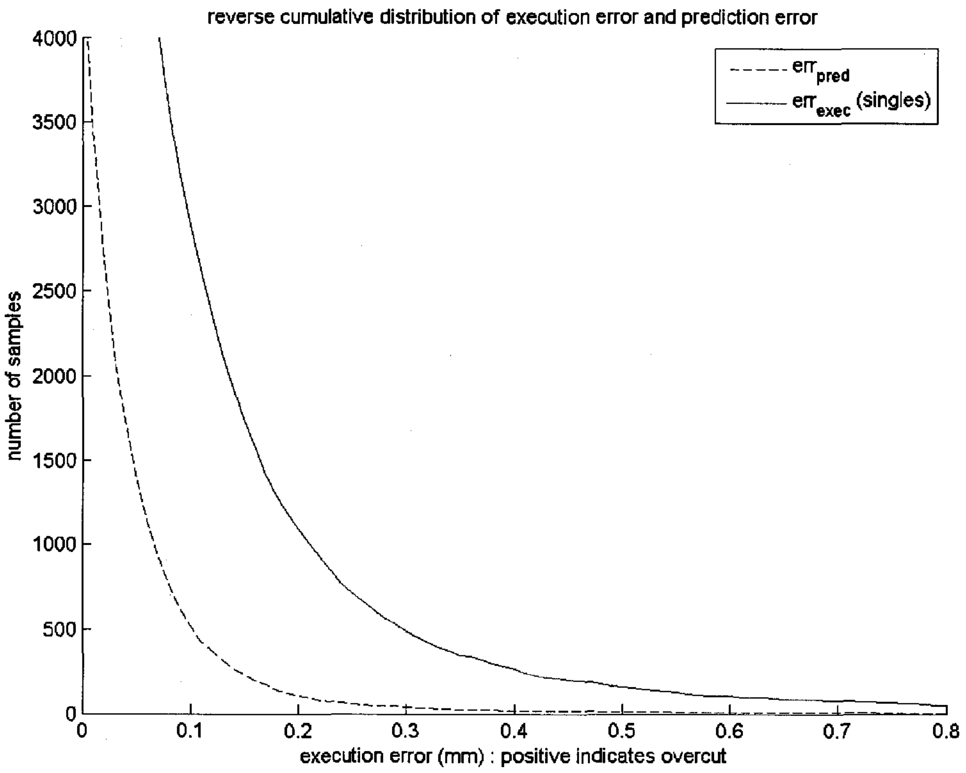


Figure 10-10. Distribution of err_{pred} compared to total err_{exec}.

The plot shows that prediction error makes up a small, but still significant, part of execution error. To understand and improve prediction error, we will examine the causes of prediction error below and estimate the contribution of each.

Causes of Prediction Error

To understand the sources of prediction error, we will use the framework described in Chapter 8. Given a limit ε_{max} on error in predicting tool *position*, the PFS can choose an extension multiplier α so that the commanded blade extension at each timestep will be less than the maximum allowable blade extension at the following timestep. We warned that this framework is an approximation, because it is based on assumptions that cannot be relied on. Prediction error is the result of failure in these assumptions. In this section we will examine when the assumptions fail and what the effect of each on prediction error is.

To use the framework described in Chapter 8, we must first extend the simple 2D model used in that chapter to full 3D. The original 2D model (Repeated here as Figure 10-11) included x, the position of the tool in the horizontal direction, ε, the error between predicted and actual x, and $y(x+w)$, the height of the workpiece surface on which the tool rests. In Chapter 9, we extended the definition of ε to 3D as the difference between the predicted and actual tool positions, measured from the center of the front sphere of the cutter's nominal position. The component of this distance normal to the target surface was removed.

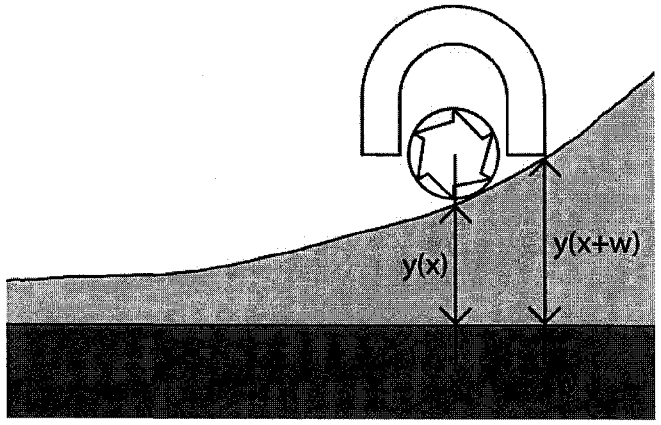


Figure 10-11. Original 2D model used to model PFS cutting process. For use analyzing cutting error, it must be extended to 3D.

In extending $y(x+w)$, the waste material thickness, to 3D, the question is at what point to measure the waste thickness. We will use the snap-to-surface calculations to find the waste thickness at the point where the tool actually rests. We define $y(x+w)$ as the height of the heightvector used by the snap-to-surface routine. That heightvector represents the point of first contact when the tool guard is translated directly towards the target shape, and the height of the heightvector represents the waste thickness at that point. Since y is now parameterized by timestep t instead of position x, we will change our notation slightly. Equation 8-4 refers to $y(x+w)$, the actual waste thickness under the tool at a timestep t, and $y(x+w+\varepsilon)$, the waste thickness under the tool at timestep t as predicted in timestep t-1. We will refer to these as $y_{real}(t)$ and $y_{pred}(t)$. When extended to 3D, Equation 8-4, which was:

$$err = \alpha\, y(x+w+\varepsilon) - y(x+w)$$

now becomes

$$err_{model}(t) = \alpha\, y_{pred}(t) - y_{real}(t)$$

This err_{model} is an approximation of what we now call err_{pred} (Equation 10-2). In the 2D model, we assumed that the tool remained horizontal, so that $y_{real}(t)$ was the allowable blade extension, which we now call $r(t)$, and $y_{pred}(t)$ was the predicted allowable extension, which we now call $p_1(t\text{-}1)$. If $r(t)$ could be substituted for $y_{real}(t)$ and $p_1(t\text{-}1)$ for $y_{pred}(t)$, we would get

$$err_{model}(t) = \alpha\, p_1(t\text{-}1) - r(t) \geq c(t) - r(t) = err_{pred}(t)$$

Here we used $c(t) \equiv \alpha \min(r(t\text{-}1), p_1(t\text{-}1), \ldots)$, so $c(t) \leq \alpha\, p_1(t\text{-}1)$.

Outside of the simple 2D model, $y_{real}(t)$ and $y_{pred}(t)$ cannot be equated with $r(t)$ and $p_1(t\text{-}1)$. However, under the usual conditions of tool use, the substitution of $r(t)$ and $p_1(t\text{-}1)$ into equation 8-4 provides a reasonable approximation. These usual conditions of use are that the guard is close to parallel with the target shape, and that the orientation of the tool changes little from one timestep to the next. Under these conditions we can estimate

$$r(t) \approx y_{real}(t) + \beta_t$$
$$p_1(t\text{-}1) \approx y_{pred}(t) + \beta_t$$

for some small β_t, as in Figure 10-12.

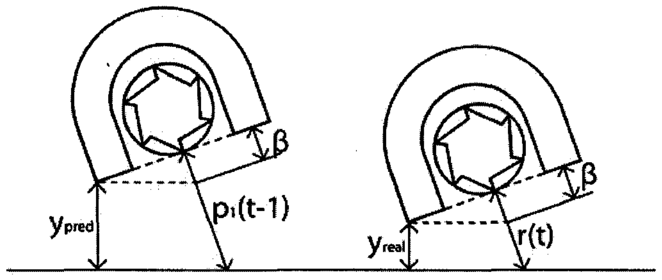


Figure 10-12. If the tool orientation is close to constant and close to flat, we can estimate $r(t) \approx y_{real}(t) + \beta_t$ and $p_1(t\text{-}1) \approx y_{pred}(t) + \beta_t$ for some small β_t.

Then we have

$$err_{pred}(t) \leq \alpha\, p_1(t\text{-}1) - r(t) \approx \alpha\, y_{pred}(t) - y_{real}(t) + (1\text{-}\alpha)\, \beta_t = err_{model}(t) + (1\text{-}\alpha)\, \beta_t$$
$$err_{pred}(t) \sim< err_{model}(t) + (1\text{-}\alpha)\, \beta_t$$

Where we use ~< to mean approximately less than. Since is (1-α) is around 10%-30%, and β_t is also small, this shows that $err_{model}(t)$ is a decent approximate upper bound on $err_{pred}(t)$. Thus if the PFS cutting algorithm can ensure $err_{model}(t) < 0$, then $err_{pred}(t) \sim< 0$ as well.

Now the 2D model has been extended to 3D. In particular, ε and $y(x)$ have been extended to 3D and can be calculated for any recorded timestep. The analysis of Chapter 8 is compatible with these definitions: if the same assumptions hold as in Chapter 8, then Equations 8-3 through 8-5 show that $err_{model}(t) < 0$. In particular, three assumptions are necessary:

- $\varepsilon < \varepsilon_{max}$. The predicted future position must be within ε_{max} of the actual future position.
- The worksurface slope assumption: $y_{\text{real}}(t) \geq (1\text{-}\alpha)^{\varepsilon\_max/w}\, y_{\text{pred}}(t)$

- The guard does not penetrate the workpiece surface. Although in reality the guard cannot penetrate the workpiece surface, errors in the heightfield model or Optotrak readings can cause the PFS to perceive the guard as violating the workpiece surface. The predicted extension may then be wrong because the algorithm expected the guard to rest on the modeled surface.

If these assumptions hold, $err_{model}(t) < 0$. This in turn means that $err_{pred}(t) \sim< 0$. Violations of these assumptions can result in prediction error. We now wish to determine how much the failure of each assumption contributed to prediction error.

Violations of the three assumptions are illustrated in Figure 10-13. The predicted tool position is shown in grey and the actual tool position in black. The solid line depicts the actual surface of the workpiece. The dotted line shows the surface of maximum slope, which is the steepest slope surface descending from the predicted position which obeys the worksurface slope inequality. In this example, all three assumptions are violated: the distance ε from the predicted tool position to the actual tool position exceeds ε_{max}, the worksurface model slopes away steeper than expected, and the tool position violates the worksurface model.

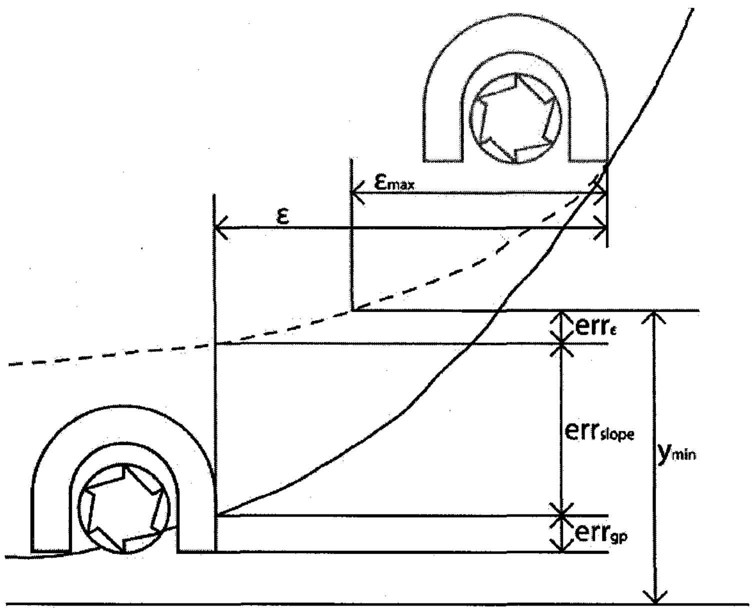


Figure 10-13. Violations of the three framework assumptions. Scale exaggerated for clarity. Predicted tool position is grey, actual tool position is black. y_{min} is the minimum height of the tool off the surface if all assumptions were met.

The height labeled y_{min} in

Figure 10-13 is the lowest that y_{real} can get without violating the assumptions. y_{min} lies on the surface of maximum slope and is distance ε_{max} from the predicted tool position:

$$y_{min} = y_{pred}\,(1\text{-}\alpha)^{\varepsilon\_max\,/\,w}$$

If $y_{real} < y_{min}$, the tool will overcut by the amount of violation: $y_{min} - y_{real}$. When any assumption fails, the contribution of that failure to prediction error is the amount that it potentially allows y_{real} to violate y_{min}.

Error due to violation of ε_{max}

The first source of prediction error is ε exceeding ε_{max}. ε_{max} can be selected based on experimental measurements of ε, but the human-generated motion of the tool is hard to characterize, so any limit ε_{max} cannot be guaranteed. Further, developer has an incentive not to set ε_{max} too high, because the higher ε_{max} is, the more cautious and less efficient the tool must be. Therefore, some violation of ε_{max} is to be expected.

The result of ε_{max} being exceeded is that the tool can travel further than expected down the worksurface slope, and get closer than expected to the target shape (labeled err_ε in Figure 10-13). Without violating the worksurface slope assumption, the minimum height of the tool is

$$y_{min\_\varepsilon} = y_{pred}\,(1\text{-}\alpha)^{\varepsilon\,/\,w}$$

The potential error is the amount that this violates y_{min}:

$$err_\varepsilon(t) = y_{min}(t) - y_{min\_\varepsilon}(t) = y_{pred}(t)\,(1\text{-}\alpha)^{\varepsilon\_max\,/\,w} - y_{pred}(t)\,(1\text{-}\alpha)^{\varepsilon\,/\,w}$$

err_ε is labeled in

Figure 10-13.

Error due to Worksurface Slope

The next source of prediction error is violation of the worksurface slope assumption. The surface of maximum slope represents the lowest the workpiece surface can get without violating the worksurface slope assumption. If the assumption is violated, the resultant error is the difference in height between the surface of maximum slope and the actual workpiece surface at the real tool position. If we let $y_{surf}(t)$ be the thickness of the actual surface at the real tool location (specifically, the point where $y_{real}(t)$ is calculated: it is $y_{real}(t)$ without the guard penetration), we have:

When the slope assumption is violated, the height of the worksurface is less than expected, allowing the tool to approach closer than expected to the target shape. Given ε, the worksurface slope assumption constrains the worksurface height as:

$$y_{expected} = y_{pred}(t)\ (1-\alpha)^{|\varepsilon(t)|/w}$$

If the actual height of the surface at the tool position is $y_{surf}(t)$, the error due to violation of the worksurface slope assumption is:

$$err_{slope}(t) = y_{expected}(t) - y_{surf}(t) = y_{pred}(t)\ (1-\alpha)^{|\varepsilon(t)|/w} - y_{surf}(t)$$

err_{slope} is labeled in

Figure 10-13.

