Software Simulation of Numerically Controlled Machining

by

Gilad Israeli

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2006

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The field of numerically controlled (NC) machining has long been interested with predicting and measuring the errors in machining. Creating a simulation of NC machining is one way of achieving this. This thesis presents one such implementation of an NC simulation. It also runs a number of numerical and physical tests to verify the simulation's correctness. The numerical tests show that the simulator work correctly as well as providing guide lines for appropriate simulation parameters. The physical tests show that the results of the simulation match the results of real NC machines. It is hoped that this thesis can provide a guide for the creation of machining simulators and their verification.

Acknowledgements

First I want to thank my supervisors, Stephen Mann and Sanjeev Bedi, for their guidance, support and funding. I'd like to thank my readers, Edward Lank and Gregory Glinka, for their helpful comments and suggestions. I also want to thank Paul Gray for explaining a number of CNC machining concepts and Kevin Moule and Armando Roman for help with the physical experiments in the thesis. I want to thank the members of the CGL for their advice and support, and also for all the coffee hours. Finally I would like to thank my family for all their love and support.

Contents

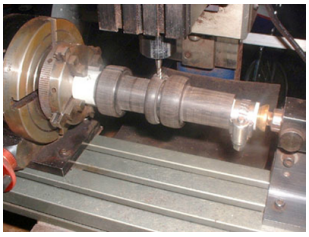
List of Tables

List of Figures

Chapter 1

Introduction

Numerically controlled (NC) machining was developed in the 1940s and 1950s in response to the requirements of jet powered aviation for precision parts with smaller tolerances for error [Wik06b]. The process of milling involves taking a piece of material, such as wood or metal, and turning it into a fancy table leg or a propeller blade. The NC machine does this by slowly carving the material using a spinning tool.



(a) A lathe

(b) A five axis machine

Figure 1.1: NC machines

Over time the control units for NC machines evolved from a simple mechanical or electronic controller to a microcomputer controller. This resulted in a better user interface, the ability to do limited tool path verification, as well as some awareness of the physical properties of the tool. Due to these advances most NC machines are now called Computer Numerically Controlled (CNC) machines. However even today's machines are only focussed on controlling their own motions. The machines are not aware of the final part, and only execute the tool path given to them. The machine will accept the tool path blindly, even if it will cause the machine to damage the resulting part or the machine itself.

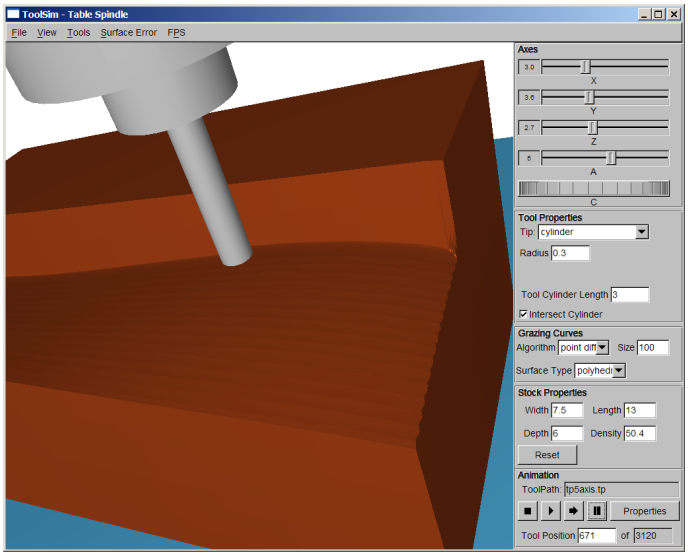
Another problem in CNC machining is that the parts milled by the machine are not exactly the same as the mathematical representation of the part. This is because the tool is not infinitely small and cannot pass directly over all areas of the part. Instead the field of CNC machining is interested in ensuring that this error is kept below some maximum error. CNC machining is also interested in computing the maximum error in a resulting part; however measuring this on a real part is difficult.

One solution to both of these problems is to design a simulator that would create a mathematical model of the milled part. This would allow us to predict what the machine will do when executing a tool path and prevent it from damaging itself or the part. Computing machining error from the results of the simulation is much easier than doing the same on a real part. Finally the simulator may even be able to interact with the CNC machine in real time, resulting in a better overall milling process.

1.1 The ToolSim Project

ToolSim, a simulation of CNC machining, was created by Stephen Mann to address the previously noted problems. Not long after ToolSim's creation I took over its development and maintenance. ToolSim can be used to simulate a variety of milling machine types. Through ToolSim, the user can control the machine interactively or load data for parts for the machine to mill. ToolSim displays the progress and results of the simulation using the 3D graphics library OpenGL. Afterwords the results can be analyzed within ToolSim, or exported for further analysis.

Figure 1.2: ToolSim in action



1.2 Objectives

The goal of this thesis is to prove that ToolSim provides a reasonably accurate simulation of parts machined on milling machines. I provide data on how to run ToolSim with simulation errors less than CNC machining error. I also note what software data structures worked well for the simulation, and which ones did not. I also show how to compute the accurate direction of tool motion from the model of the machine. This is needed to compute the intersection between the machine and the part. Finally I compare the results of milling with ToolSim and milling with a real machine to show that the results of the simulation correspond to the results of the real machine. It is hoped that this thesis can serve as a guide to other people building CNC simulators.

1.3 Outline

The next chapter gives some more extensive background on CNC machining, as well as previous work done in this area. Chapter 3 describes how ToolSim was designed and details its various parts. Chapter 4 describes the numerical tests that were run to verify that ToolSim worked correctly. It also analysed the error inherit in the simulation and how to set various parameters to minimize it. Chapter 5 details the comparison between surfaces that were milled on both the real CNC machines and the simulator. Chapter 6 summarizes the important results in the thesis, notes the limitations of those results, and outlines future work that could be done in this area.

Chapter 2

Background

Like all fields CNC machining has its own terms and concepts, which are covered in the first section of this chapter. In the second section of this chapter I review some previous work done on simulation of CNC machines.

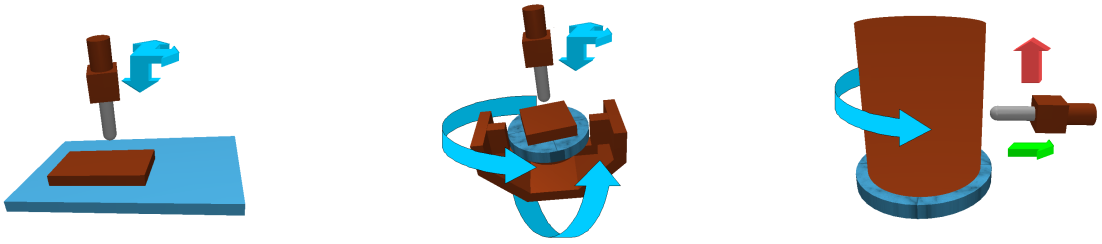
2.1 CNC Machining

There are many different types of CNC machines, however they all have some common features. All CNC machines have a component, called the *tool*, that mills through metal or wood, machining it into the desired part. The desired part's shape is called the *design surface*, while the metal or wood being machined is referred to as the *stock*. The machine can move the stock and/or tool along a number of *axes* in relation to one another. The series of positions a machine moves through to create the part is called a *tool path* and the process of creating the final part from the stock is called *milling*.

2.1.1 Machine Types

Although there are many different possible machine configurations, three common configurations are shown in Figure 2.1. The most common configuration in industrial applications is the three axis machine (Figure 2.1(a)). Here the stock moves along the x, y, z translation axes in relation to the tool. The five axis machine is a more flexible machine type, adding two rotation axes to the three axes configuration (Figure 2.1(b)). This allows for a greater number of tool orientations which can be used to reduce machining time. Unfortuantly tool path creation for a five axis machine is more complicated than for a three axis machines since the extra variables increase the possible ways the part could be damaged.

The CNC lathe is used for making cylindrical parts. While the lathe has three axes, most commonly it is configured so that only one axis is independently controlled. For example in the lathe shown in Figure 2.1(c), the motion of both the red and blue axes are tied to the motion of the green axis. Lathes in this configuration are referred to as single axis machines.



(a) Three axes (b) Five axes (c) Lathe

Figure 2.1: Machine types

2.1.2 Tool Types

The tools used by CNC machines are either drill bits made from hardened steel (Figure 2.2(a)) or components that contain inserts (Figure 2.2(b) and (c)). These inserts are what perform the cutting on tools that contain them and are usually made from hard materials such as tungsten, boron, or diamond. The difference between the two types is that the drill bit tools are cheaper and simpler but wear out faster, while the tools with inserts are more durable and last longer. The type of tool and the material it is made of depends on the machine type, as well as the type of stock, such as wood or steel.

(a) Drill bits

(b) Toridial tool

(c) Tungsten carbide tipped tool[1]

Figure 2.2: Machine tools

While machining, the tool spins much faster than the rate of motion of the tool relative to the stock. Therefore, when analysing tool motion it is common to treat the tool as if it were a *surface of revolution*, such as a sphere, cylinder, or torus. In the rest of this thesis when I refer to the tool, I am actually referring to the surface of revolution.

[1]Photograph taken by Glenn McKechnie on the 26th March 2005. This image is licensed under Creative Commons Attribution ShareAlike 2.0 License.

2.1.3 Tool Paths

Tool paths are a set of positions of the machine's axes, which are known as *machine coordinates*. The machine linearly interpolates these coordinates to move between these positions while machining the part. Tool paths are generally specified as a set of commands known as g-codes. Besides machine coordinates, *g-codes* also contain commands for setting tool properties or changing CNC machine settings [Wik06c]. While the syntax of the format is relatively consistent, the semantics of the codes can vary greatly between different manufactures of machines. An example of a tool path file with g-codes can be found in Figure 2.3.

```
G17 S849 T1 M67
G90 M3
G1 X 0 Y 0 Z 3 M8
G1 F135
G1 X -2.5506 Y 3.3808 Z -9.3362 B 4.7315 C 130.743
G1 X 13.6054 Y 5.6641 Z -5.761 B 47.9775 C -283.7976
G1 X 12.9397 Y -1.9475 Z -4.0253 B 47.9775 C -283.7976
G1 X 12.1885 Y -9.5556 Z -2.2081 B 47.9775 C -283.7976
G1 X 11.3497 Y -17.1641 Z -0.307 B 47.9775 C -283.7976
G1 X 10.4209 Y -24.7773 Z 1.6809 B 47.9775 C -283.7976
G1 X 9.399 Y -32.4008 Z 3.7593 B 47.9775 C -283.7976
```

Figure 2.3: Tool path — g-codes

Occasionally the machine controller is slightly more advanced and is aware of some of the properties of the machine and the tool. These machines can process tool path data that either contains the contact point of the tool with the stock instead of machine coordinates, or a mix of both contact point and machine coordinates. Some of the five axis machines manufactured by Deckel Maho Gildemeister support this enhancement. They call this M128 machining, named after the code that enables it. When this mode is enabled, each point along the tool path contains three pieces of data: the tool contact point with the surface, the normal of the surface at that point, and the tool axis. Then instead of interpolating machine parameters, it

interpolates the contact point, normal and tool axis. It then uses these parameters to compute the machine axis parameters. An example of a tool path with M128 mode machining data can be seen in Figure 2.4.

```
LN X0   Y42.99   Z41.65  NX-0.297285 NY-0.067536 NZ0.952397 TX-0.294399 TY-0.154236 TZ0.943154
LN X0   Y49.96   Z42.13  NX-0.290455 NY-0.065502 NZ0.954644 TX-0.287688 TY-0.152221 TZ0.945550
LN X0   Y57.1    Z42.61  NX-0.280401 NY-0.063989 NZ0.957748 TX-0.277767 TY-0.150722 TZ0.948751
LN X0   Y64.38   Z43.1   NX-0.267318 NY-0.062946 NZ0.961550 TX-0.264832 TY-0.149690 TZ0.952606
```

Figure 2.4: Tool path file with M128 data

2.2 Previous Work

The seminal work on real time simulation and display of CNC machines was published by Van Hook [Hoo86]. He used a z-map to represent the stock, which is similar though less flexible then the approach I will take.

The simulation presented in this thesis makes heavy use of swept surfaces, first introduced in CNC machining by Blackmore et al. [BLW92]. Initially swept surfaces were computed by solving differential equations, such as the Sweep-Envelop Differential Equation (SEDE) method used by Wang et al. [WLB97]. This thesis takes a different approach, using the work of Roth et al. [RBIM01] and Mann and Bedi [MB02]. Roth et al. [RBIM01] computed the swept surface of a toroidal tool using grazing points. These points were computed by partitioning the tool through its axis of revolution into a finite number of cross sections. The direction of motion of the insert and the insert's normal vector was used to compute the grazing point on the surface of the insert. The direction of motion of the insert was computed using the current and next position of the insert. Later, Mann and Bedi [MB02] generalized this idea to work with any surface of revolution.

Chapter 3

Software Design

This chapter describes the architecture and design of ToolSim. In the following sections I discuss the major components of ToolSim, how ToolSim computes its results, and how those results are rendered. I will also discuss alternative design options that were attempted, and some of the analysis ToolSim can perform on the results.

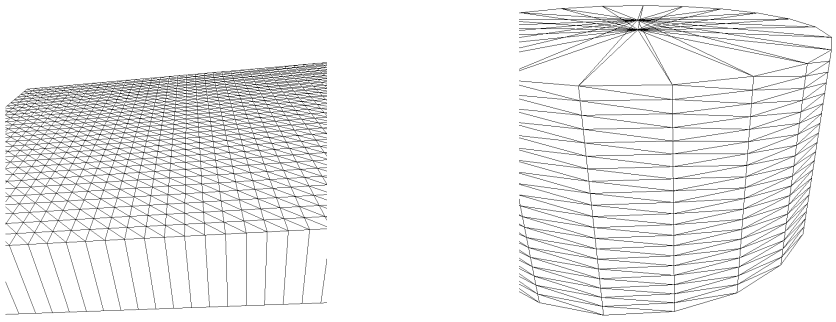
3.1 Components

The architecture of ToolSim can be grouped into three major components: the stock, the machine, and the tool path. Together these components are used to compute the parts resulting from milling. The tool path provides the machine positions used to create the part. The machine is responsible for computing the position and motion of the tool relative to the stock. Finally the stock is responsible for computing the intersection between the machine tool and the stock, as well as storing the result. Building the simulator from these different components allows us to change the machine, tool, tool path, and stock independently from one another.

3.1.1 The Stock

There are a number of ways to represent the stock. The straight forward way is with a three dimensional matrix, with each entry representing a discreet volume. However there are many challenges to successfully implementing this representation. One problem is that the memory cost of this representation grows very quickly with stock density and size. Other problems are that both the intersection of geometric shapes with volumes and the rendering of volumes are nontrivial.

Fortunately, most parts created by CNC machining can be represented by functional surfaces. This allows the use of a simpler representation such as a vertical height field on a regular grid, which is a grid of numbers representing the height of the stock at that point. These points are then tessellated into a surface of triangles to be rendered (Figure 3.1(a)). The height field can also be placed radially along a line to represent the cylindrical stocks cut by lathes (Figure 3.1(b)). Here the grid numbers represent the height of the stock along particular radii. In both cases, *stock density* is defined as the amount of height numbers per unit area of the grid. the density can be varied to achieve either faster or more accurate results.



(a) Vertical height field (b) Radial height field

Figure 3.1: Stock representation

3.1.2 The Machine

ToolSim uses a hierarchal model to represent the machine. This representation is used to render the machine and compute the position of the tool relative to the stock. These models are stored in a text file that is loaded at runtime. What follows is a summary of hierarchal and their use in ToolSim. For more details see [HB94][WNDS00].

A *hierarchal model* is a tree of nodes. In ToolSim's implementation, each node has a unique parent and zero or more children. There are never any cycles in the parent relationship. The node at the top of the tree, which has all the other nodes as its descendants, is called the root. Leaf nodes (that is those without children) contain objects that are visible, such as rendering primitives like spheres or cubes, or the stock. Internal nodes in the model contain affine transformations, such as translations, rotations and scales, which affect the position and size of the descendants of those nodes. These transformations may be dependent on machine coordinates or other machine or stock properties. Both leaf and internal nodes can contain commands related to the material and texture of the visible objects. The composition of the transformations of the leaf and all its ancestors up to the root gives the location of the leaf in world coordinates.

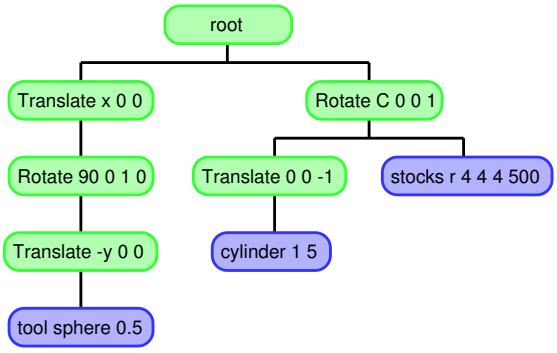
An example of a machine file can be seen in Figure 3.2(a). This is a simplified version of the lathe model that ToolSim uses. For the full specification of the machine file format see Appendix B.1. As mentioned before, machines can have any number of axes. By convention, an x y or z axis would be a translational axis, while the A or C axis would be rotational. After ToolSim loads the file it is converted into a hierarchal model (Figure 3.2(b)).

Using the hierarchal model, there is a method to compute the transformation

from one node to another. This algorithm is most commonly used by ToolSim to compute the transformation from the tool frame to the stock frame. To compute the transform from one node to another, move up from the starting node to the common ancestor of the two nodes and then move down to the destination node. During this traversal, the algorithm composes the transformations of all the nodes it passes through. However, the algorithm uses the inverse of the transform of the nodes it passes while moving down in the tree. This is because the transformations are specified for transforming points from the node's space to world space, but when an algorithm moves down the tree it is transforming points from world space to the node's space. This traversal is explored in further detail in Section 3.2.4.

```
push
      Rotate C 0 0 1

      push
            Translate 0 0 -1
            cylinder 1 5
      pop

      stock r 4 4 4 500
pop

Translate x 0 0
Rotate 90 0 1 0
Translate -y 0 0

tool sphere 0.5
```

(a) The model file



(b) Hierarchal model

Figure 3.2: A simple lathe

Tools

As mentioned before, the tool tip is spun at high velocity relative to the machine motion, and therefore can be treated as a surface of revolution. The shapes used by

ToolSim to represent the tool are the sphere, cylinder, cone, truncated cone, and torus. Other then the cylinder, the tools can optionally include a cylinderical shaft so the tool may be composed of two shapes. ToolSim uses the geometric representation of shapes to intersect the tools with the stock rays. The one exception is the torus, for which a polygonal representation is intersected with the stock. All tools are rendered using their polygonal representations.