There are many ways the worksurface slope assumption can fail: Most significantly, although the bounds on worksurface slope and position prediction error prevent the blade from violating the target shape, they don't necessarily prevent the blade from cutting past the slope defined by the worksurface slope inequality.

Error due to Guard Penetration of the Workpiece Surface

The final source of prediction error is guard penetration of the worksurface model. Although the actual guard does not penetrate the actual workpiece, sensing inaccuracies and numerical approximation make it inevitable that the sensed guard position will sometimes penetrate the worksurface model. The algorithm compensates for guard penetration in its prediction by assuming that guard penetration will not change from the current timestep to the next, so prediction error is only caused by a *change* in guard penetration. If guard penetration for a timestep is deeper than it was during prediction, the contribution to error is the change in depth:

$$err_{gp}(t) = gp(t) - gp(t\text{-}1)$$

where $gp(t)$ is the guard penetration at timestep t. err_{gp} is labeled in Figure 10-13.

Contribution of Error Sources to Prediction Error

The three error contributions err_{ε}, err_{slope}, and err_{gp} were extracted from the recorded trials. The distributions of these over all trials are shown in Figure 10-14 along with total prediction error.

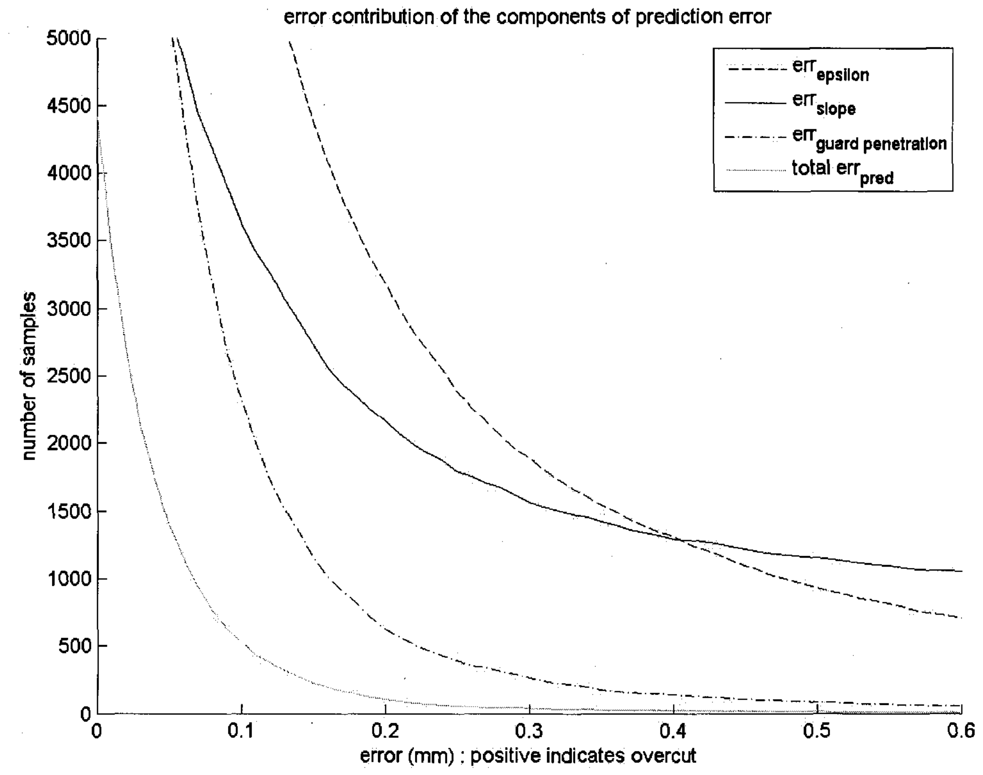


Figure 10-14. Distributions of raw error err_ε, err_{slope}, err_{gp}. The total err_{pred} actually has fewer instances of overcut, because when one component is positive the other two are often negative.

The components $err_\varepsilon(t)$, $err_{slope}(t)$, $err_{gp}(t)$ sum to $err_{model}(t)$, which is an approximate upper bound on $err_{pred}(t)$, so it may be surprising that the individual error components are positive much more often than err_{pred} is. The explanation is that even if one error component has a large positive value, the other two error components can cancel it out, and result in total err_{model} being negative, meaning no cutting error. Because the prediction algorithm only aims for α times the allowable extension, the error components are usually negative, so even when one error component is positive, prediction error is usually negative.

So many of the incidents where an error source was positive, it did not actually result in overcutting. The question is how to present the information in a way that better represents how much each error source contributed to actual overcutting.

One option is to consider min(err_{ε}, err_{pred}), min(err_{slope}, err_{pred}), and min(err_{gp}, err_{pred}). With this formulation, any error source that exceeds err_{pred} is assigned the full blame for error. This means that two error sources can both receive full blame for prediction error, which may seem odd, but it can be though of intuitively as, "This is the amount that err_{ε} should be reduced to eliminate cutting error." Importantly, no error source is blamed for error when $err_{predict}$ is negative.

The distributions of min(err_{ε}, err_{pred}), min(err_{slope}, err_{pred}), and min(err_{gp}, err_{pred}) are shown in Figure 10-15. Note that the y-axis scale is smaller than in Figure 10-14 by a factor of 10, because there were many fewer samples where overcutting actually occurred. This plot shows guard penetration to be the largest source of prediction error by a fair
margin.

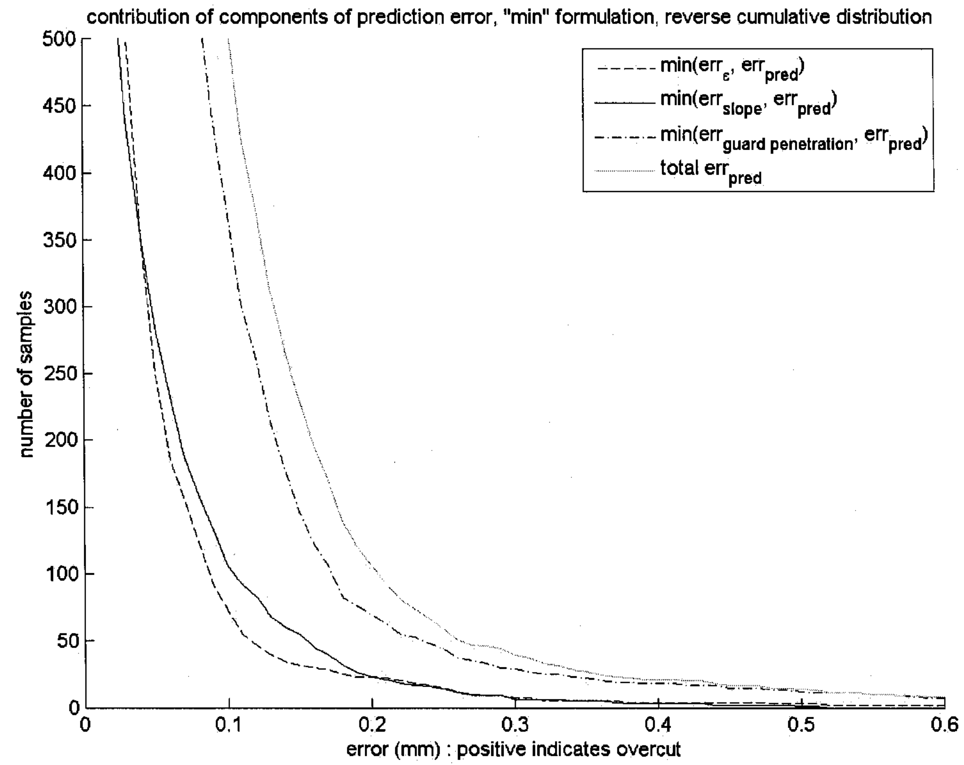


Figure 10-15. Distributions of min(err_{ε}, err_{pred}), min(err_{slope}, err_{pred}), min(err_{gp}, err_{pred}), and err_{pred}.

Although the error formulation involving "min" may be a better indication of which error sources actually created prediction error, the raw distributions of err_ε, err_{slope}, and err_{gp} are also useful. These can be seen as representing the "potential" error that could be caused by the error source. Although err_ε and err_{slope} were only responsible for small amounts of error, their potential impact on prediction error is much larger. There is still a danger that under different circumstances, the existing levels of err_ε and err_{slope} could cause significantly more prediction error, so efforts to reduce these error sources may be warranted.

Fixing the Error

It is unfortunate that guard penetration error is the largest source of prediction error, because it is the least easy to fix. Error due to ε can be fixed by increasing ε_{max}, and in Section 8.2.2 we proposed improving slope error by controlling slope explicitly in software rather than implicitly relying on the guard. Guard penetration, on the other hand, is caused by Optotrak and other modeling error, and will not be so easy to correct.

The good news is that guard penetration error may not actually indicate actual cutting error. Guard penetration error can come in two forms. If the model is in error, the tool can get closer than expected to the target shape, resulting in cutting error. On the other hand, if the current Optotrak reading is in error, it may be that the tool is only *perceived* to be inside the heightfield model and in reality is not overcutting. Several incidences of this latter case were demonstrated by the error maps in Figure 10-4a-c: the heightfield model showed significant cuts that did not appear in the laser scan. Over 8 trials, there were 29 samples where min(err_{gp}, err_{pred}) > 0.3mm, which makes an average of 3.6 such incidents per trial block. Based on the error maps in Figure 10-4a-c, it is reasonable that a fair number of incidents with large err_{gp} were errors in the current Optotrak reading and did not actually result in cutting error. If this is the case, we should also assume that prediction error made up an even smaller fraction of execution error than originally assumed.

10.1.3 Error due to Blade Response Time

Error due to inadequate blade reaction time is the companion to prediction error. Once the necessary blade retraction is predicted, the blade must move fast enough to achieve the requested position by the following timestep. The dynamic constraints of the blade retraction motors limit the blade reaction time. Reaction time is also affected by the calculation time between when the optotrak position is read and when the blade position command is given.

We can separate the effect of blade error from that of prediction error by assuming that the commanded retraction $c(t)$ issued by the algorithm is correct:

$$c(t) = r(t+1)$$

It is then up to the retraction motors to achieve this command before the start of the following timestep $t+1$. If the blade does not achieve the commanded retraction, it will cause cutting error. The cutting error is the difference between the commanded blade position $c(t)$ and the actual blade position at the following timestep, $b(t+1)$:

$$err_{blade}(t+1) = b(t+1) - c(t) \qquad \text{Equation 10-3}$$

Positive error represents overcut. Compare with Equations 10-1 and 10-2, and note that $err_{exec}(t) = err_{blade}(t) + err_{predict}(t)$. Figure 10-16 shows the reverse cumulative distribution of err_{blade} compared to err_{exec} and err_{pred}.

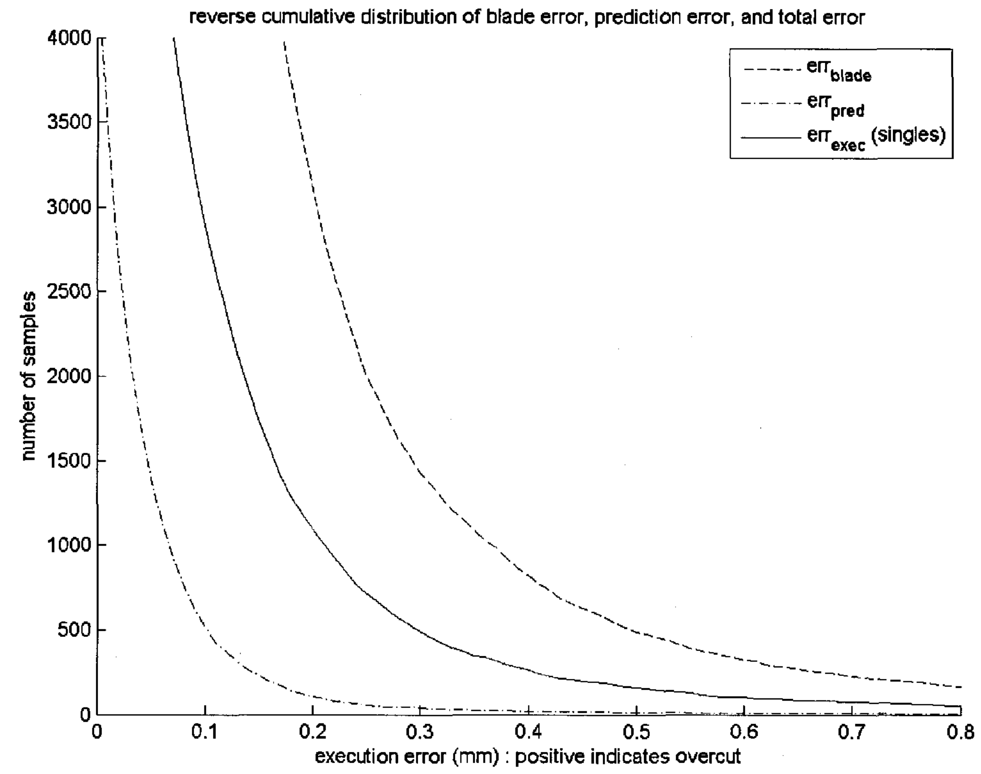


Figure 10-16. Distribution of err_{blade} compared to err_{exec} and err_{pred}. Blade error is a major source of execution error.

The figure shows that blade response error is a much larger error source than prediction error. In fact, it is larger than total execution error. This simply means that $err_{predict}$ was negative during many samples when err_{blade} was positive. The resulting sum, err_{exec} was less than err_{blade}. This is actually expected, because the algorithm aims for slightly negative prediction error.