3.1.3 The Tool Path

A tool path contains the positions that the machine will move through. The positions are loaded into ToolSim from a text file containing data in the same format used by real CNC machines (Section 2.1.3). In addition to this the file can also contain information about tool and stock parameters. See Appendix B.3 for full specification of the tool path file format.

ToolSim approximates the CNC machine's continuous motion with discreet time steps. To generate the motion between the positions given in the tool path file, ToolSim linearly interpolates the machine coordinates over time:

$$P(t) = P_i * (1 - t) + P_{i+1} * t, \tag{3.1}$$

for $t = t_0, t_1, \ldots, t_n$. These steps in between the tool positions are referred to as *in-between frames* or *in-between steps*. The direction of motion computation uses the derivative of the tool parameters with respect to time, and therefore uses

$$P'(t) = -P_i + P_{i+1}, \tag{3.2}$$

to compute the direction of motion of the machine at a particular step.

M128 Machining

ToolSim also supports M128 machining, where instead of machine axis data, tool contact point, tool axis, and surface normal are specified. In M128 machining ToolSim linearly interpolate these parameters over time instead of interpolating the axis data. Then the results of the interpolation are used to compute the axis data with these equations:

$$P_{xyz}(t) = r_2 * (norm(t) - axis(t)) + (r_1 - r_2)* \frac{norm(t) - (norm(t) \cdot axis(t)) * axis(t)}{\|norm(t) - (norm(t) \cdot axis(t)) * axis(t)\|} + pos(t) \tag{3.3}$$

$$P_A(t) = \arccos(axis_z(t)) \tag{3.4}$$

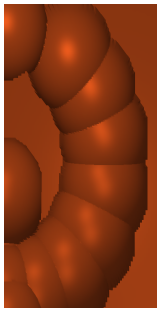
$$P_C(t) = \arctan(-axis_y(t)/axis_x(t)). \tag{3.5}$$

In Equation 3.3 the values of r_1 and r_2 depend of the type of tool being used. If the tool is a torus then r_1 and r_2 are the major and minor axis of the torus. A sphere tool can be considered a degenerate torus with r_2 being the radius of the sphere and r_1 being zero. If the tool is a cylinder then r_1 is the radius and r_2 is zero. For all other shapes this type of data input is undefined. The P_A and P_C axis can be computed by converting the tool axis, which is in rectangular coordinates, to spherical coordinates as is done in Equations 3.4 and 3.5.

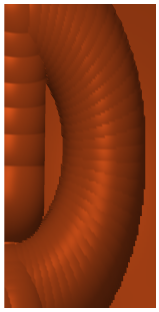
3.2 Computation

The main purpose of ToolSim is computing the results of machining. The simplest way to achieve this is to intersect the tool with the stock at all tool positions and in-between steps. This is referred to as *stamping*. However because this only

takes into account the discrete positions the tool moves through and not the continuous motion of the tool, this will result in high simulation error (Figure 3.3(a), unlike the more accurate result shown in Figure 3.4(b). The error can be reduced by increasing the number of in-between steps in the tool path, but this will require a very large number of stamps and will slow down the computation speed (Figure 3.3(b)). A better solution is to use swept surfaces.



(a) Large step size



(b) Small step size

Figure 3.3: Intersection using stamping

(a) Large step size

(b) Small step size

Figure 3.4: Intersection using swept surfaces

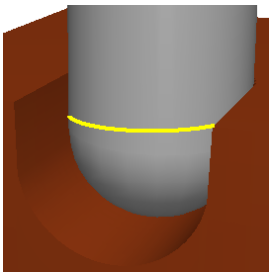
3.2.1 Swept Surfaces and Grazing Curves

A *swept surface* of a tool is the surface of the volume of space that the tool moves through. Intersection of the stock with a swept surface, instead of just using stamping, gives more accurate results without slowing down the computation speed. This improvement of the simulation can be seen by comparing Figure 3.3 to Figure 3.4.

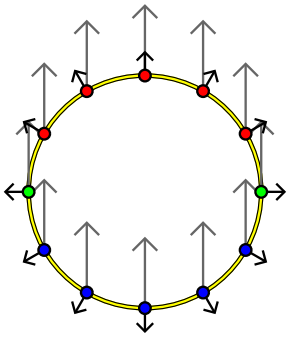
A swept surface generally has a complex shape. Consequently the representation I used for the swept surface is a piecewise polygonal surface. To construct this approximation we first note that the tool is always in contact with the swept surface.

This region of contact forms a curve on the surface of the tool. By connecting the points on these curves together we can create the approximation to the swept surface (Figure 3.5(c)). This surface representation gives a reasonable approximation to the swept surface. However if the path has any angular motion, the step size needs to be small to avoid high simulation errors (compare Figure 3.4(b) to Figure 3.4(a)). This step size will still be larger than the step size needed if only stamping is used.

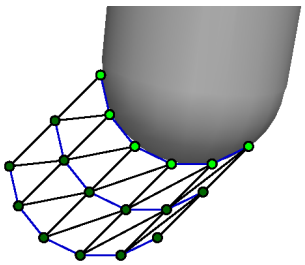
Now we just need to compute the curves that form the contact regions between the tool and swept surface. These curves are known as *grazing curves*. If we imagine the tool moving through the swept surface then the place where the tool touches the swept surface will aways be on the sides of the tool, relative to the direction of motion. This gives us a way to compute the grazing curves. More formally, the points in a grazing curve will be the points on the surface of the tool where the surface normal is perpendicular to the direction of motion of the tool [BLW92]. This is illustrated in Figure 3.5. To simplify the illustration we will only look at



(a) A cross section of the tool



(b) The grazing curve points on the cross section



(c) Swept surface created from the grazing curve points

Figure 3.5: Construction of a swept surface

a cross section of the tool. The yellow circle in Figure 3.5(b) is the cross section of the tool, shown in yellow, in Figure 3.5(a). In Figure 3.5(b) the grey arrows represent the direction of motion and the black arrows are the normals to the tool surface at that point. The red points are points along the surface of the tool that will soon be cut from the stock, while the blue points are points that have already been cut away. Only the points where the normal is perpendicular to the direction of motion, marked in green, will remain on the stock after the tool has moved through that tool position. After computing the location of the green points for other cross sections they are combined to create a piecewise linear approximation of the grazing curve (the blue lines in Figure 3.5(c)).

Sometimes the change in the direction of the tool motion is discontinuous. This can occur when the machine finishes moving into one tool position and begins to move into another. At this point there will be a big hole in the swept surface (Figure 3.6). To overcome this the tool is intersected with the stock at the point of discontinuity. In between machine positions the machine coordinates are linearly interpolated, so the change in direction is always continuous.

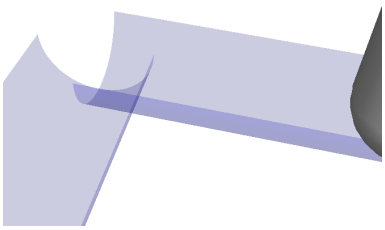


Figure 3.6: Swept surfaces resulting from a discontinuous tool path

3.2.2 Back-Edge Cutting

In CNC machining some tools have the ability to cut with either the front or back edge. Most of the time cutting is done with the front edge. While the back edge can also be used to cut, in most cases a back edge cut would indicated an error in the tool path. ToolSim supports milling with the back edge of the torus tool, but does not indicated when back edge cutting has happened and therefore cannot be used to automatically detect back edge cutting errors at this time.

3.2.3 Intersection

Since the stock is represented using height fields each line segment of the height field can be treated as a ray. These rays are then intersected with the shapes of the tool. The exceptions are the torus and swept surfaces, where the rays are intersected with their polygonal approximation. Algorithms for intersecting rays with geometric objects and triangles have been throughly investigated for ray tracing [Gla89]. If a ray intersects the shape, the location of the intersection is computed. If the location is lower than the value of the corresponding height field then the height field is updated with the new value. A bounding box is computed for each shape and triangle to avoid computing unnecessary intersections.

For the vertical height field stock I can perform a few more optimizations, since all the rays are pointing in the direction of the local z-axis. This allows me to perform the bounding tests in 2D. The fact that the rays are all pointing in the same direction and are placed on a regular grid means that 3D hardware can be used to compute the intersections. This would be achieved by rendering the tool shapes and swept surfaces and reading back the z-values from the z buffer [Gra02]. Currently ToolSim only uses the Graphics Processing Unit (GPU) for intersections

with B-spline surfaces (Section 3.4.1). One issue with using the GPU is that for most hardware today reading back information from the graphics card is quite slow. Whether the GPU version would actually be faster remains to be seen.

3.2.4 Direction of Tool Motion

As mentioned before the direction of tool motion in relation to the stock is required to compute the grazing curve. A simple way of approximating this direction is to store the previous position of a point on the tool and subtract it from the current position of that point. This is reasonably accurate. Unfortunately it does not scale well if the direction of motion of other points on the tool are required.

However the actual direction of motion can be easily computed using the transformation from the tool's affine frame to the stock's affine frame. If we take the derivative of this transformation with respect to time, it gives us another transformation that maps a point on the tool (in the tool's frame) to the point's direction of motion (in the stock's frame). This map gives us the accurate direction of motion.