Causes of Blade Error

The two causes of blade error are dynamic limitations of the motors and the delay between the start of the timestep and when the motor actually begins moving. To separate the effects of these two causes, we can simulate the motor response without any latency. We used the dynamic model for the motor that was experimentally derived in Chapter 4. Blade positions $b_{nodelay}(t)$ for every timestep were simulated based on the position commands $c(t\text{-}1)$, assuming the commands were issued at the start of the timestep. Error due to dynamics is $err_{blade}(t)$ with the no-delay blade model:

$err_{dyn}(t) = b_{nodelay}(t) - c(t\text{-}1)$

Error due to the delay is the difference between the blade position with and without the delay:

$err_{delay}(t) = b(t) - b_{nodelay}(t)$

Note that $err_{blade}(t)=err_{dyn}(t)+err_{delay}(t)$. The distributions of err_{dyn} and err_{delay} are profiled in Figure 10-17.

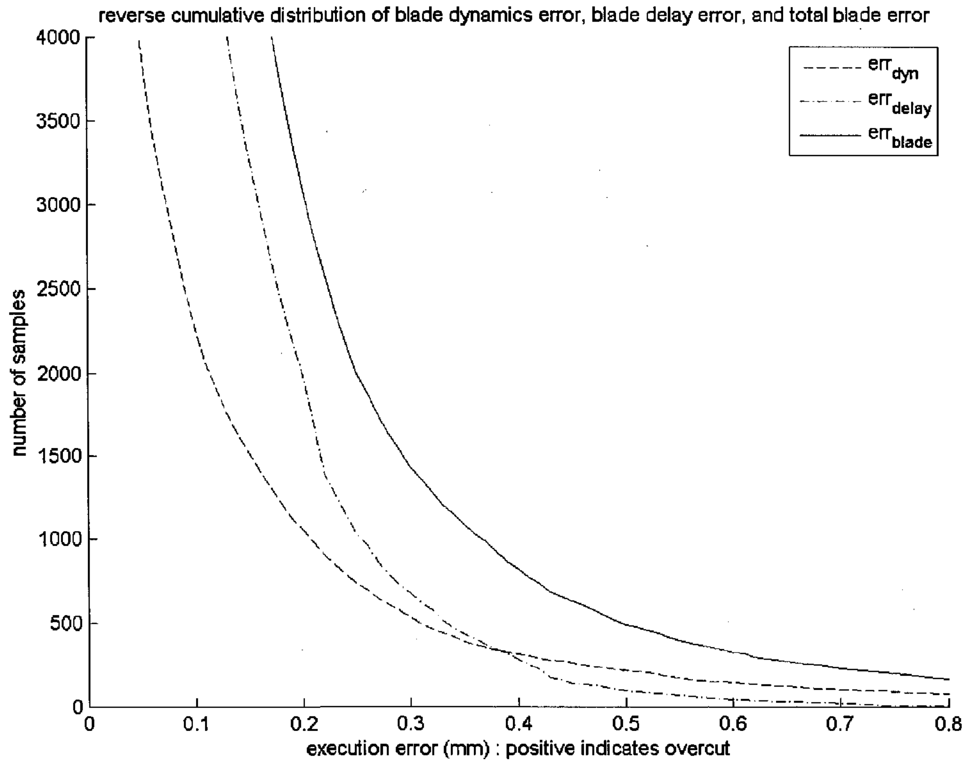


Figure 10-17. Distributions of err_{dyn}, and err_{delay} compared to err_{blade}.

In the figure, err_{delay} has a higher incidence of small errors, but err_{dyn} actually caused more large errors. Since rare large overcuts influence cutting error more than many smaller overcuts, dynamics error is probably the more important error to eliminiate. Both dynamics error and delay error deserve improvement, though, because they are two of the largest sources of execution error.

Fixing the Error

One fix for dynamics error is just to use faster motors. Another fix is to improve prediction to reduce the demands on the motors. In particular, the dynamic model that combines predicted extensions into a single extension command for the "Find Extension Constraint" step is inadequate. The dynamic model assumes constant velocity, but for the radial extension motor, the acceleration phase is significant. We developed a dynamic constraint algorithm that takes into account the acceleration and top speed of the motor, but did not integrate it into the PFS software because of technical limitations. The dynamic constraint calculation requires knowledge of the current blade velocity, which is not currently provided by the software. Integrating this fix into the PFS software could reduce the dynamic demands on the motors, thus reducing dynamics error.

The fix for delay error is twofold. First, extension commands can be issued earlier in the timestep by using partial results. Instead of calculating all predicted extensions before commanding blade position, the software should issue the command after the first allowable extension is calculated, and refine the command as additional extension predictions are calculated. This should work well because the most immediate predicted extensions provide a good approximation of the final extension constraint, and are the most time-critical.

The second fix for blade delay error is a change of hardware. Based on observations of tool motion, in Chapter 9 we found a significant delay from issuing the command to when the extension motor responded. To remove this delay, tighter communication with the motion controller is necessary. A PCI motion control card would be a good choice.

Performance of Ultrasonic Motor

The definitions of err_{dyn} and err_{delay} can also be used to compare the performance of the ultrasonic motor to the DC motor. The blade position $b(t)$ was simulated with the USM constant velocity model, and the resulting err_{dyn} was virtually nil. The extremely high acceleration of the USM allows it to respond very quickly to the small motions required. However, the ultrasonic motor was too unreliable to justify continuing further development with it, and we are not aware of any other commercial supplier of USMs with appropriate specifications.

10.1.4 Error due to Not Cutting Enough

Discussion so far has concerned execution error where the tool cut too far. Yet the spatial distribution of execution error in Figure 10-3 shows slightly over half of execution error was due to waste material that wasn't removed. Areas on the workpiece that are not cut enough are not due to any single error event, because they can always be cut more later. It is up to the user to decide when to stop cutting, and only when cutting is complete are areas that are undercut considered errors.

Although the user is nominally responsible for execution error in the undercut direction, problems with the PFS may prevent the user from being able to cut far enough in some places. Subjectively, I found it very difficult to remove some areas that were shown as undercut on the display. Replaying the cases, the main mode of failure seemed to be not resting the guard flat on the surface in the front-to-back direction. (Figure 10-18) To correct this, users could be trained to raise the back of the tool a bit to positively engage the front tip of the blade. This is easier than trying to position the guard exactly parallel to the workpiece.

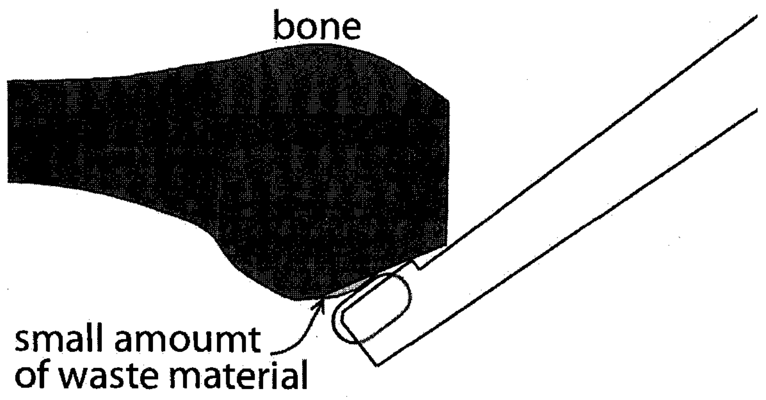


Figure 10-18. Sometimes the user was unable to remove material because the tool was not positioned flat to the bone. Rather than trying to orient the tool perfectly flat, the user should intentionally tip the tool so that the front tip makes contact.

Another problem is that some remaining waste material was difficult to discern on the display. Given the appearance of the final heightfields, I was surprised by the amount of undercut execution error in the results. The problem is that the green that indicates waste material fades gradually into the yellow that indicates cutting is complete, and the colors are close enough that they can be hard to differentiate in the fade. This can be easily fixed by adjusting the colors.

One final phenomenon that can prevent the user from removing waste material is poor calibration of the blade extension mechanism. If the computer commands a light cut, but the blade actually hides slightly behind the guard, it will prevent the user from removing the remaining waste material. This can only be avoided by careful machining and calibration of the mechanism.

10.1.5 Summary of Execution Error

Execution error is cutting error that is reflected in the heightfield model. Execution error can be divided into two distinct phenomena, overcut error and undercut error. Overcut error, where the model was cut too far, is due to failure of the algorithm to retract the blade at one point in time. Undercut error, where the model was not cut far enough, is not due to failure at any one point in time, but is the result of the user stopping cutting before all waste material is removed. Although undercut error is nominally the user's fault, the PFS needs to make material removal as easy as possible for the user.

Execution error comprised about half of total cutting error. In this section, we identified and analyzed potential sources of execution error, to improve our understanding of execution error and to determine where to focus future development. Figure 10-19 summarizes the relative contributions of the execution error sources examined. Undercut error was the largest error source, followed by blade dynamics and blade response latency. Prediction error and missed software frames were relatively minor error sources.

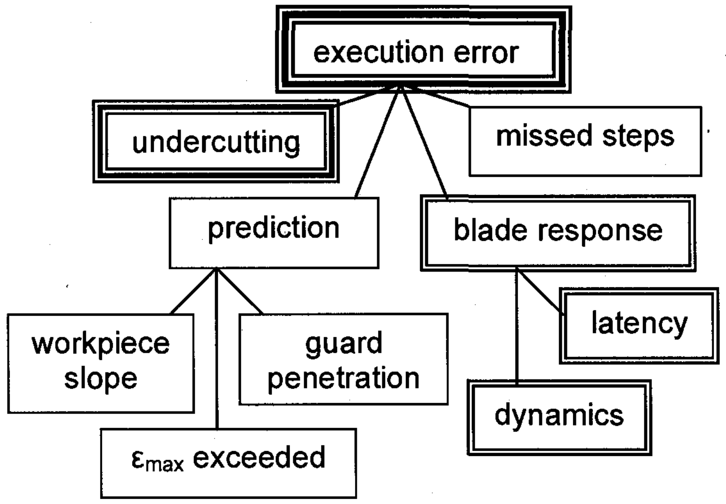


Figure 10-19. Sources of execution error that were examined. Line weight of box represents contribution of error source.

Undercut error accounted for just over half of execution error. We identified some potential sources of undercut error, but they are difficult to quantify because they depend heavily on human actions and perceptions. More precise study of undercut error could be useful going forward. In the meantime, we suggested two easy fixes that may reduce undercut error: adjusting the coloring of the 3D model, and training the user to tilt the tool so the tip makes contact.

Overcut error was studied in the time domain, based on algorithm data recorded at every timestep. For timesteps that were not directly preceded by any missed frames, blade and prediction error sum to total execution error:

$$err_{exec}(t) = err_{blade}(t) + err_{pred}(t)$$

Note that by substituting the components of blade and prediction error, we have:

$$err_{exec}(t) = err_{delay}(t) + err_{dyn}(t) + err_{\varepsilon}(t) + err_{slope}(t) + err_{gp}(t)$$

The most significant sources of overcut error were blade dynamics, err_{dyn}; and blade response delay, err_{delay}. These distributions are plotted in comparison to err_{exec} and err_{pred}

in Figure 10-20. To reduce err_{dyn}, a more accurate dynamic model should be used. To reduce err_{delay}, different motion control hardware is necessary, and the algorithm should issue preliminary motion commands as it computes each predicted extension, rather than waiting for all predicted extensions to be calculated.

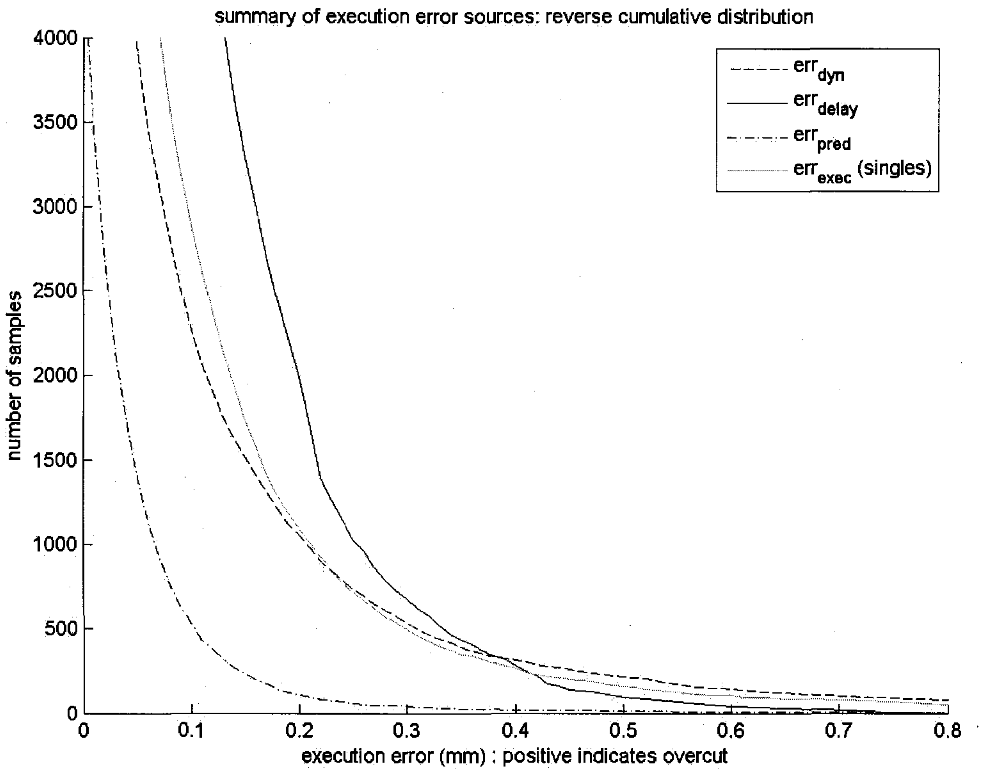


Figure 10-20. Comparison of major sources of overcut execution error: err_{dyn}, err_{delay}, and err_{pred} compared to total err_{exec}.

10.2 Analysis of Modeling Error

Modeling error is the difference between what material the PFS computer thinks is being cut, and what is actually being cut. Modeling error is important because the PFS cannot be expected to cut with more accuracy than it can sense what it is cutting. The direct effect of modeling error is to limit how accurately the tool cuts at the current moment. But modeling error has a second effect as well: by creating error in the heightfield model, it can throw off prediction and therefore cause execution error in the future.

At the beginning of this chapter (Figure 10-3), we measured modeling error in aggregate by matching the scanned bones to the heightfield model. Here, we want to break down modeling error into its components and measure the effect of each. Figure 10-21 highlights the potential sources of modeling error that we have identified. These are:

- tracking error: Error in Optotrak readings.
- blade position error: Error in sensing the position of the blade with respect to the tool handle and the tracking markers.
- software rate: Error in the worksurface model caused by updating it only at discrete timesteps.
- heightfield sampling: Error in the worksurface model due to representing the surface only on a discrete grid of points
- data synchronization: Error in the estimated blade position if encoder and Optotrak samples do not correspond to the same instant.

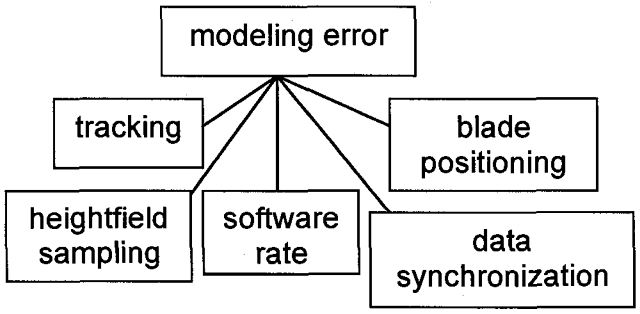


Figure 10-21. Potential sources of modeling error.

Unlike execution error, modeling error cannot be analyzed by studying the data recordings of cutting trials, because by definition modeling error is not reflected in any of the computer data. Instead, we examined each potential error source in benchtop experiments.

Most benchtop experiments yield only a single measurement of the error source. How this value relates to the magnitude of the error during PFS use depends on the complex conditions affecting the tool during use. The error may be roughly constant, or it may have a statistical distribution. For instance, if backlash in the blade positioning mechanism is measured as 0.25mm, the blade position during actual cutting might be constant at 0.25mm or it might vary through a range with 0.25mm as the maximum. The shape of any such distribution can be difficult to determine experimentally. However, we can often make an educated guess based on the nature of the error source.

We must next consider how error measured in benchtop experiments translates to error in the final heightfield model. Many of these error sources (tracking, blade position, and data synchronization errors) act through inaccuracy in the estimated position of the blade with respect to the workpiece. Here we face the same problem as with execution error: correspondence between the benchtop-measured error, which is error in time, and error in the final heightfield model, which is error in space. Although no exact correspondence can be made, the following rules-of-thumb can provide some insight.

First, the direction of the error matters. When the error is toward the target shape, the workpiece model will be updated inaccurately. However, if the error in sensing blade position is parallel to the target shape, the effect on the workpiece model may be relatively minor, because neighboring areas often have similar waste thicknesses.

Also, a constant error may have less effect than one that varies. If the tool is operated with only a small range of orientations, a constant error will mainly serve to offset the resulting cut without affecting its shape. Conversely, if the error is not constant, different areas will be overcut different amounts, and the shape of the cut will be affected as well. Since fit accuracy is our primary focus, an accurate surface that is just offset may be more tolerable.

Another important factor is that the cutting process retains maximum errors. The rare occasions of large modeling error have more effect than the large percentage of times when the modeling error is small. Any single occasion of large modeling error is recorded permanently in the workpiece surface.

These are guidelines, but it can be very difficult to accurately extrapolate from a benchtop experiment to expected aggregate modeling error. However, to some extent we can bypass this step by directly comparing benchtop experiments for different error sources to estimate the largest contributors to modeling error. Since modeling error in aggregate is a significant source of cutting error, we can assume that working to reduce the largest sources of modeling error is a good way to improve cutting accuracy.

10.2.1 Tracking, Calibration, and Registration

Tracking, along with the associated calibration and registration, is one of the largest components of modeling error. Tracking, calibration, and registration combine to tell the computer where the tool is with respect to the bone. The heightfield model is updated based on this position. Errors in perceived tool position will cause the heightfield to be updated incorrectly.

To estimate modeling error due to tracking, we first estimate position error in tracking. Below we review the PFS tracking setup and discuss some of the fundamental characteristics of tracking error. Tracking error can be approached from the level of individual LEDs, individual markers, or at the level of the entire system. We discuss each, and perform some experiments to measure tracking error at the system level.

Tracking Error in the PFS Task

Figure 10-22 depicts the PFS tracking setup. A and B are coordinate frames attached to the workpiece and tool markers respectively. C is the coordinate frame of the cutter, and S is that of the target shape. The basic tracking task of the PFS is to compute ${}^{S}_{C}T$, the transformation matrix of the cutter in the coordinate frame of the target surface. This is calculated via ${}^{S}_{C}T={}^{S}_{A}T\,{}^{A}_{W}T\,{}^{W}_{B}T\,{}^{B}_{C}T$. Here, ${}^{S}_{A}T$ is the registration, which places the target shape with respect to the workpiece marker and ${}^{B}_{C}T$ is the calibration, which describes cutter position with respect to the tool marker. ${}^{S}_{A}T$ is a constant, and ${}^{B}_{C}T$ changes only to reflect blade extension. ${}^{A}_{W}T$ and ${}^{W}_{B}T$ are the tracked marker positions provided by the Optotrak, which are different for each timestep.

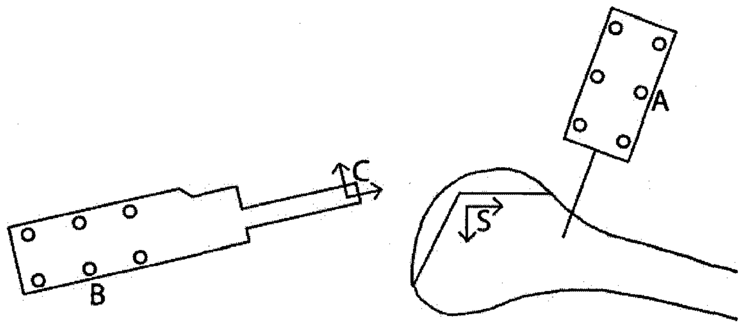


Figure 10-22. Overview of PFS tracking setup. A and B are marker frames of reference, C is cutter frame of reference, and S is target shape frame of reference.

Each of these component matrices ${}^{S}_{A}T$, ${}^{A}_{W}T$, ${}^{W}_{B}T$, and ${}^{B}_{C}T$ contains some error. Errors in ${}^{A}_{W}T$ and ${}^{W}_{B}T$ are tracking errors. Tracking and other errors during the calibration and registration routines result in error in ${}^{B}_{C}T$ and ${}^{S}_{A}T$ as well. Let ${}^{S}_{A}T'$ be the registration matrix used by the PFS, in contrast to the "true" registration ${}^{S}_{A}T$. Likewise let ${}^{B}_{C}T'$ be the calibration matrix used by the PFS. Let ${}^{A}_{W}T'$ and ${}^{W}_{B}T'$ represent the actual marker position data returned by the Optotrak each cycle.

The PFS takes these erroneous transformation matrices and computes

$${}^{S}_{C}T' = {}^{S}_{A}T' \; {}^{A}_{W}T' \; {}^{W}_{B}T' \; {}^{B}_{C}T'$$

the position of the cutter with respect to the target surface and heightfield model, as sensed by the PFS. When the PFS updates the heightfield model based on the perceived cutter position, the error in ${}^{S}_{C}T'$ causes error in the heightfield. Our goal in this section is to estimate the error in ${}^{S}_{C}T'$, and its effect on modeling error.

Basic Characteristics of Tracking Error

The study of tracking error can be aided by an understanding of the basic process of tracking and some of its characteristics. Tracking error can be considered at several levels, from individual LEDs to the complete system: First the tracking camera locates LEDs, which are combined into markers. From marker positions, the application finds the location of points of interest such as the blade, and finally finds the location of points on the tool with respect to the workpiece frame of reference.

Tracking begins with individual LEDs. Northern Digital lists Optotrak LED accuracy as 0.1mm rms. This specification is for 30 averaged consecutive readings with a stationary LED. The error will be worse for a single reading, and worse yet if the LED is in motion.

An important property of LED error is that it is not simply Gaussian noise that can be filtered out. The majority of error is actually bias that is constant for a given LED position and orientation. A major source of this bias is refraction through the epoxy coating on the diode. [Crouch 2005] To illustrate this bias, we pivot-calibrated a probe, and then recorded the error in sensed probe tip position as the probe was rotated around its axis. Figure 10-23 shows error as a function of viewing angle. At high viewing angle, there is a large bias in sensed position, which can be significantly worse than the stated RMS error. Therefore, it seems that this bias is a more significant effect than Gaussian noise. We attempted to minimize this error in the surgical system by careful design of tracker mounting angles.

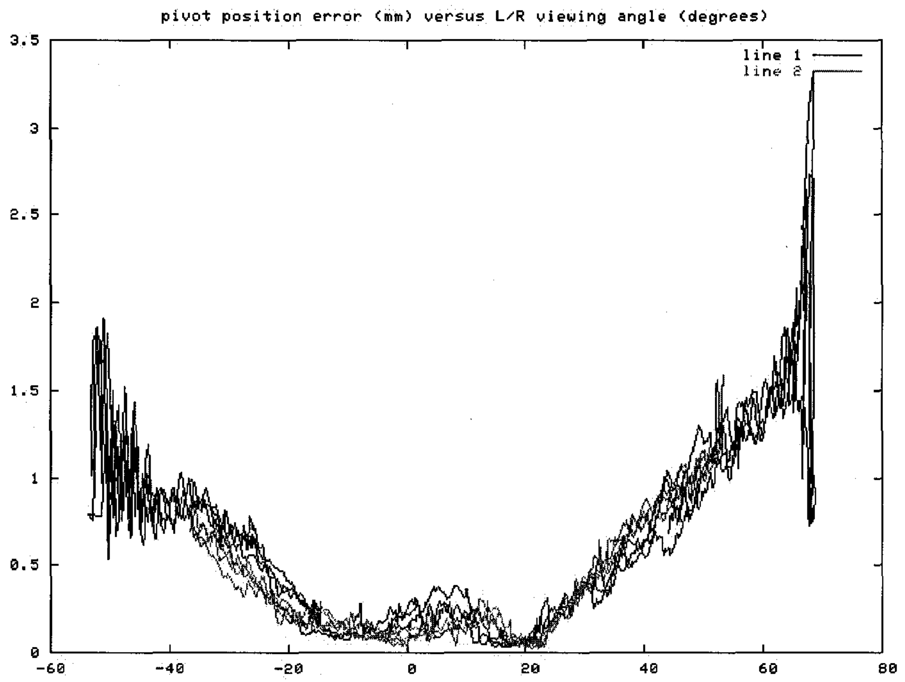


Figure 10-23. Tracking error as a function of viewing angle.

Individual 3D LED positions are used to find the 6D position of a tracking marker. The known nominal LED positions on the marker are matched to the sensed LED positions to determine the marker position. Because that match is overconstrained, a least-squares fit must be found. The rotational accuracy of the 6D position depends on the marker geometry: A wider baseline between LEDs will give more accurate orientation.

Motion also affects accuracy at the marker level. The Optotrak flashes the LEDs on the marker sequentially. If the marker is in motion, each recorded LED position will correspond to a slightly different marker position. This can cause orientation as well as translation error as the Optotrak tries to match the recorded LED positions to the known nominal positions.

Next the 6D marker position is used to find the position of other points of interest on the object that the marker is attached to, such as the blade position. Error here depends on angular error in the marker position and position of the point of interest with respect to the marker. If the point is far from the marker, small angular errors will cause big error in the point of interest position.

Given a measurement for LED error, it is possible to calculate from marker geometry the expected error at the marker and point-of-interest levels. [West 2004] and [Smith 2005] are two studies that take this approach. However, these studies assumed a Gaussian error model, whereas I believe that bias from viewing angle and other sources is a more significant error source.

[Chassat 1998] and [Khadem 2000] measured accuracy at the marker level. [Khadem 2000] constructed an accurate 3D positioning grid and translated the marker throughout the measuring space of the camera. [Chassat 1998] translated a tracked marker along one linear dimension and compared the sensed displacement versus actual displacement. This was repeated along each of the principal axes of the camera.

The problem with studying accuracy at the marker level is that these studies cannot take rotation into account. Tolerances in the manufacture of the marker LEDs and in gluing the LEDs to the marker frame make it impossible to know the true location of the coordinate frame on the marker. [Crouch 2005] Therefore, as the tool is rotated, it is impossible to tell to what extent motion of the coordinate frame comes from motion of the actual coordinate frame, or from bias error.

The lack of ground truth for the marker coordinate frame also makes it impossible to verify calibration and registration. The relationship between the tool or workpiece and the marker frame of reference cannot be physically measured because the marker frame of reference is unknown.

Measuring Ground Truth Accuracy at the System Level

A viable alternative to measuring the error in each of the tracking, calibration, and registrations matrices is to measure the error in the end-to-end matrix sTc, the position of the tool with respect to the workpiece. This can be done because ground truth geometry *can* be known for the tool and workpiece, or special models thereof, by careful machining.

This approach was taken by [Bach 2007]. A precision-machined workpiece was manufactured with 47 fiducial divots. The workpiece was registered, and then a pointed probe was used to measure the location of each divot. Since the ground truth location of each divot is known, the difference between the measured and known position of the divot is ground truth error. Mean error was 0.41mm for the Northern Digital Polaris. Northern Digital lists Polaris accuracy at 0.25mm RMS, and lists the Optotrak at 0.1mm RMS, so these results may be slightly worse than we can expect. The authors note that "Averaging of 5 or 10 consecutive measurements did not significantly improve the results," which illustrates that the error is indeed bias rather than gaussian noise.

In this experiment, the probe is analogous to the PFS. The accuracy with which the probe's position on the surface is known is equivalent to the accuracy with which the cutter's position can be known with respect to the workpiece.

However, the PFS includes one additional source of error. Unlike the ball-pointed probe, the PFS is pivot-calibrated at one location (the front calibration divot), and then the position of the tool with respect to the workpiece is measured at another location (the blade.) More specifically, the location of each point on the blade with respect to the workpiece is important for cutting accuracy. To be able to locate points away from the front calibration divot, the orientation of the PFS frame of reference must be determined. In PFS calibration, pivot calibrating the front divot determines the divot location (the translation component of ${}^{B}_{C}T$), and then the other two divot points are touched to determine the orientation of the PFS frame of reference (the rotation component of ${}^{B}_{C}T$).