The tool-to-stock map is computed from the machine's hierarchal model. Using the previously given model (Figure 3.2(b)) as an example the map would be

$$F(P(t)) = T\left(-P_y(t), 0, 0\right) \times R_y\left(90\right) \times T\left(P_x(t), 0, 0\right) \times R_x(P_C(t))^{-1}, \tag{3.6}$$

which is the result of using the traversal described in Section 3.1.2 to traverse the model from the tool node to the stock node. See Appendix A for the definition of T_x, R_x and R_y. To get the derivative of the map the product rule is used on Equation 3.6,

$$\begin{aligned} D_t^1 F(P(t)) = \; & D_t^1 T_x(-P_y(t)) \times R_y(90) \times T_x(P_x(t)) \times R_x(P_C(t))^{-1} \\ & + T_x(-P_y(t)) \times R_y(90) \times D_t^1 T_x(P(t)) \times R_x(P_C(t))^{-1} \\ & + T_x(-P_y(t)) \times R_y(90) \times T_x(P_x(t)) \times D_t^1 R_x(P_C(t))^{-1} \end{aligned} \tag{3.7}$$

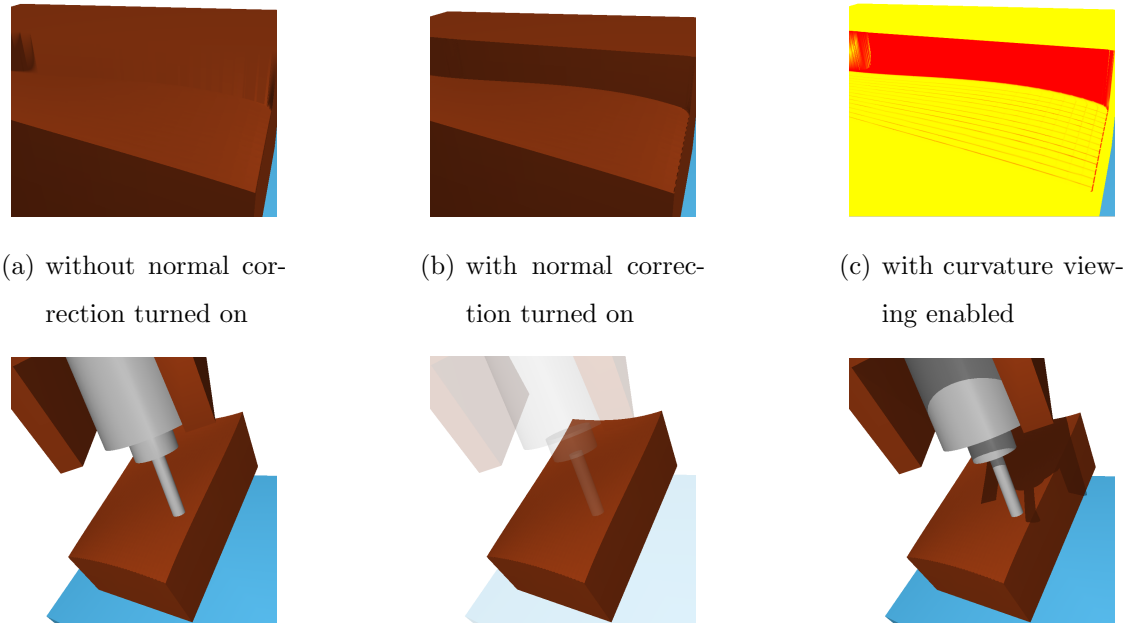
with $D_t^1 T$ and $D_t^1 R_x$ defined in Appendix A and $D_t^1 P(t)$ is Equation 3.2. Each term in Equation 3.7 corresponds to a factor in Equation 3.6. For each term the derivative of each successive factor is taken, skipping factors that do not depend on a machine coordinate (in this example, $R_y\,(90)^{-1}$). Summing all these factors gives the derivative [MBIZ].

3.3 Rendering

Although ToolSim easily allows computations with the stock, another benefit is being able visualize the results of machining. The stock height field data is triangulated (as seen in Figure 3.1) and rendered with the normals stored with each height field. This normal to the machined stock is the negation of the normal to the surface that cut away the stock at that point (Figure 3.7(a)). Optionally, if the difference between the stored normal and the surface normal of the triangulated surface is large, then the normal of the triangulated surface is used instead. This slows rendering, but lets the user see sharp edges more clearly (Figure 3.7(b)). Another feature that helps the user to visualize the stock data is the curvature display mode, which colours the stock depending on the angle between the stored normals of adjacent points (Figure 3.7(c)).

There are a few other features in ToolSim that aid usability. One is the option to have the machine drawn partially transparent so that it does not obscure viewing of the stock (Figure 3.7(e)). Another is the option to have the machine cast shadows on the environment. This helps the user see where the machine is relative to the stock (Figure 3.7(f)).

To render the machine, a depth first traversal of the hierarchal model is performed. Before the rendering algorithm traverses an internal node's children, it



(a) without normal correction turned on

(b) with normal correction turned on

(c) with curvature viewing enabled

(d) machine with no extra modes

(e) Transparent machine mode

(f) Shadows turned on

Figure 3.7: ToolSim GUI viewing options

saves the current transformation matrix on a stack and composes the node's transformation with the current transformation. When the algorithm passes a leaf node it draws the node's contents, which are transformed by the matrix at the top of the matrix stack. Finally when the algorithm finishes traversing an internal node's children, it pops the matrix off the previously mentioned stack and set that to be the current transformation matrix. This restores the current transformation to what it was before the algorithm traversed this node and its children [HB94][WNDS00].

The stock usually contains a lot of data and rendering can take quite some time. To speed this process up ToolSim uses *display lists* to render the stock. Display lists compile OpenGL commands so that the data can be sent quickly to the video card.

However, during machining ToolSim needs to repeatedly recompile the display list. This requires all the data and commands to be repeatedly resent to the video card, defeating the purpose of using display lists. To solve this problem the stock is split up into a grid of display lists, so while machining only a small area (usually 10%) of the stock needs to have its display list recompiled. Another optimization is letting the user control the frame rate. This way the user can have the display update at a rate they select, or not at all until the machining is finished. This prevents rendering from being a speed bottle neck.

3.4 Alternative Designs

This section deals with other ideas for ToolSim that where explored but not used, due to the costs being higher than the benefits.

3.4.1 B-Splines

A more accurate representation of swept surfaces was attempted by using B-spline surfaces. The control points of the surface in the first and last rows (red and blue in Figure 3.8) were fitted using least squares to the points of the grazing curve. The second row from the front (in purple) was computed using the direction of motion of the points on the first row and the third row (in dark blue) was similarly computed using the points from the last row. The B-splines were rendered using the OpenGL NURBS library.

Unfortunately, intersecting rays directly with a B-spline surface is quite complex and slow. The usual solution, implemented in ToolSim, is to triangulate the surface and then use that for intersection. This makes the accuracy of the algorithm similar

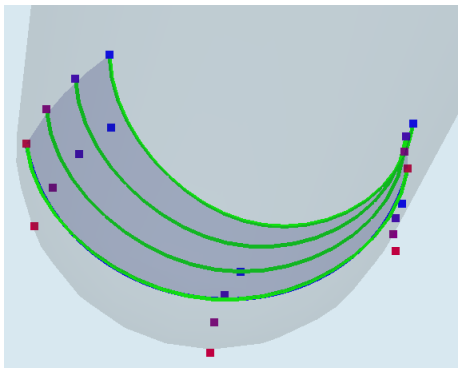


Figure 3.8: A B-spline swept surface with control points

to the one constructed in Section 3.2.1. The only benefit to using B-splines is that they use less memory to represent the grazing curves and swept surfaces. However, the memory used by the swept surface and grazing curves are marginal in comparison to the memory used by the stock, so the benefit is inconsequential.

3.4.2 Octrees

In ToolSim I experimented with volume representations of stocks by using an octree. An octree is a data structure that recursively divides space into eight equal segments. This allowed a volume representation with more reasonable memory cost, but made intersection and rendering even more complex. When intersecting with a discrete volumes we have to decide what to do when that volume is partially cut away. Also intersecting boxes with geometric shapes is more computationally expensive then intersecting rays. Finally figuring how to tessellate the volume into one or more surfaces of triangles and computing normals for that surface is also non-trivial. In the end octrees were not added to ToolSim because the issues and disadvantages out wighted the benefits.

3.5 Data Analysis

One of the goals in creating ToolSim was to make error analysis easier. Included in ToolSim there are number of ways of computing machining errors. ToolSim can estimate scallop heights and it can also compare the machined surface to a design surface. I will discuss the latter first.

3.5.1 Surface Comparison

ToolSim can load B-spline and ruled surface representations of design surfaces to compare with the results of machining. It supports various different methods of computing the error such as vertical height delta, closest point, closest triangle, and an approximation of closest triangle [LMB05]. For all of these error metrics it iterates over the stock or the design surface, repeatedly computing the error. When ToolSim is done computing the error it reports a few statistics on the errors, such as the minimum, maximum, and average error. Currently this feature is only implemented for vertical height fields.

The vertical height delta error is slightly different from all the other error metrics in that it iterates directly over the stock. It computes the difference between the height field and the point on the surface directly above it. All the other error metrics iterate over the design surface and try to find the closest point on the stock to measure against. Closest point takes a point on the surface and computes the error using the closest point on the stock. Closest triangle finds the closest point on the stock triangulated surface. This error metric exists in ToolSim for comparison with the approximation of closest triangle error metric. The approximation of closest triangle uses the surface normal to find the closest triangle, making it much faster but making the error estimate less accurate.

3.5.2 Scallop Height Estimation

Another analysis that ToolSim can perform is scallop height estimation. The *scallop height* is the height of the extra material that is not there on the design surface but is left behind on the stock after machining is complete (Figure 3.9). In ToolSim's scallop height estimation I assume that the lowest local point machined by the tool is on the design surface.

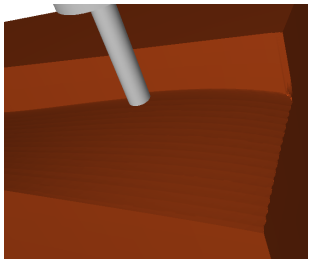
To compute the scallop height ToolSim first needs to find where the scallops are. To do this it iterates over the stock points in a direction perpendicular to the scallop lines (Figure 3.10(a)). First, using the current point, p_i ToolSim computes two vectors

$$v_i = p_i - p_{i-1} \tag{3.8}$$

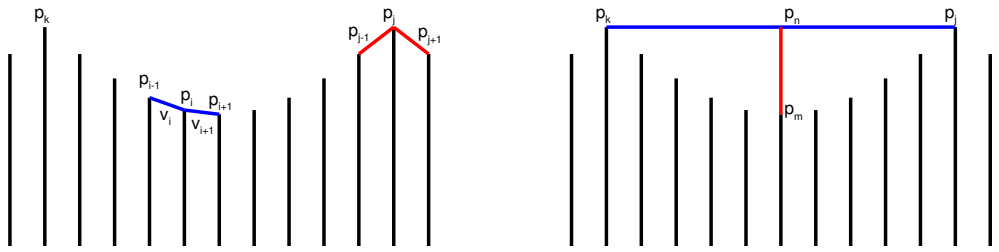
$$v_{i+1} = p_i - p_{i+1} \tag{3.9}$$

and then computes their dot product. For most of the surface these vectors are close to parallel so the dot product should be close to one. However when ToolSim reaches a point such as p_j in Figure 3.10(a), which is a scallop peak, then the surface is more discontinuous so the dot product is smaller. The scallop finding algorithm has a maximum iteration length, m_i, that defaults to the tool radius but

Figure 3.9: The scallops are the horizontal lines appearing on machined part of the stock



is modifiable by the user. Once m_i points have been tested, the algorithm selects the point that has an associated dot product with the smallest magnitude to be the next scallop.



(a) Finding scallops (b) Estimating scallop height

Figure 3.10: Intersection using swept surfaces

To compute the scallop height (Figure 3.10(b)) ToolSim computes the midpoint between the current and previous scallop peaks

$$p_n = \frac{p_k + p_{k+1}}{2}, \tag{3.10}$$

and use the middle point between the two peaks

$$p_m = p_{\frac{k+j}{2}}, \tag{3.11}$$

which should be in the middle of the valley between the scallops. Therefore scallop height is $|p_n - p_m|$. After this p_j becomes the current scallop and ToolSim repeats the scallop finding algorithm to compute the next scallop height. When it is finished scanning the stock it reports the maximum and average scallop heights.

The distance measure used to compute the scallop height can have a large error when the slope of the line between p_k and p_j is high. This error measure is the same

as the vertical error metric, the shortcoming of which are further discussed in Section 4.1.2. Since the design surface may be complicated or unknown, implementing a more accurate error measurement is difficult. However most real design surfaces do not have steep sides so this error measure should be a reasonable approximation.

There were no formal test done to verify that the scallop detection algorithm works. However using the curvature view mode of ToolSim I could check if the detected scallop were located on the actual scallops (Figure 3.11). This feature was requested by one of the mechanical engineering students and he was satisfied with the results of the computation.

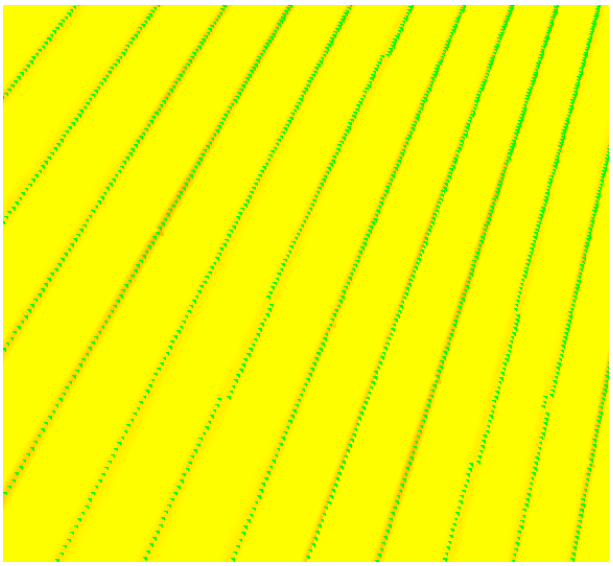


Figure 3.11: Checking that the scallop detection algorithm works — the red areas show locations, such as scallops, with higher discontinuities in the surface normals

Chapter 4

Numerical Verification of ToolSim

While working on ToolSim it was important to know that the simulation was working correctly. Simple visual inspection of the results could pick out the most glaring errors. Grazing curves were tested to check if the points were actually where the surface normal was perpendicular to the direction of motion. Stocks between two versions of the program were compared to check if the results had changed. Unfortunately if the computation actually changed this check did not indicate if the change was correct.

Deriving full proofs of code and algorithms is possible but tedious and error prone. An easier solution is to validate ToolSim's correctness based on a series of numerical tests, which are described in Section 4.1. The basic idea of the test is to compute the error between a machined stock and a design surface. As noted in Section 4.1.3 there is a theoretical bound on the resulting error. If the error does not stay within these bounds then there is likely an error in the implementation. If the error stays within these bounds then we should have confidence that the implementation is correct.

Computing error bounds for our simulation can prove useful in other ways as well. ToolSim implements an accurate direction of motion algorithm in addition to the usual approximation used (Section 3.2.4). I can test the implementation by checking if it produces a lower error than the approximation. This comparison will also tell us how much of an improvement the accurate direction of motion algorithm is over the approximation of the direction of motion.

Another issue was that the selection of a number of parameters in the simulation such as grazing curve density or stock density was arbitrary. Experience suggested that these parameters were related. By varying these parameters between test runs and seeing their effect on the error results I can determine the relationship between these parameters.

Finally when surfaces are milled on real machines the errors are kept below a specific tolerance. If we want to compare ToolSim's results with that of real CNC machines we have to also be able to bound the error in the simulation. This error is due to a number of approximations in the simulation and is affected by parameters such as stock density, grazing curve density and tool path in-between-steps. Using the results from the numerical tests we can establish the relationship between simulation error and these parameters and adjust them to bound the error (Section 4.2.2).

In short, the goals of this chapter are:

- Verification of ToolSim
- Adjusting parameters of the simulation to ensure simulation results are within certain engineering tolerances
- Analysis of the effectiveness and correctness of the accurate direction of motion algorithm

4.1 Test Overview

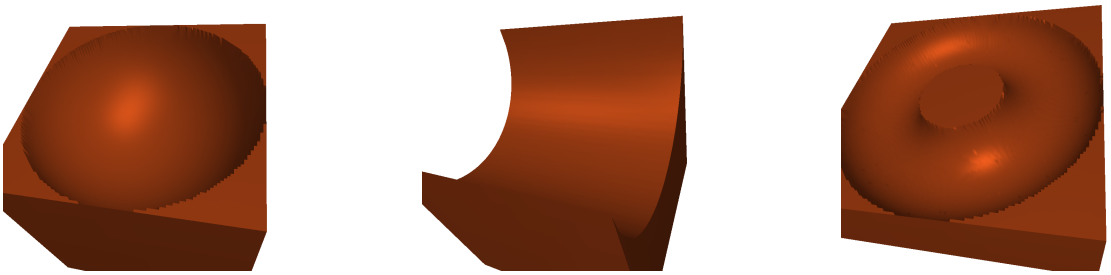
Running a numerical test involves simulating machining of a simple surface and then comparing the machined stock to the design surface. Computing this comparison over the entire surface gives us the maximum error. If we only compare the height field with the design surface then the error should be on the order of floating point precision. The realization is that the machined stock's height field representation is not a set of points in space but a piecewise linear approximation of a surface. By comparing the surface at points in between the points on the stock height field grid we can get a sense of the error in our approximation.

However it is not that simple. There are a number of different types of design surfaces, error metrics, and linear approximations of the machined stock that can be use in computing the simulation error. The following sections look into each of those issues.

4.1.1 Design Surfaces

I used three different design surfaces for my tests: hemisphere, half-cylinder and half-torus. A spherical tool was used to machine all these surfaces. For the hemisphere surface, the machine stamped the centre of the stock (Figure 4.1(a)). For the half-cylinder surface, the machine swept the tool horizontally along the x-axis through the stock (Figure 4.1(b)). For the half-torus surface the tool was placed on an angle and rotated around the vertical axis (Figure 4.1(c)).

Each design surface exists to test different aspects of the simulation. The hemisphere tests stamping, while the half-cylinder and half-torus test grazing curves. The half-cylinder design surface test grazing curves that are axis aligned and have



(a) Hemisphere (b) Half-cylinder (c) Half-torus

Figure 4.1: Verification tests

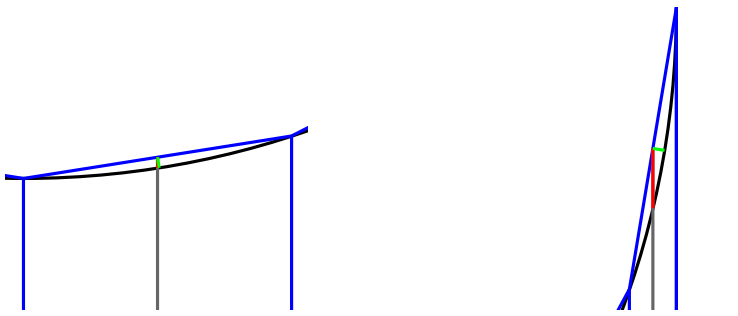
no rotational motion, while in the half-torus the grazing curves are not axis aligned. The tests using these surfaces also test different simulation variables, as noted in Section 4.2.