The error due to this rotational component simply sums with the error measured by [Andres 2007], which we will call "pivot position error". Let D be a reference frame attached to the PFS front calibration divot, and let ${}^{D}P = [x\ y\ z\ 1]^T$ be a point of interest on the blade (in homogeneous coordinate representation). Then the calculated position of P with respect to the workpiece is

$${}^{S}P' = {}^{S}_{A}T'\ {}^{A}_{W}T'\ {}^{W}_{B}T'\ {}^{B}_{D}T'\ {}^{D}P = {}^{S}_{A}T'\ {}^{A}_{W}T'\ {}^{W}_{B}T'\ {}^{B}_{D}T'\ ([x\ \ y\ \ z\ \ ]^T + [0\ \ 0\ \ 0\ \ 1]^T)$$

$${}^{S}P' = {}^{S}_{A}T'\ {}^{A}_{W}T'\ {}^{W}_{B}T'\ {}^{B}_{D}T'\ [0\ 0\ 0\ 1]^T + {}^{S}_{A}T'\ {}^{A}_{W}T'\ {}^{W}_{B}T'\ {}^{B}_{D}T'\ [x\ y\ z\ 0]^T$$

Error in the first term is pivot position error. Error in the second term is affected only by the rotational components of ${}^{S}_{A}T'$, ${}^{A}_{W}T'$, ${}^{W}_{B}T'$, and ${}^{B}_{D}T'$.

The experimental method of [Andres 2007] could be augmented to take into account the rotation component that affects the PFS by constructing a tool with calibration divots identical to the PFS, and a ball tip where the PFS cutter is, as in Figure 10-24. The tool would be calibrated with the three divots just as the PFS is, and then the ball tip would be used to sample the location of the divots in the accuracy validation phantom.

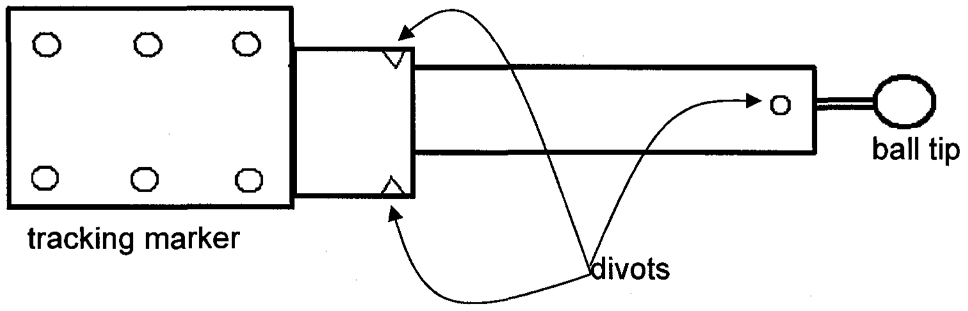


Figure 10-24: Proposed PFS mockup for evaluating optical tracking error.

We did not implement this experimental setup, but we did employ some simpler methods to estimate the pivot position error and error due to rotation.

To estimate the pivot position error, a ball-end probe was pivot calibrated in the front calibration divot of the PFS. Then the probe was pivoted in the divot once again, and the measured position of the divot center with respect to the probe was recorded. This is like a limited version of the experiment used in [Andres 2007], where the workpiece consists of a single divot. The error $\varepsilon_{\text{point}}$ was defined as the distance of each recorded divot position from the mean. The experiment was repeated with three separate markers. The distributions of $|\varepsilon_{\text{point}}|$ for each of the markers are shown in Figure 10-25. Note that these results underscore the importance of testing tracking markers for best performance. Marker three did significantly worse than the first two.

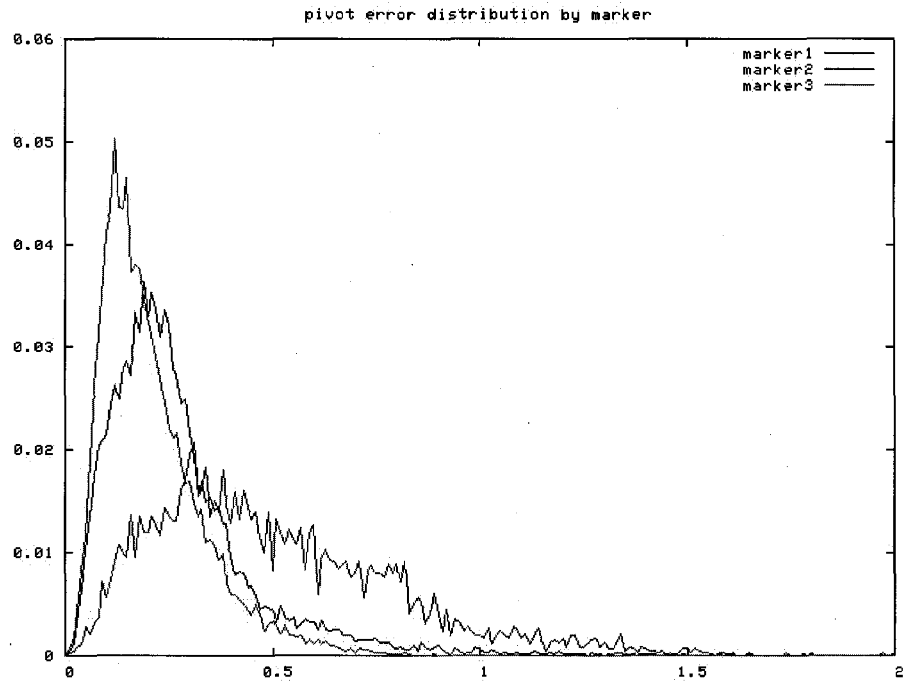


Figure 10-25. Distribution of pivot position error (mm) for three different probe markers.

To measure the effect of rotation error, we tested the repeatability of the rotation component of PFS calibration. Since there is only one "true" calibration, lack of repeatability in calibration is a minimum on error in calibration. In addition to rotation error in the calibration, rotation error in the end-to-end matrix ${}^{S}_{C}T'$ incorporates rotational error in the tracking measurements ${}^{A}_{W}T'$ and ${}^{W}_{B}T'$. These same rotational errors ${}^{A}_{W}T'$ and ${}^{W}_{B}T'$ are also present during the calibration procedure, so calibration repeatability is an estimate of end-to-end rotation error.

To measure the rotation repeatability of calibration, the PFS was calibrated 8 times. The effect of rotation error on sensed blade position was calculated by multiplying the rotation component of the calibration by the offset ${}^{D}P$ between the calibration divot and the center of the front sphere of the blade. Variation of each trial from the mean is listed in the table below. These errors are a significant part of modeling error, but they are certainly a smaller effect than pivot position error.

Estimated contribution of rotation error in calibration To error in the position of a point of interest on the blade (mm).							
0.0554	0.0684	0.0224	0.1285	0.0236	0.0779	0.0721	0.0313

Effect on Modeling Error

Based on these experiments, we can expect roughly 0.3mm RMS modeling error from optical tracking effects. The error will be different each timestep, and it may tend to be larger in some places and smaller in others. The error is omnidirectional. Since the heightfield model is mostly affected by error normal to the target surface, the effect on modeling error will be less than the full magnitude of tracking error. Based on the measured spatial distribution of total modeling error, we can conclude that tracking error probably comprises the majority of modeling error.

Fixing Optical Tracking Error

Not much can be done to fix optical tracking error. The Optotrack is an off-the-shelf product that has been carefully engineered and calibrated. The precautions described above, such as limiting marker viewing angle and the distance from marker to point-of-interest, should continue to be observed. However, we must accept that any sensor will have some noise in its readings.

10.2.2 Blade Positioning

To properly update the heightfield model, the tracked tool position must be adjusted to account for the position of the blade within the tool. The blade position is sensed by incremental encoders mounted on the actuation motors. The encoders have adequate resolution, but there may still be some error between the blade position as read by the encoders and the actual blade position. This may be due to homing problems, play/backlash, flex, or simply miscalculation. We performed several experiments to test these factors. To measure the blade position in these

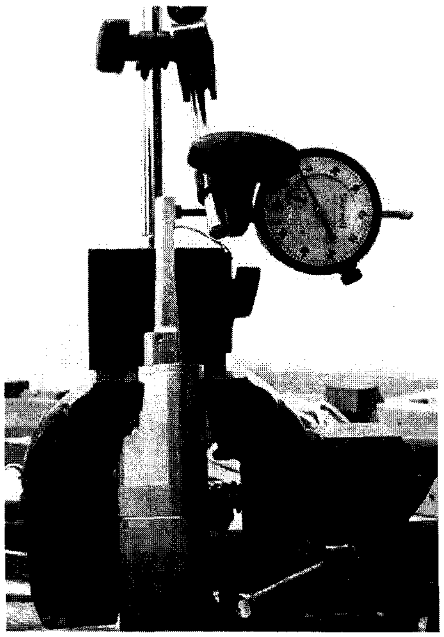


Figure 10-26: Dial indicator measuring blade position

experiments, the tool was held in a vice while a dial indicator (Starrett 25-441) fitted with a flat anvil measured the actual blade position. (Figure 10-26.)

General Positioning

First we measured general blade positioning accuracy. The blade was moved to several positions under computer control and the position read on the dial indicator. The dial indicator provided enough force to preload the blade against backlash, without so much force to induce flex. For axial tests, the radial position was kept at zero, and vice versa.

Results are listed below. General positioning error spanned a total range of 0.07mm radial and 0.02mm axially.

Radial Position		Axial Position	
Encoders (mm)	Dial Indicator (mm)	Encoders (mm)	Dial Indicator (mm)
0.01	(ref) 0.01	0	(ref) 0
0.50	0.57	0.51	0.51
1.00	1.00	1.01	0.99
1.50	1.53	1.51	1.50
1.80	1.84	0	0
-0.13	-0.07		
0.01	0.01		

(These results are converted from inches. The accuracy is ± 0.025mm.)

Backlash

The PFS blade positioning mechanism includes several potential sources of blade positioning backlash. The mechanism is based on a carriage which rides on a round shaft. It slides forward and back to allow axial blade motion and rotates around the shaft to approximate linear travel in the radial direction. The interface between the carriage and shaft includes 0.0005" – 0.0015" clearance so that the carriage can run freely. In the radial direction, this theoretically means 0.004" – 0.012" (0.1mm – 0.3mm) play at the cutter tip due to the cantilever of the blade. Axial backlash is not affected by this rotational slop in the carriage, so we expect it to be smaller.

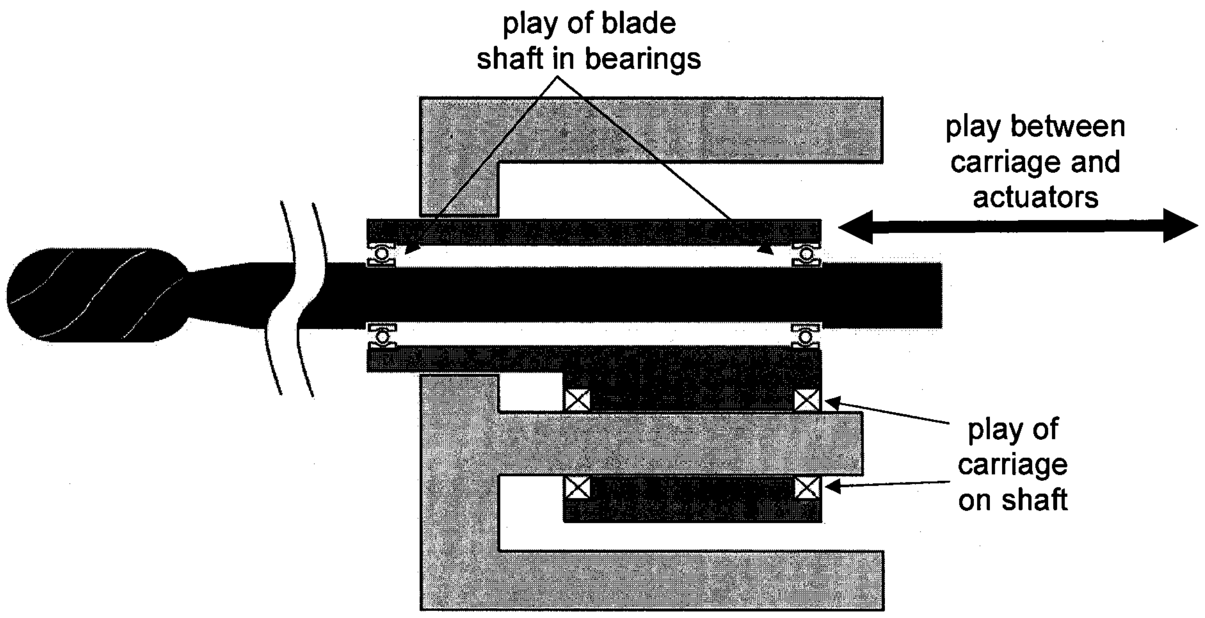

play of blade shaft in bearings
play between carriage and actuators
play of carriage on shaft

Figure 10-27. Sources of backlash in blade positioning.

The carriage contains ball bearings on which the actual blade shaft rotates. The ball bearings also introduce slop into the mechanism. Finally, there is backlash due to the gears and couplings that connect the carriage to the extension actuators.

Backlash was measured with the PFS in the vice being read by the dial indicator. The blade was moved by the computer to several positions and at each the blade was moved with finger pressure through the full range of backlash. Backlash could be easily felt by hand, and distinguished from further pressure that might induce flex. Results are summarized below:

Radial Tests				
Position from Encoders (mm)		Dial Indicator Radial Position (mm)		Radial Backlash (mm)
Ax	Rad	Min	Max	
0	0	0 (ref)	0.229	0.229
0	1	0.889	1.143	0.254

1	1	0.787	1.041	0.254
1	-1	-1.194	-0.965	0.229
0	0	-0.203	0.076	0.279
Axial Tests				
Position from Encoders (mm)		Dial Indicator Axial Position (mm)		Axial Backlash (mm)
Ax	Rad	Min	Max	
0	-1	0 (ref)	0.178	0.178
1	-1	1.194	1.372	0.178
0	0	0.051	0.102	0.051

(These results are converted from inches. The accuracy is +- 0.025mm.)