4.1.2 Error Metrics

There are a number of ways to measure the error between two surfaces. The simplest way is to measure the difference between the height field and the z-value of the surface at that point. This is simple to compute because both the stock and design surface are functional. Another advantage is that there is a theoretical error bound on this metric. This bound is further discussed in Section 4.1.3, and it will become important when I want to verify that ToolSim works correctly.

Although vertical error is simple and has a theoretical error bounds, the actual error values it provides may not be very accurate. When the sampling plane is roughly parallel to the xy-plane, error values should be accurate (Figure 4.2(a)). However, when the normals of the sample plane and the xy-plane have a higher angle between them the sampled error becomes much bigger than the real error (compare the difference between the red line, which is vertical error, and the green

line which is the actual error in Figure 4.2(b)). Therefore if we want accurate error numbers, we need a more accurate error measure.



(a) Low angle (b) High angle

Figure 4.2: Non uniform vertical sample error

A better error measure would be to compare the sample point with the closest point on the design surface. Since the sample planes are small and our design surfaces are simple and well behaved, we can consider them functions over the local sample plane. Computing this error for the hemisphere and half-cylinder is simple as seen in Equations 4.1 and 4.2. These equations compute the distance between the sample point and the centre of the sphere (or closest point on the cylinder's axis) and compare it to the radius:

$$Error_{hemisphere} = \left| \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} - radius \right| \tag{4.1}$$

$$Error_{half-cylinder} = \left| \sqrt{(y - y_0)^2 + (z - z_0)^2} - radius \right| \tag{4.2}$$

where (x, y, z) is the sample point and (x_0, y_0, z_0) is the centre of the sphere. For the cylinder, (x_0, y_0, z_0) is the closest point on the cylinder's axis. Since it is aligned with the x-axis, as seen in Figure 4.1(b), y_0 and z_0 are set to the centre of the cylinder and is set x_0 to x.

Computing the closest point distance for the torus is slightly more complicated.

$$Error_{half-torus} = \left| \sqrt{(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2} - radius_{minor} \right| \tag{4.3}$$

The steps to compute Equation 4.3 are shown in Figure 4.3. Instead of using the centre of the torus we need to use the closest point on the circle running through the centre of the torus tube. We will call this the local centre. If we think of a plane going through the z-axis and the sample point, then the tube embeds a circle on this plane, which we can use to compare the distance. To find the circle we compute $dir = \frac{(x,y,0)-(x_0,y_0,0)}{||(x,y,0)-(x_0,y_0,0)||}$ which is a vector in the direction of the circle's centre from the torus origin. Once we have that computing the local centre is easy:

$$(x_c, y_c, z_c) = dir \cdot radius_{major} + (x_0, y_0, z_0).$$

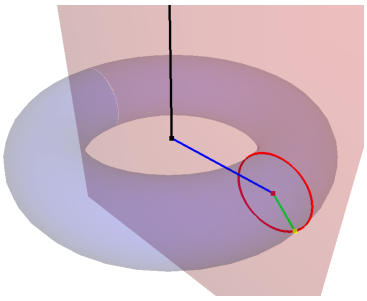


Figure 4.3: Computing the closest point on the torus — the yellow point is the sample point (x, y, z), the red point is the local centre (x_c, y_c, z_c), the blue line is dir, and the black line is the z-axis.

4.1.3 Error Bounds

When using interpolated data it is useful to have a bound on the error. If we know the design surface f of the sample data then we can compute an error bound on the piecewise linear interpolation. As long as our design surface is C^2 then the error bound is

$$\|f - s\| \leq Mh^2, \tag{4.4}$$

where f is the design surface, s is the approximation function, $h = \max\{h_x, h_y\}$, h_x is the maximum distance between adjacent sample points in the x direction, and h_y is similarly defined in the y direction. The value of M is

$$M = 4 \cdot \max_{i+j=2} \left\| D_x^i D_y^j f \right\|, \quad \forall i, j \ 0 \leq i, j \leq 2, \tag{4.5}$$

when linear interpolation is used (These error bounds are from [Pre75]). With this error bound if the density of s is doubled then the error given by Equation 4.4 should drop by a factor of four, due to the h^2 term. Therefore we repeatedly run the tests and double the density — if we see the error drop by a factor of four then we can gain confidence that the simulation works properly.

One minor complication is that the constant M depends on the derivatives of the design surface f. As you approach the edge of the hemisphere, half-cylinder, or half-torus, the derivatives approach infinity. This results in unbounded error near the edge. This also gives a mathematical reason why the vertical error metric becomes a poor measurement of error along the edges of our shapes (Figure 4.2(b)). To avoid this, the tests that used vertical error excluded data along the edge of the shape. For example for the hemisphere only the area within $0.6 * radius$ of the centre sample was used. For the half-cylinder it is the area within $0.6 * radius$ of the cylinder axis and for the half-torus it is the area within $0.6 * radius_{minor}$ of the circle in the middle of the torus tube. I refer to this area as the *range of interest*.

4.1.4 Sample Surfaces

Since ToolSim uses linear approximations, I can simulate the error by comparing the design surface with a linear interpolation of the sample points of the machined stock. To generate the points on this sample surface, I can use bilinear interpolation on a rectangular grid or linear interpolation on a triangular grid. In the test I ran, I used linear interpolation because it is more general and easier to use.

One issue with the sampling surface is what happens when a sample triangle straddles the sharp edge of the design surface. Along this edge the design surface only has C^0 continuity (Figure 4.1). When using the vertical error metric this violates the C^2 requirement of our error bounds. Therefore for this error metric I skipped any triangles lying on a discontinuity. This does not affect the results when using the vertical error metric since the goal with this metric is to verify that the simulation works properly. When analyzing the surface with closest point error metric, data from triangles on discontinuities were considered separately. This was done because not all design surface have discontinuities and the results were quite different depending on if the design surface was continuous or not.

4.1.5 Density Variables

There are many different densities used in ToolSim and in the tests in this chapter. The *stock density* is the number of height points per unit area of the stock representation mentioned in Section 3.1.1. The *grazing curve density* is the number of points used to construct the grazing curve. The *sampling density* is the number of points I compute per unit area when interpolating the stock values to estimate the errors. A sampling density of one only samples at the stock points, where as a sampling density of two or higher would also sample between the stock points.

Finally the *in-between-steps* are the number of positions between each tool position. This was later discovered to be related to the grazing curve density (Section 4.2.1).

Grazing Curve Density

One important thing to note is that the grazing curve density used in ToolSim is not a real density but the number of points use to create the grazing curve. In exploring the relationship between grazing curve density and stock density it is useful to have similar units for comparison. That requires using an actual density, which involves computing the length of the grazing curve.

There are two ways of computing the length of the grazing curve. One is the *parameterized grazing curve length* which is the length of the grazing curve on the tool. For the sphere this length is πr. The other way of computing the length is using the *length of the projected grazing curve*. This is the length of the grazing curve after it has been projected onto the plane of the stock. This would be the area the curve cuts out of the stock. For the sphere this length is $2r$. This calculation becomes more complicated for other tool types since the length would vary depending on either the tool motion for parameterized length, or on tool orientation for projected length.

In the tests I ran I used projected grazing curve length. Since a sphere tool of radius 5 is used in all the tests, the projected grazing curve length is 10 in all cases.

4.2 Results and Analysis

A total of ten tests were run, the first nine of which pared each design surface (Section 4.1) with each of the error metrics. The three error metrics were vertical error, closest point on triangles within the continuous parts of the design surface, and closest point on triangles lying on the discontinuous parts of the design surface. The tenth test was closest point without the triangles lying on the discontinuous parts of the design surface, with the half-torus design surface, and the accurate direction of motion algorithm.

For all the tests using the hemisphere as the design surface, the maximum error was computed repeatedly while sampling density and stock density were varied. For the test using the half-cylinder and half-torus, the maximum error was computed repeatedly while grazing curve density and stock density were varied to see their effect on the error. Table 4.1 contains the parameters used with each design surface type when they were sampled using the vertical error metric and closest point on triangles within the continuous area. The closest point on triangles lying on discontinuities tests used the same parameters except that the sample density for the hemisphere test was set to the values $\{1, 2, 4, 8, 16, 32\}$. The sampling density for the half-torus and half-cylinder tests with closest point on triangles lying on discontinuities was set to 8.

Table 4.1: Parameters used in the tests for each design surface

	hemisphere	**half-cylinder**	**half-torus**
stock width	10 mm		30 mm
stock length	10 mm		30 mm
stock depth	10 mm		
tool	sphere		
tool radius	5 mm		
sampling	linear		
sampling density	$\{1, 2, 4, 8\}$	2	
stock density	{1, 2, 4, 8, 16, 32, 64, 128}		{1, 2, 4, 8, 16, 32, 64}
grazing curve density	NA	{1, 2, 4, 8, 16, 32, 64, 128}	
angle step	NA		$4 * density_{\text{grazing curve}}$

In addition to the test data, there are tables that present the ratio between the columns and rows of the tests using the vertical height error. This makes it easier to see the effect of the h^2 term on Equation 4.4 from Section 4.1.3.

4.2.1 Verification

Before I could run the verification tests, I first had to determine what sampling density I should use. To find the optimal sampling density, sampling density is one of the parameters I varied while running the test using the hemisphere. If we look at Table 4.2, then we see that increasing the sampling density beyond two does not change the error. Therefore for the sake of speed, we will use a sample density of two for the verification tests. The results of these tests are displayed in Table 4.4.

The simplest verification test used the hemisphere, which only involved stamping and not swept surfaces. Therefore the only factor affecting s in Equation 4.4 is the stock density. If we look at the the ratio between the columns of Table 4.2 (illustrated by Table 4.3), then we can see that the error decreases by a factor of four when the stock density is doubled, verifying that ToolSim works correctly when only stamping is used.

For the cylinder and torus test when we doubled both the grazing curve and stock density we also saw a factor of four decrease (Tables 4.5(c) and 4.6(c)). If we only increased one of them there was a more limited improvement (Tables 4.5(b), 4.5(a), 4.6(b), and 4.6(a)) implying a relationship between these parameters. This will be further explored in Section 4.2.1.

Table 4.2: Maximum Error (mm) — Hemisphere Test with vertical error metric and 0.6 range of interest

		stock density							
		1	2	4	8	16	32	64	128
sample density	1	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
	2	0.0939004	0.0225692	0.0058032	0.0014800	0.0003772	0.0000949	0.0000238	0.0000059
	4	0.0939004	0.0225692	0.0058032	0.0014800	0.0003772	0.0000949	0.0000238	0.0000059
	8	0.0939004	0.0225692	0.0058032	0.0014800	0.0003772	0.0000949	0.0000238	0.0000059

Table 4.3: Analysis of hemisphere test data — ratio between successive stock densities

		stock density$_i$/stock density$_{i+1}$						
		d_1/d_2	d_2/d_4	d_4/d_8	d_8/d_{16}	d_{16}/d_{32}	d_{32}/d_{64}	d_{64}/d_{128}
sample density	1	undefined	undefined	undefined	undefined	undefined	undefined	undefined
	2	4.16	3.88	3.92	3.92	3.97	3.98	3.99
	4	4.16	3.88	3.92	3.92	3.97	3.98	3.99
	8	4.16	3.88	3.92	3.92	3.97	3.98	3.99

Table 4.4: Maximum Error (mm) — with vertical error metric and 0.6 range of interest

(a) Half-cylinder Test

		stock density							
		1	2	4	8	16	32	64	128
grazing curve density	1	0.1187823	0.0894227	0.0894227	0.0894227	0.0894227	0.0894227	0.0894227	0.0894227
	2	0.0495521	0.0297029	0.0206271	0.0206271	0.0206271	0.0206271	0.0206271	0.0206271
	4	0.0434738	0.0132945	0.0072835	0.0050696	0.0050696	0.0050696	0.0050696	0.0050696
	8	0.0396484	0.0117982	0.0036838	0.0016805	0.0012278	0.0012278	0.0012278	0.0012282
	16	0.0389957	0.0107988	0.0029091	0.0009255	0.0004528	0.0003018	0.0003018	0.0003018
	32	0.0388863	0.0108109	0.0029092	0.0007882	0.0002412	0.0001129	0.0000746	0.0000749
	64	0.0388534	0.0107687	0.0028660	0.0007497	0.0002030	0.0000621	0.0000299	0.0000184
	128	0.0388418	0.0107640	0.0028599	0.0007404	0.0001910	0.0000516	0.0000155	0.0000073

(b) Half-torus Test

		stock density						
		1	2	4	8	16	32	64
grazing curve density	1	0.1363228	0.1037322	0.0995873	0.0980008	0.1006394	0.1009843	0.1009843
	2	0.0989301	0.0381199	0.0267250	0.0247300	0.0247325	0.0249510	0.0250009
	4	0.0948554	0.0248746	0.0104826	0.0067159	0.0061857	0.0062443	0.0062604
	8	0.0938597	0.0241302	0.0071245	0.0026564	0.0016853	0.0015559	0.0015559
	16	0.0935855	0.0235213	0.0061061	0.0017751	0.0007034	0.0004313	0.0003908
	32	0.0935451	0.0234599	0.0060048	0.0015778	0.0004581	0.0001829	0.0001064
	64	0.0935345	0.0234636	0.0059931	0.0015431	0.0003973	0.0001150	0.0000486
	128	0.0935345	0.0234562	0.0059931	0.0015270	0.0003852	0.0001012	0.0000305

Table 4.5: Analysis of half-cylinder test data

(a) ratio between successive grazing curve densities — (each cell $t_{i,j}$ of this table is $d_{i,j}/d_{i,j+1}$ where d is a cell from Table 4.4(a))

		stock density							
		1	2	4	8	16	32	64	128
$\frac{\text{grazing curve density}_i}{\text{grazing curve density}_{i+1}}$	d_1/d_2	2.39	3.01	4.33	4.33	4.33	4.33	4.33	4.33
	d_2/d_4	1.13	2.23	2.83	4.06	4.06	4.06	4.06	4.06
	d_4/d_8	1.09	1.12	1.97	3.01	4.12	4.12	4.12	4.12
	d_8/d_{16}	1.01	1.09	1.26	1.81	2.71	4.06	4.06	4.06
	d_{16}/d_{32}	1.00	0.99	0.99	1.17	1.87	2.67	4.04	4.02
	d_{32}/d_{64}	1.00	1.00	1.01	1.05	1.18	1.81	2.49	4.05
	d_{64}/d_{128}	1.00	1.00	1.00	1.01	1.06	1.20	1.92	2.51

(b) ratio between successive stock densities — (each cell $t_{i,j}$ of this table is $d_{i,j}/d_{i+1,j}$ where d is a cell from Table 4.4(a))

		stock density$_i$/stock density$_{i+1}$						
		d_1/d_2	d_2/d_4	d_4/d_8	d_8/d_{16}	d_{16}/d_{32}	d_{32}/d_{64}	d_{64}/d_{128}
grazing curve density	1	1.32	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.66	1.43	1.00	1.00	1.00	1.00	1.00
	4	3.27	1.82	1.43	1.00	1.00	1.00	1.00
	8	3.36	3.20	2.19	1.36	1.00	1.00	0.99
	16	3.61	3.71	3.14	2.04	1.50	1.00	1.00
	32	3.59	3.71	3.69	3.26	2.13	1.51	0.99
	64	3.60	3.75	3.82	3.69	3.26	2.07	1.61
	128	3.60	3.76	3.86	3.87	3.70	3.32	2.10

(c) ratio between successive stock and grazing curve densities — (each cell $t_{i,j}$ of this table is $d_{i,j}/d_{i+1,j+1}$ where d is a cell from Table 4.4(a))

		stock density$_i$/stock density$_{i+1}$						
		d_1/d_2	d_2/d_4	d_4/d_8	d_8/d_{16}	d_{16}/d_{32}	d_{32}/d_{64}	d_{64}/d_{128}
$\frac{\text{grazing curve density}_i}{\text{grazing curve density}_{i+1}}$	d_1/d_2	3.99	4.33	4.33	4.33	4.33	4.33	4.33
	d_2/d_4	3.72	4.07	4.06	4.06	4.06	4.06	4.06
	d_4/d_8	3.68	3.60	4.33	4.12	4.12	4.12	4.12
	d_8/d_{16}	3.67	4.05	3.98	3.71	4.06	4.06	4.06
	d_{16}/d_{32}	3.60	3.71	3.69	3.83	4.00	4.04	4.02
	d_{32}/d_{64}	3.61	3.77	3.88	3.88	3.88	3.77	4.04
	d_{64}/d_{128}	3.60	3.76	3.87	3.92	3.93	4.00	4.06

Table 4.6: Analysis of half-torus test data

(a) ratio between successive grazing curve densities — (each cell $t_{i,j}$ of this table is $d_{i,j}/d_{i,j+1}$ where d is a cell from Table 4.4(b))

		stock density						
		1	2	4	8	16	32	64
$\frac{\text{grazing curve density}_i}{\text{grazing curve density}_{i+1}}$	d_1/d_2	1.37	2.72	3.72	3.96	4.06	4.04	4.03
	d_2/d_4	1.04	1.53	2.54	3.68	3.99	3.99	3.99
	d_4/d_8	1.01	1.03	1.47	2.52	3.67	4.01	4.02
	d_8/d_{16}	1.00	1.02	1.16	1.49	2.39	3.60	3.98
	d_{16}/d_{32}	1.00	1.00	1.01	1.12	1.53	2.35	3.67
	d_{32}/d_{64}	1.00	0.99	1.00	1.02	1.15	1.58	2.18
	d_{64}/d_{128}	1.00	1.00	1.00	1.01	1.03	1.13	1.58

(b) ratio between successive stock densities — (each cell $t_{i,j}$ of this table is $d_{i,j}/d_{i+1,j}$ where d is a cell from Table 4.4(b))

		$\text{stock density}_i/\text{stock density}_{i+1}$					
		d_1/d_2	d_2/d_4	d_4/d_8	d_8/d_{16}	d_{16}/d_{32}	d_{32}/d_{64}
grazing curve density	1	1.31	1.04	1.01	0.97	0.99	1.00
	2	2.59	1.42	1.08	0.99	0.99	0.99
	4	3.81	2.37	1.56	1.08	0.99	0.99
	8	3.88	3.38	2.68	1.57	1.08	1.00
	16	3.97	3.85	3.43	2.52	1.63	1.10
	32	3.98	3.90	3.80	3.44	2.50	1.71
	64	3.98	3.91	3.88	3.88	3.45	2.36
	128	3.98	3.91	3.92	3.96	3.80	3.30