Radial backlash was approximately 0.25mm and axial backlash was approximately 0.18mm.

Note also that there is a positioning error between the first radial reading and subsequent readings that is not consistent with our prior general position accuracy tests. This may be due to experimental error such as the dial indicator being bumped.

Blade flex

Next the blade was pulled in the axial extension and radial extension directions with a spring scale to measure blade deflection under load. The force was applied at the base of the cutting tip. The blade was held in position by computer control and no change in position was measured through the encoders, to an accuracy of 0.01mm in the radial direction, and 0.017mm in the axial. Results are listed below.

	Radial flex (mm) at 0mm radial ext	Radial flex (mm) at 1mm radial ext	Axial flex (mm) at 0mm axial ext	Axial flex (mm) at 1mm axial ext
Finger pressure	0 (ref)	0 (ref)	0 (ref)	0 (ref)
1lb	0.13	0.13	0.10	0.05
2lb	0.25	0.25	0.13	0.13
3lb	0.46	0.38	0.15	0.18

4lb	0.64	0.46	0.15	0.18

(These results are converted from inches. The accuracy is +- 0.025mm.)

As discussed in Chapter 4, the maximum forces observed on the blade were 6N (1.3lb) axial and 2.5N (0.6lb) radial, so the maximum expected modeling error caused by flex is about 0.1mm. These maximum forces represented the point where the blade started to chatter violently. I expect cutting force in general to be even less, especially during the critical final stages of cutting, where the PFS takes light cuts.

Homing

Because the blade extension motors use incremental encoders, the blade position must be homed on power-up. This is done by running each motor to the end of its range of motion. The offset from end of travel to the neutral position where the guard is neither extended nor retracted is manually calibrated beforehand.

The homing procedure must be repeatable or the perceived blade position will be off, and the heightfield model will be updated incorrectly.

To measure homing repeatability, the tool was homed 5 times consecutively, and 4 more times where the blade was moved back-and-forth briefly in between each homing. The encoder position of each homing relative to the first homing was recorded. In the axial direction, the tool homed to the same encoder count every time, where one encoder tick covers 0.017mm of axial motion. Results for the radial direction are given below.

encoder	mm
0	(ref) 0
-14	-0.00235
2	0.00034
3	0.00050
-1	-0.00017
0	0
12	0.00202
-11	-0.00185

19	0.00319

Blade position (encoder counts and mm) as measured by encoders, for successive rounds of homing in the radial direction. Results are relative to the first homing.

The blade position error due to homing is negligible.

Summary of Blade Position Errors

The table below summarizes the results of the experiments on blade position error.

	Radial	Axial	Notes
Position	0.05	0.02	Variable.
Backlash	0.25	0.18	Load from cutting forces may limit the range of this error.
Flex	0.13	0.10	Variable. Probably much less than this maximum.
Homing	0.003	0	Constant bias for a given run. Negligible.

Backlash was the largest measured source of error, and in fact its 0.25mm error in the radial direction is very significant compared to the measured distribution of total modeling error. However, cutting forces will probably push the mechanism against one side of the backlash range so that the actual error due to backlash will be somewhat smaller than 0.25mm.

Blade flex was the next largest measured source of error. This too will probably be smaller than the listed magnitude of 0.13mm radial / 0.10mm axial. Those numbers are based on the largest cutting forces seen the cutting force experiments, which occurred just before the blade began to chatter violently. Blade force in general should be much smaller than that, especially during the final stages of cutting, which are most crucial to final cut accuracy. The exact amount of flex will be variable through the cutting use, so the effect of flex error is variable, not just a bias.

General positioning accuracy is the next largest measured source of error. The experimental results should give a good indication of this error during actual cutting. This error will be variable during use, as opposed to giving a constant bias.

The final error source is homing error, which is negligible.

Effect on Final Modeling Error

For the UKR procedure, almost all cutting is done in the radial direction, so radial error is our primary concern. Unlike tracking error, which is omnidirectional, the direction of blade error is usually normal to the target shape, so the full error magnitude applies to modeling error.

The biggest question about the magnitude of total blade positioning error is how much variability will actually be caused by backlash. In general, I expect cutting forces to provide ample load to limit backlash. However, during the critical finishing cuts, the PFS takes very light cuts and so the cutting forces may no longer limit backlash. If this is indeed a problem, adding an active preload against backlash could improve cutting accuracy.

Error from flex will in general be much smaller than listed, because cutting forces will be much smaller than the maximum. Depending on the behavior of backlash, the total modeling error due to blade error will probably be 0.1mm – 0.25mm. Unlike tracking error, which has long tails on its distribution, blade position error is likely more limited to the ranges measured above. In all, blade error is probably a slightly smaller source of modeling error than tracking error is.

How to Fix the Error

One easy fix for blade positioning error is to preload the blade carriage to reduce or eliminate backlash. Another possible improvement is to move the encoders from sensing motor position to sensing position of the blade carriage directly, although that modification would be very challenging and may not be the best application of improvement efforts. Otherwise, there are no easy solutions. It is inevitable that any mechanical system will be subject to flex and inaccuracy in construction, and careful design is necessary to mitigate these problems.

10.2.3 Software Rate

Another potential source of modeling error is the rate at which the software updates the heightfield model. The heightfield model is updated by removing the material that intersects the perceived blade position for each Optotrak sample. The model doesn't reflect any intermediate positions the blade passes through in the time between software updates. (Figure 10-28) With straight-line motion at a tool speed of 0.2m/s, this will result in maximum errors of 0.17mm. For 0.1m/s tool speed, the maximum error will be 0.085mm.

These numbers are significant compared to modeling error, but in general modeling error due to software update rate will be significantly smaller. User velocity, profiled in Figures 9-4 and 9-12, is usually much less than 0.1m/s. Further, the maximum modeling error only occurs at a sharp peak, and most of the error is significantly closer to zero.

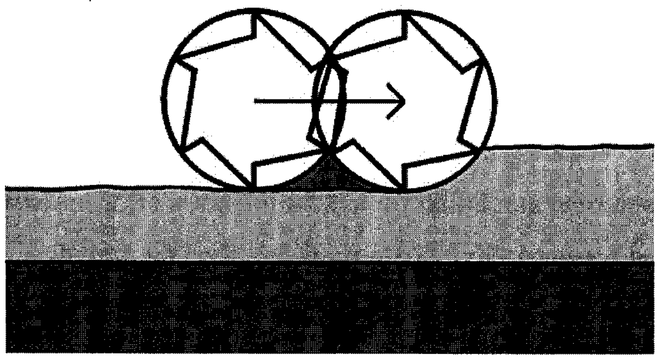


Figure 10-28. Shaded area between two consecutive position samples is not modeled

We also estimated this error experimentally in a previous set of trials. The heightfield model was recalculated with 0.1mm position interpolation steps between the original update positions. The error between the original heightfield model and the model recalculated with interpolation was insignificant.

Theoretically, the potential effect of software rate on modeling is less than 0.1mm, and experimentally it has been see to be negligible. If software rate does become a problem in the future, one solution is to interpolate between the blade positions, and update the heightfield based on these interpolated positions.

10.2.4 Data Synchronization

The synchronization of Optotrak tool position readings with encoder blade position readings is also a potential source of modeling error. Poor synchronization can cause the estimated blade position with respect to the workpiece to assume a position that it never took in reality.

The encoder data is read directly from the PCI card in the PC, with negligible delay, but the timing of Optotrak data is more complex. The tracking data is collected over a period of 12ms because each LED is fired sequentially. The Optotrak hardware solves the position of each marker and extrapolates all the positions forward to the time when they're reported to the PFS program. After reading that data, the PFS then reads the encoder values from the PCI card. If the Optotrak extrapolation is accurate in time, the Optotrak and encoder data should be well synchronized.

Note that the accuracy of the readings given by the Optotrak was verified in Section 10.2.1, but this does not guarantee the timeliness of the readings.

We can calculate a theoretical maximum for synchronization error based on the Optotrak period, and the tool and blade speeds. We assume that the Optotrak data and the blade encoder data both come from within the same Optotrak period. The worst-case scenario is that the readings are separated by the entire period: 12ms. The error due to synchronization is the distance between the perceived blade position with respect to the workpiece, and the closest actual position that occurs within the Optotrak period. Since the tool moves much faster than the blade retraction, the actual position that best matches the perceived position is that corresponding to the sampled Optotrak position. The true blade extension at this point is at most 12ms off from the measured blade extension, so the error from synchronization is 12ms times the blade speed, giving 12ms * 40mm/s = 0.48mm.

The worst-case estimate of 0.48mm is very significant compared to total modeling error, but in reality it will probably be significantly smaller. The radial blade extension motor almost always moves much more slowly than top speed, because the motions commanded of it are typically very small, as seen in Figures 9-9 and 9-15. Further, preliminary experiments performed by mounting a tracking marker to the ultrasonic motor showed very good synchronization between encoder and Optotrak for constant velocity motions.

On the other hand, the Optotrak's extrapolation of positions to the reporting time may be worse for erratic motions than for the smooth motions studied. It is also important to remember that any single instance of large error is recorded in the heightfield, so even though the blade retraction motor rarely reaches top speed, the impact on modeling error can be large when it does.

Synchronization error probably is not a major contributor to modeling error. However, the evidence for this claim is preliminary, and the potential error is large. Therefore, we should remain aware of the potential for error, which may merit further investigation in the future.

10.2.5 Heightfield Resolution

Thus far modeling error has been defined as error between the heightvector points and the actual surface. Another error in the heightfield model is sampling error – the difference between the actual surface and the implicit surface interpolated between the heightvectors. In these tests, the spacing between heightvectors was approximately 1mm.

The potential impact of modeling error due to inadequate heightfield resolution is the same as any other modeling error: There is error in the final workpiece shape that the PFS computer is not aware of and so cannot address. Additionally, this modeling error, like any other, can contribute to execution error if the PFS guard does not rest where the software predicts it will.

The magnitude of the error that can occur between heightvectors is determined by the radius of the bur and the distance between heightvectors. In between heightvectors, the actual worksurface may be raised or depressed. In the latter case, the amount of depression is limited by the radius of the bur, as in Figure 10-30. With a 3mm bur radius and 1mm heightvector spacing, the maximum depression is 0.04mm. Raised deviations come in the form of cusps (Figure 10-30), which can actually be arbitrarily large. However, the practical height of cusps is limited by material properties and smooth, continuous motion typical of PFS use.

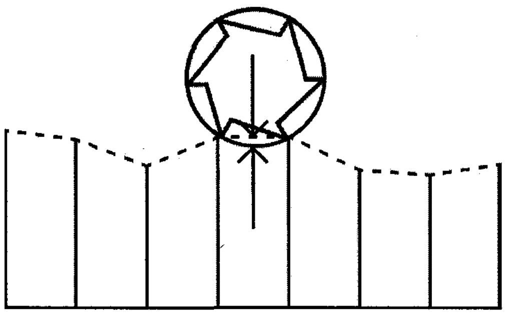


Figure 10-30: Unsampled depressions are limited by the cutter radius. Dotted line indicates the worksurface interpolation between vertical heightvectors.

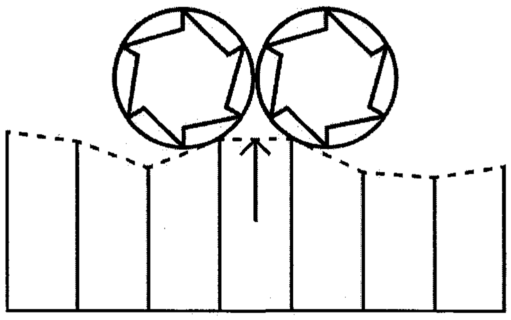


Figure 10-30: Unsampled cusps can theoreticallv be large.

Modeling error based on limited resolution is relatively benign. The potential size of depressions is insignificant, and although cusps can theoretically be larger, they should be rare because of the multiple passes the tool makes over the workpiece in the finishing stage. As for the effect on execution error, overcut modeling error poses the greater risk, because it allows the guard to rest closer than expected to the workpiece and can lead to overcut execution error. Since overcut modeling error (depressions as in Figure 10-30) is so small, the practical effect on execution error will be negligible.

10.2.6 Summary of Modeling Error

In this section we identified potential sources of modeling error, and estimated their contributions with benchtop experiments and mathematical analysis. Connecting benchtop results to a specific spatial distribution of modeling error is difficult, but we can instead compare error sources by comparing benchtop experiments directly. Modeling error makes up about half of total cutting error, so the largest sources of modeling error are also significant sources of total cutting error.

Figure 10-31 summarizes the measured sources of modeling error. Optical tracking error was the largest source of modeling error. The contribution of blade position error was also significant, but probably slightly smaller than optical tracking error. The contributions of software update rate and heightfield sampling resolution were negligible. Data synchronization also probably has a negligible effect, but may warrant more investigation.

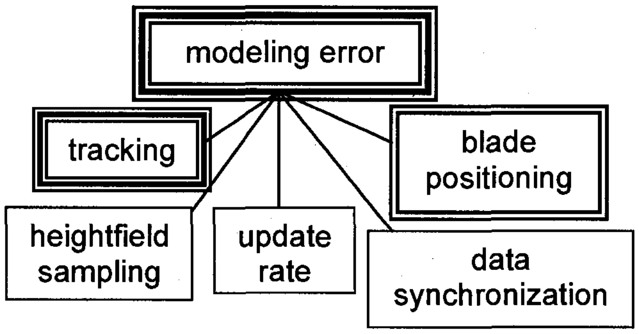


Figure 10-31. Sources of modeling error examined. Line weight of box represents contribution of error source.

There is no easy fix for optical tracking error. The Optotrak is an off-the-shelf product which has been carefully designed and meticulously calibrated. Some amount of error is unavoidable in any sensor.

For blade position error, anti-backlash loading may reduce error due to backlash. Other than that, there are no easy solutions for blade position error. Any mechanical system will be subject to flex and imperfect construction. Good design is necessary to minimize these effects.

Chapter 11. Evaluating the Usefulness of Prediction

In Chapter 9, we showed that the PFS cutting algorithm, which uses prediction, achieved acceptable cutting accuracy. In this chapter, we wish to evaluate the usefulness of prediction in the algorithm. To compare the cutting accuracy of the prediction algorithm to a "simple" algorithm without prediction, we will simulate the response of the simple algorithm on a recorded cutting trial. Cutting accuracy alone is not sufficient for evaluating the usefulness of the prediction algorithm, because accuracy can always be improved by cutting more conservatively, sacrificing efficiency. We will introduce a formula for cutting efficiency, and demonstrate that prediction improves cutting accuracy with little loss of efficiency.

11.1 Simulating the Response of a Simple (Non-predictive) Algorithm on Recorded Data

In order to compare the accuracy of the prediction algorithm to a simple algorithm, we simulated the response of the simple algorithm based on data recordings from the cutting trials. Recall some of the variables described in Chapter 10 which can be extracted from the data recordings:

$b(t)$ is the actual blade extension at the start of timestep t.
$r(t)$ is the required maximum blade extension at t.
$p_i(t)$ is the predicted allowable blade extension for i steps ahead, i.e. predicted $r(t+i)$.
$c(t)$ is the commanded blade extension issued at t. This is based on $r(t)$ and $p_i(t)$.

For the prediction algorithm, the blade extension command is:

$$c_{\text{pred}}(t) = \alpha \min(r(t), p_1(t), p_2(t) + s\,\Delta t, \ldots)$$

Where α is the extension multiplier. For a simple algorithm, the blade extension command would just be:

$c_{\text{simple}}(t) = \alpha\, r(t)$

Since we can extract $r(t)$ from the recorded data, we can simulate what $c_{\text{pred}}(t)$ would be for each recorded timestep in the trials. From that we can calculate the execution error that would result from the simulated algorithm. In Equation 10-2, we defined the error due to the prediction algorithm as the difference between the commanded blade extension and the actual required blade extension at the next timestep. We repeat that formula here, writing $c_{\text{pred}}(t)$ in place of $c(t)$, to distinguish from $c_{\text{simple}}(t)$:

$err_{\text{pred}}(t) = c_{\text{pred}}(t\text{-}1) - r(t)$

We can likewise define the error due to the simple algorithm as:

$err_{\text{simple}}(t) = c_{\text{simple}}(t\text{-}1) - r(t)$

The concern with simulating the simple algorithm response this way is that it does not take into account how the decisions of the algorithm affect the situations the algorithm will encounter down the road. At each recorded timepoint, the simulation method asks the question, "If algorithm X were confronted with this situation, how would it perform?" and it addresses that question very well. The limitation of the simulation is just whether algorithm X would ever encounter the given situation. This is certainly an issue, since management of the worksurface slope is an important aspect of the PFS cutting strategy. However, the effect should be limited if α is not significantly changed between recording and simulation. Also, it is still useful to examine how well multiple algorithms react to the same stimulus. In all, simulation is a useful tool for comparing different algorithms.

We simulated the performance of two simple algorithms: simple90, which used the same α=0.90 extension multiplier used by the prediction algorithm in the trials, and simple70, which used α=0.70 instead. The algorithms were simulated over all of the initial block cutting trials, and $err_{\text{simple90}}(t)$ and $err_{\text{simple70}}(t)$ were calculated. Figure 11-1 shows the reverse cumulative distribution of these errors compared to $err_{\text{pred}}(t)$.

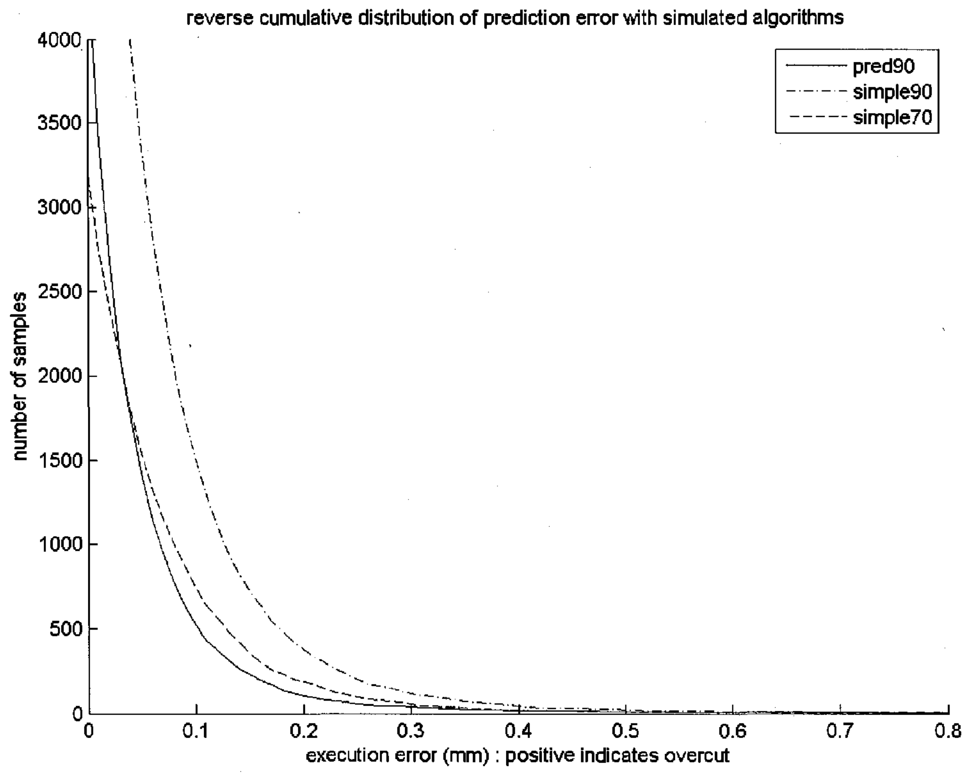


Figure 11-1. Reverse cumulative distribution of execution error caused by the prediction algorithm (pred90), and the simulated simple algorithms simple90 and simple70. The prediction algorithm has the smallest incidence of errors larger than 0.05mm.

Although simple70 caused slightly fewer errors in the range of 0-0.05mm, the prediction algorithm resulted in the fewest errors larger than 0.05mm.

11.2 Evaluating Algorithms Based on Cutting Efficiency

Although the prediction algorithm performed more accurately than both simple90 and simple70, the improvement over simple70 was small. This raises the question of whether the prediction information is useful, or whether perhaps the prediction algorithm just randomly retracts the blade by an amount equivalent to simple70. Cutting error can always be reduced by cutting more conservatively, at the expense of efficiency. We want to know that the prediction algorithm improves accuracy without sacrificing cutting efficiency.

Cutting efficiency can be defined as what percentage of the allowable blade extension the PFS actually extends. In other words, efficiency is the ratio between the actual blade extension at *t* and the allowable blade extension at *t*.

$efficiency(t) = b(t) / r(t)$

To study the efficiency of algorithms, we neglect the blade dynamics and substitute the commanded blade position for the actual blade position:

$eff_{\text{pred}}(t) = c_{\text{pred}}(t) / r(t)$
$eff_{\text{simple90}}(t) = c_{\text{simple90}}(t) / r(t)$
$eff_{\text{simple70}}(t) = c_{\text{simple70}}(t) / r(t)$

The efficiency of each algorithm was calculated for the initial block cutting trials, and the distributions are plotted below. As expected, simple90 and simple70 show efficiency peaks and 90% and 70%. The efficiency profile of the prediction algorithm closely matches simple90, proving that the prediction algorithm significantly increased performance over simple90 without sacrificing cutting efficiency.

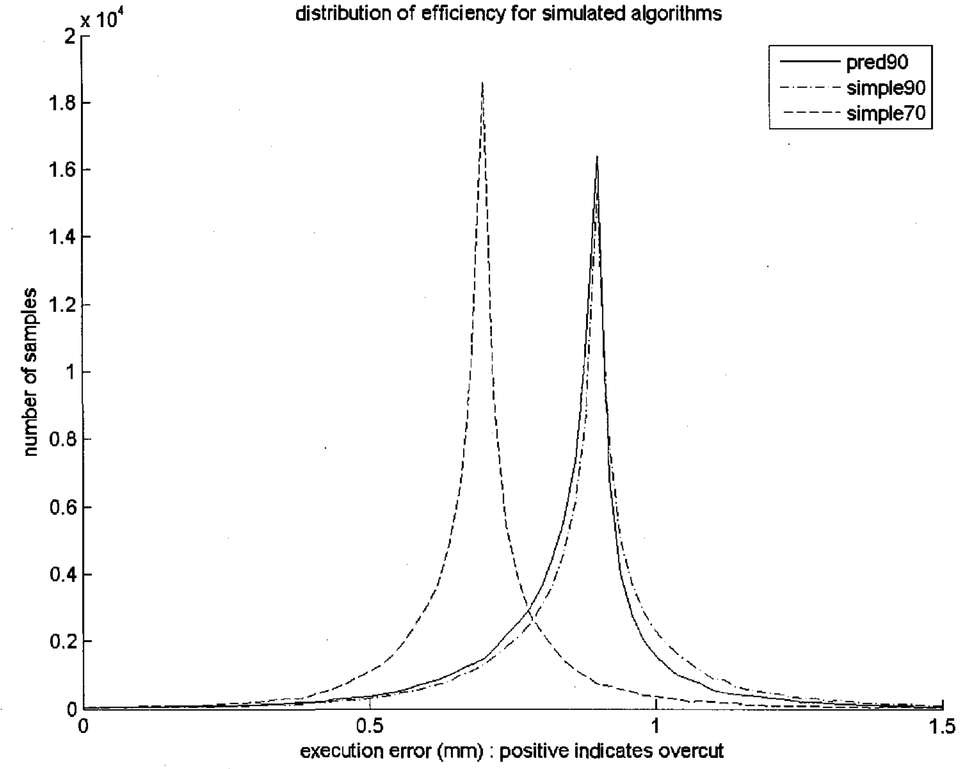


Figure 11-2. Distribution of cutting efficiency for the prediction algorithm (pred90), and the simulated simple algorithms simple90 and simple70.

Chapter 12. Conclusion

12.1 Contributions

The contributions of this work are the concept and development of the PFS. Specifically:

- PFS concept. A CAOS tool designed from the ground up for LIS.
- PFS implementation. Especially the algorithm for accurate cutting control.
- Proof of feasibility. Demonstration that desired accuracy can be achieved.
- Tools for understanding PFS. A theoretical framework for dealing with uncertainty in tool motion, and a taxonomy of potential sources of cutting error.

PFS Concept

Clinical Significance

Less invasive surgical techniques (LIS) are being developed for joint replacement surgery. These techniques have demonstrated improved short-term outcomes, but the techniques are more challenging, and risk inaccurate results. Since CAOS systems can deliver high cutting accuracy, it is a natural fit to use CAOS to enable LIS.

The PFS differs from most existing CAOS systems in that it was designed from the ground up for the goal of LIS. The long slender nose is much better suited than previous systems for operating through small incisions. The approach described in this thesis for implementing the long slender nose concept has advantages over simply adapting navigation or robot arm approaches.

Navigation is not well suited for implementing the PFS concept, because it would require the surgeon to continuously regulate the tool position. Any momentary error could cause a permanent mark in the finished surface.

Robot arm approaches could be easily adapted to use a long slender nose like the PFS. However, I feel that the PFS still offers more immediacy of manipulation than using a force-sensitive handle to control a robot arm. The PFS also offers larger range of motion than has so far been demonstrated by robot arm systems for CAOS. These advantages in manipulation can translate into real benefits for LIS by allowing the surgeon to more dexterously work among soft tissues.

Larger Significance

In the larger picture, the PFS is one example of work which exploits collaboration between a human and robot. For many complex tasks requiring human-level reasoning, autonomous robots are still far off. In the meantime, collaboration between human and robot can accomplish tasks that neither could complete alone, by combining the strengths of both. Collaboration can come in simple forms like teleoperation, but the coupling between the PFS and the user is more interesting and more intimate. This more intimate coupling is seen in semi-active CAOS systems in general. Another example is the Micron tremor reduction tool [Ang 2004], which was an inspiration for the idea of the PFS. Micron is a tool for micro-eye surgery, with a tip that deflects to cancel out hand tremor of the surgeon. Outside of the medical field, the work of Kazerooni combines robot strength with human perception and judgment in a very intuitive way. Examples include the Berkeley Lower Extremity Exoskeleton [Kazerooni 2006], or the Magic Glove [Kazerooni 2004], which is a force-sensing glove that allows the user to manipulate large and heavy loads in an intuitive fashion with robot assistance.

One principle that may make the PFS approach desirable for applications outside of orthopedic surgery is that sensing position is often cheaper than controlling position. For instance, a robot arm designed only to sense position does not require motors, and can be lightweight because it is not subjected to forces that can cause strain. As the workspace grows, the cost of a large milling machine or robot arm to control position becomes tremendous, while an optical tracking system can still cost-effectively measure position in a very large workspace. The PFS approach may be useful in material shaping applications where the workpiece is too large for most milling machines. Another advantage is that the PFS system is more portable than a large milling machine.

PFS Implementation

The key insight of PFS implementation is that the algorithm should first predict tool position, and then calculate the allowable blade extension from there. Further, the PFS implementation offers proof that it is computationally feasible to do so. Every 12ms, the PFS software considers the current and four future timesteps. For each timestep, it calculates the allowable blade extension in 5 candidate directions.

To achieve this computational efficiency, custom geometric algorithms were written that are specific to the shape of the guard and cutter. This includes the capsule-to-triangle first contact algorithm. Intersection detection algorithms have been developed for most common shapes in the graphics community, but the distance to first contact is a more specialized test.

Another important insight was the identification of the slabs data structure as appropriate. A heightfield is ideal for the PFS because it provides excellent resolution in the direction normal to the target shape. This is important for calculating accurate blade extension. The slabs implementation of heightfields is ideal because it maintains uniform spacing among the heightvectors, and because it renders well, without inverting.

Proof of Feasibility

The biggest challenge in developing the PFS was achieving the required cutting accuracy, specifically fit accuracy. The experimental results in this thesis demonstrate that the PFS can achieve accuracies on par with those required for cementless implants, and those achieved by conventional orthopedic techniques. We also suggested several improvements which should further improve cutting accuracy. This indicates that the PFS is worth developing further.

Tools for Understanding PFS

Two major tools were presented which aid in understanding the causes of cutting error in the PFS: the theoretical cutting error model (chapter 8), and the taxonomy of error sources (chapter 10).

In the theoretical error model of chapter 8, the workpiece slope determines how an error ε in predicted position corresponds to an error in predicted allowable extension. Therefore, if limits are known on the workpiece slope and the position prediction error, they determine a limit on error in allowable extension. The extension multiplier α can then be chosen so that the expected error in allowable extension can be tolerated without overcutting. This model is an approximation, but it is useful as a tool in understanding the sources of prediction error.

The error taxonomy of chapter 10 enumerates the potential sources of PFS cutting error. For designiners of PFS tools for new applications, the error taxonomy can serve as a checklist of error sources that must be accounted for. We also presented methods for estimating the contribution of each of these error sources in a real PFS system based on recorded cutting trials and benchtop experiments. For the current prototype, these results highlight the largest sources of cutting error, so that work on improving cutting accuracy can be focused efficiently. For those designing new PFS tools, these results can provide some information as to which error sources require careful attention, and which are probably not significant factors.

12.2 Future Work

Future work for the PFS consists of making the improvements suggested in this thesis to increase cutting accuracy. Additionally, development continues towards the goal of clinical tests and productization.

12.2.1 Suggested Improvements for Increased Accuracy

The cutting analysis found that the largest sources of error that could be corrected were the components of blade error: blade dynamics and blade latency. Blade dynamics was the largest source of large cutting error. Improving motor transient response can reduce the error. Using a better dynamic model for prediction could also limit the effect of blade dynamics on cutting error. Blade latency can be improved with better motion control hardware, and by commanding blade retraction as each predicted timestep is computed.

One other improvement which could have significant impact is the "virtual guard" described in Section 8.2.2, which was suggested by work on the theoretical cutting error model. The virtual guard regulates the workpiece slope based on the height of surrounding areas, in a more thorough way than the actual guard does by simply resting on the surrounding worksurface. This could result in a much smoother workpiece slope, which could reduce the demanded blade retraction speeds. This would have the effect of indirectly reducing the impact of blade dynamics error.

12.2.2 Ongoing Work

Since completion of the work described here, development has continued (without my involvement) at Blue Belt Technologies, a startup company. The PFS was able to piggyback onto two cadaver studies that were studying ACL biomechanics. The PFS was used on the knees after the ACL work was done. Neither trial was fully completed because of technical difficulties, but the PFS performed well up until failure.

The PFS was later evaluated by a surgeon on two pig legs. Material removal time was acceptable and the general reaction was positive. However, the PFS had trouble reaching the back of the tibia because there was not enough room in the joint even with the space opened by the distal femur cut. This is an example of a problem that could not be tested for using only the Sawbones setup. Blue Belt is developing a thinner tool to improve accesibility of the tibia, but using a smaller blade risks reducing material removal rate. Blades were located from another manufacturer that provide excellent material removal rate.

References

Ang W, Khosla P, Riviere C (2004) "Active tremor compensation in microsurgery", Proc 26th Int'l Conf Engineering in Medicine and Biology Society. 2738-2741.

Bach JM, Barrera OA, Kazanzides P, Haider H (2007) "Evaluation of the draft ASTM CAOS standard". 7th Annual Meeting of the International Society for Computer Assisted Orthopaedic Surgery, Heidelberg, Germany.

Bach CM et al (2002) "No functional impairment after Robodoc total hip arthroplasty: Gait analysis in 25 patients" Acta Orthop Scand 2002; 73 (4): 386–391

Bargar WL, Bauer A, Borner M (1998) "Primary and revision total hip replacement using the Robodoc system" Clinical Orthopaedics & Related Research. 354:82-91

Bathis H, Results of the BrainLAB CT-free navigation system in total knee arthroplasty. CAOS2003. p21.

Berger, RA, et al (2001), "Problems With Cementless Total Knee Arthroplasty at 11 Years Followup", Clinical Orthopaedics & Related Research. 392:196-207

Bonutti PM, Neal DJ, Kester MA (2003) "Minimal incision total knee arthroplasty using the suspended leg technique" Orthopedics 26(9):899–903

Brandt GZ et al (1999) "CRIGOS: a compact robot for image-guided orthopedic surgery" Information Technology in Biomedicine, IEEE Transactions on 3(4):252–260

Chassat F, Lavallée S (1998) "Experimental Protocol of Accuracy Evaluation of 6-D Localizers for Computer-Integrated Surgery: Application to four Optical Localizers" MICCAI 1998

Chung JH et al (2003) "Robot-Assisted Femoral Stem Implantation Using an Intramedulla Gauge" IEEE Trans Robotics and Automation 19(5):885-892

Chin PL, Yang KY, Yeo SJ, Lo NN (2005) "Randomized control trial comparing radiographic total knee arthroplasty implant" J Arthroplasty 20(5):618–26

Cleary K. IGST, The Book. 2007. http://www.igstk.org/papers/IGSTKTheBook.pdf

Cobb J (2004) "Robot assisted minimally invasive unicompartmental knee arthroplasty: results of first clinical trials" Computer Assisted Orthopaedic Surgery, 4th Annual Meeting of CAOS International.

Cristofolini L, et al (2007) "Increased long-term failure risk associated with excessively thin cement mantle in cemented hip arthroplasty: a comparative in vitro study", Clin Biomech, 22(4):410-21.

Crouch DG (2005) “Designing and Manufacturing Tools Incorporating IRED Markers” Northern Digital Inc, Waterloo ON, Canada

Cychosz JM, Waggenspack WN Jr (1994) “Intersecting a Ray with a Cylinder”, Graphics Gems IV, 356-365.

Dalton JE, Cook SD, Thomas KA, Kay JF (1995), “The effect of operative fit and hydroxyapatite coating on the mechanical and biological response to porous implants”, The Journal of Bone and Joint Surgery, 77(1):97-110

DiGioia AM, et al. HipNav: Pre-operative Planning and Intra-operative Navigational Guidance for Acetabular Implant Placement in Total Hip Replacement Surgery. Proc. of the Computer Assisted Orthopaedic Surgery Symposium, 1995

Forman RE, et al (2004) “Computer-Assisted Freehand Navigation for TKR”, CAOS2004 192-193

Haider H, Barrera OA, Garvin, KL (2007) “Minimally Invasive Total Knee Arthroplasty Surgery Through Navigated Freehand Bone Cutting” J Arthroplasty 22(4) 535-542

Harris SJ, Cobb J, Davies BL (1999) “Intra-operative Application of a Robotic Knee Surgery System” MICCAI'99

Heldreth, MA (2003) Method and apparatus for controlling a surgical burr in the performance of an orthopaedic procedure. US Patent Application 2004/0097948

Honl M et al (2003) “Comparison of robotic-assisted and manual implantation of a primary total hip” J Bone Joint Surg Am 85-A(8):1470–8

Horn BKP (1987) “Closed-form solution of absolute orientation using unit quaternions”, Journal of the Optical Society of America 4:629–642

Jagnow R, Dorsey J. (2002) *Virtual sculpting with haptic displacement maps*. Graphics Interface, 125--132

Jakoped M et al (2001) “The first clinical application of a "hands-on" robotic knee surgery system” Computer Aided Surgery 6(6):329-39

Jakopec M et al (2003) “The hands-on orthopaedic robot "acrobot": Early clinical trials of total knee replacement surgery” IEEE Trans Robotics and Automation 19(5): 902-911

Kazanzides P (1999) “Robot Assisted Sugery: The ROBODOC Experience” 30th Int’l Symposium on Robotics: 281-286

Kazerooni H, et al (2004) “The Magic Glove”, Int’l Conf on Robotics and Automation. 757-763.

Kazerooni H, R. Steger (2006) "The Berkeley Lower Extremity Exoskeletons", ASME Journal of Dynamics Systems, Measurements and Control, v128

Khadem et al (2000) “Comparative Tracking Error Analysis of Five Different Optical Tracking Systems” Computer Aided Surgery: 5(2), 98-107

Kienapfel H, Sprey C, Wilke A, Griss P (1999), “Implant fixation by bone ingrowth”, J Arthroplasty 14(3):355-68

Kienzle TC III, Stulberg, SD, Peshkin M, Quaid A, Wu C (1992) “An integrated CAD-robotics system for total knee replacement surgery” IEEE Trans Systems, Man, and Cybernetics 2: 1609-1614

Kneissler MH A. Matzig M. Thomale U.W. Lueth T.C. Woiciechowsky C. (2003) Concept and clinical evaluation of navigated control in spine surgery. Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on 2:1084–1089

Koulechov K, Strauss G, Richter R, Trantakis C, Lüth TC (2005) Mechatronical Assistance for Paranasal Sinus Surgery. CARS 2005 636–641

Koulechov K, Lüth T: A new metric for drill location for Navigated Control in navigated dental implantology. CARS 2004: 1220-1225

Labadie RF, Fitzpatrick JM. System and Method for Surgical Instrument Disablement via Image-Guided Position Feedback. US Patent Application 11/079,898. 2005.

Leitner, et al: Computer-assisted knee surgical total replacement, CVRMed-MRCAS’97

Levinson TJ, et al (2000) “Surgical Navigation for THR: A Report on Clinical Trial Utilizing HipNav,” MICCAI2000: 1185-1187

Lilikakis AK, Villar RN (2004) “Incisions great and small” J Bone Joint Surg Br 86(6):781–2

McReynolds, Tom, Organizer. “Programming with OpenGL: Advanced Techniques” SIGGRAPH ’97 course. p. 7-8.

Moore CAP M.A. Colgate J.E. (1999) “Design of a 3R cobot using continuous variable transmissions” Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on 4:3249–3254

Müller PE et al (2004) “Influence of minimally invasive surgery on implant positioning and the functional outcome for medial unicompartmental knee arthroplasty” J Arthroplasty 19(3): 296-301

Ogonda L et al (2005) "A minimal-incision technique in total hip arthroplasty does not improve early postoperative outcomes" J Bone Joint Surg Am 87(4):701–10

Otani T, Whiteside LA, White SE (1993), "Cutting errors in preparation of femoral components in total knee arthroplasty", J Arthroplasty 8(5):503-510

Pieck S, Gross I, Knappe P, Kuenzler S, Kerschbaumer F, Wahrburg J. (2003)
"A navigated mechatronic system with haptic features to assist in surgical interventions"
Comput Aided Surgery 8(6):292-9

Plaskos C et al (2005) "Praxiteles: a miniature bone-mounted robot for minimal access total knee arthroplasty" Intl J Medical Robotics and Computer Assisted Surgery 1(4):67-79

Price AJ et al (2001) "Rapid recovery after Oxford unicompartmental arthroplasty through a short incision" J Arthroplasty, 16(8): 970-976

Ranawat CS, Ranawat AS (2003) "Minimally invasive total joint arthroplasty: where are we going?" J Bone Joint Surg Am 85-A(11):2070–1

Rand JA, Coventry MB (1988), "Ten-Year Evaluation of Geometric Total Knee Arthroplasty", Clinical Orthopaedics & Related Research. 232:168-173

Sandborn PM, Cook SD, Spires WP, Kester MA (1988) Tissue response to porous-coated implants lacking initial bone apposition. J Arthroplasty 3(4):337–46

Simon D, Hebert M, Kanade T. Techniques for Fast and Accurate Intra-Surgical Registration. The Journal of Image Guided Surgery, Vol. 1, No. 1. - April 1995.

Søballe, Kjeld, Hansen, Ebbe Stender, Brockstedt-Rasmussen, Helle, Pedersen, Claus Møger and Bünger, Cody (1990) "Hydroxyapatite coating enhances fixation of porous coated implants: A comparison in dogs between press fit and noninterference fit", Acta Orthopaedica, 61(4): 299 – 306

Sparmann M, Wolke B, Czupalla H, Banzer D, Zink A (2003) "Positioning of total knee arthroplasty with and without navigation support" J Bone Joint Surg Br 85(6):830–5

St Erbse N Radermacher Rau (1998) "Design of a Passive Haptic Guidance System for Computer Assisted Surgical Interventions" Helmholtz-Institut Aachen Research Report 97/98

Stulberg SD, Loan P, Sarin V (2002) "Computer-Assisted Navigation in Total Knee Replacement: Results of an Initial Experience in Thirty-five Patients" J Bone Joint Surg Am. 84-A Suppl 2:90-8

Sun J, Smith M, Smith L, Nolte L-P (2005) "Simulation of an optical-sensing technique for tracking surgical tools employed in computer-assisted interventions" IEEE Sensors Journal 5(5):1127-1131

Taylor RH, et al (1994) “An image-directed robotic system for precise orthopaedic surgery” IEEE Trans Robotics and Automation 10(3): 261-275

Taylor R et al (1999) “Computer-integrated revision total hip replacement surgery: concept and preliminary results” Medical Image Analysis 3(3): 301-319

Toksvig-Larsen S, Ryd L (1994), “Surface characteristics following tibial preparation during total knee arthroplasty”, J Arthroplasty 9(1):63–6

Tria AJ (2003) “Advancements in minimally invasive total knee arthroplasty” Orthopedics 26(8 Suppl):s859–63

Troccaz J, Peshkin M, Davies B (1998) “Guiding systems for computer-assisted surgery: introducing synergistic devices” Med Image Anal 2(2):101–19

Turner MT, et al (1989), “Bone Ingrowth into the Tibial Component of a Canine Total Condylar Knee Replacement Prosthesis”, J Orthopaedic Res, 7:893-901

Villarreal MR (2007) “Human Male Skeleton”, Wikimedia Commons: commons.wikimedia.org/wiki/Image:Human_skeleton_front_no-text_no-color.svg

West JB, Maurer CR (2004) “Designing optically tracked instruments for image-guided surgery” IEEE Trans Med Imaging

Wickens CD (2000) Engineering Psychology and Human Performance, Chapter 4.

Wiesel U, Boerner M (2001) “First Experiences using a Surgical Robot for Total Knee Replacement” Computer Assisted Orthopaedic Surgery, 1st Annual Meeting of CAOS International

Wolf A, Lisien B, DiGioia AM III (2005) “MBARS: mini bone-attached robotic system for joint arthroplasty” The International Journal of Medical Robotics and Computer Assisted Surgery 1(2):101–121

Zimmer Inc (1997), “Nexgen Complete Knee Solution Intramedullary Instrumentation Surgical Technique”