(c) ratio between successive stock and grazing curve densities — (each cell $t_{i,j}$ of this table is $d_{i,j}/d_{i+1,j+1}$ where d is a cell from Table 4.4(b))

		$\text{stock density}_i/\text{stock density}_{i+1}$					
		d_1/d_2	d_2/d_4	d_4/d_8	d_8/d_{16}	d_{16}/d_{32}	d_{32}/d_{64}
$\frac{\text{grazing curve density}_i}{\text{grazing curve density}_{i+1}}$	d_1/d_2	3.57	3.88	4.02	3.96	4.03	4.03
	d_2/d_4	3.97	3.63	3.97	3.99	3.96	3.98
	d_4/d_8	3.93	3.49	3.94	3.98	3.97	4.01
	d_8/d_{16}	3.99	3.95	4.01	3.77	3.90	3.98
	d_{16}/d_{32}	3.98	3.91	3.86	3.87	3.84	4.05
	d_{32}/d_{64}	3.98	3.91	3.89	3.97	3.98	3.76
	d_{64}/d_{128}	3.98	3.91	3.92	4.00	3.92	3.76

As mentioned before, while studying the previous error data, I noticed relationships between the grazing curves density and stock density, and between the tool path in-between-step size and grazing curves density.

Grazing Curve and Stock Density

Looking at Tables 4.6(a) and 4.5(a) one can see that increasing the grazing curve density without increasing the stock density produces diminishing returns. Similarly increasing the stock density without increasing the grazing curve density produces zero returns after some point (Tables 4.6(b) and 4.5(b)). This implies that we want grazing curve density per unit to be equal to stock density per unit. This makes sense as our data cannot be more accurate then our sensors (the stock) nor our approximation (the grazing curve).

ToolPath Step Size

In running the experiments I found that if I used a constant number of steps between tool positions then I would not get the factors of four in Table 4.6(c), and the error did not tend to improve as it did for the other tests. I realized that this was because the distance between steps affects the simulation in the same way that the grazing curve density does, but in the direction of motion of the tool, as opposed to perpendicular to the direction of motion. By setting the number of steps to be

$$4 * density_{\text{grazing curve}},$$

I was able to get the factor of four improvements. The reason I used four times the grazing curve density is because

$$\frac{\text{tool path length}}{\text{grazing curve length}} = \frac{2\pi r_{major}}{\pi r_{minor}} = \frac{2 * 10}{5} = 4,$$

for the particular grazing curve and tool path lengths used in the torus test.

4.2.2 Error Tolerance

To compare ToolSim data with real machined surfaces and use it to predict the result of machining, we need to minimize the simulation error. *Simulation error* is the error in relation to the real machined surface introduced by all the various approximations in ToolSim. CNC machines have a similar property called machine tolerance. *Machine tolerance* is the maximum amount of error a machine will introduce when machining. If the engineers want to machine a surface with machine tolerance, $\epsilon_{machine}$, then we need to ensure that the simulation results do not have error larger then $\epsilon_{machine}$. Therefore we need the simulation error, $\epsilon_{simulation}$, to be bounded by $\epsilon_{machine}$.

As noted in Section 4.1.4 we ignore sampling triangles that lie along discontinuous edges of the surface for verification, since we do not care about the actual error values. However for estimating simulation error we want to know the maximum possible error. While many machined surface have sharp edges like the edge of the hemisphere, half-cylinder or half-torus (Figure 4.1), others are smooth continuous surfaces. Therefore I analyzed the edge error separately from the errors in the continuous part of the surface. This will allow for more optimal settings depending on the type of surface being machined.

To compute the simulation error I computed the maximum error with closest point for both the continuous area (Table 4.7) and discontinuous edges (Table 4.8). When looking for stock density resulting in simulation error less than a certain amount, I would select an equivalent grazing curve density, the reasons for which are noted in Section 4.2.1. This is the same as looking along the diagonal in tables

4.7(b), 4.7(c), 4.8(b), and 4.8(c). Looking at the data one can see that occasionally smaller grazing curve density can be used to achieve similar results. However the space saved by using a sparser grazing curve will be negligible since the stock uses much more memory in either case. Also there should be no difference in speed since we are intersecting the same number of stock points. Therefore it is easier to be consistent and keep the stock density and grazing curve density equivalent.

Unlike the vertical error metric or the closest point error metric on triangles over the continuous part of the surface, the hemisphere test for closest point error metric on triangles lying on discontinuous parts of of the surface saw increases in maximum error with sampling densities beyond a sampling density of two. Therefore with closest point error metric on triangles lying on discontinuities I used a sampling density of 8. Higher sampling densities would give marginally more accurate results, but would take much longer to compute.

A final question is how to combine the results of tests using the three design surfaces in order to calibrate the stock and grazing curve densities. All tool paths use both stamping and swept surfaces, so we would use the higher of the resulting stock densities we got from looking at the stamping and sweeping test. However we have two different tests with swept surfaces, one with only translation (the half-cylinder) and one with only rotation (the half-torus). In most tool paths the magnitude of the translational motion is much bigger then that of the rotational motion. Therefore most of the time the swept surface error would be closer to that of the half-cylinder test. Alternatively one could view the results of the half-cylinder and half-torus as a minimum to maximum range of possible stock densities depending on the tool path.

Table 4.7: Maximum Error (mm) — with closest point error metric, continuous part of the design surface, and full range of interest

(a) Hemisphere

		stock density							
		1	2	4	8	16	32	64	128
sample density	1	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
	2	0.1183888	0.1076167	0.0490868	0.0311530	0.0175903	0.0079918	0.0046920	0.0025467
	4	0.1183888	0.1076167	0.0490868	0.0311530	0.0175903	0.0079918	0.0046920	0.0025467
	8	0.1183888	0.1076167	0.0490868	0.0311530	0.0175903	0.0079918	0.0046920	0.0025467

(b) Half-cylinder

		stock density							
		1	2	4	8	16	32	64	128
grazing curve density	1	0.1098295	0.0759612	0.0759612	0.0759612	0.0759612	0.0759612	0.0759612	0.0759612
	2	0.0617903	0.0349963	0.0216694	0.0170775	0.0170775	0.0170775	0.0170775	0.0170775
	4	0.0521595	0.0255834	0.0134242	0.0077298	0.0050336	0.0040550	0.0040550	0.0040550
	8	0.0509702	0.0239107	0.0115425	0.0060514	0.0031589	0.0017906	0.0012048	0.0009883
	16	0.0504215	0.0233245	0.0112282	0.0056186	0.0028037	0.0014862	0.0007688	0.0004303
	32	0.0502988	0.0231832	0.0111728	0.0055077	0.0027344	0.0013864	0.0006992	0.0003712
	64	0.0502624	0.0231456	0.0111429	0.0054763	0.0027174	0.0013574	0.0006793	0.0003458
	128	0.0502543	0.0231383	0.0111325	0.0054640	0.0027088	0.0013503	0.0006747	0.0003381

(c) Half-torus

		stock density						
		1	2	4	8	16	32	64
grazing curve density	1	0.3448235	0.2825455	0.1553315	0.1062850	0.0976063	0.0943459	0.0941679
	2	0.3509356	0.1695926	0.1068851	0.0676081	0.0380390	0.0247375	0.0243683
	4	0.3547222	0.1720569	0.1068851	0.0411849	0.0328073	0.0188841	0.0109492
	8	0.3596230	0.2825455	0.1545382	0.0516933	0.0262078	0.0109138	0.0072672
	16	0.3611218	0.2825455	0.0871232	0.0676081	0.0377759	0.0128007	0.0065207
	32	0.7959403	0.2825455	0.0879242	0.0516933	0.0217383	0.0128007	0.0093921
	64	0.7959403	0.2825455	0.0881344	0.0516933	0.0262078	0.0128007	0.0071895
	128	0.7959403	0.1779653	0.1068851	0.0438238	0.0377759	0.0188841	0.0071895

Table 4.8: Maximum Error (mm) — with closest point error metric, discontinuous part the design surfaces, and full range of interest

(a) Hemisphere

		stock density							
		1	2	4	8	16	32	64	128
sample density	1	0.8309518	0.5901699	0.2974050	0.1250000	0.0655854	0.0315411	0.0164767	0.0079039
	2	0.8309518	0.5901699	0.2974050	0.1345185	0.0660674	0.0327297	0.0173101	0.0086853
	4	0.8309518	0.5901699	0.2974050	0.1509442	0.0765863	0.0377057	0.0193002	0.0096681
	8	0.8309518	0.6128340	0.2974050	0.1592084	0.0818556	0.0403417	0.0206121	0.0103485
	16	0.8309518	0.6348004	0.3047968	0.1633532	0.0844927	0.0416603	0.0212683	0.0106888
	32	0.8309518	0.6458165	0.3099315	0.1654289	0.0858118	0.0423197	0.0215964	0.0108589

(b) Half-cylinder

		stock density							
		1	2	4	8	16	32	64	128
grazing curve density	1	0.2690998	0.1402599	0.0758600	0.0737256	0.0497934	0.0280902	0.0148373	0.0076158
	2	0.2584845	0.1303416	0.0644932	0.0348097	0.0169582	0.0169582	0.0120594	0.0069222
	4	0.2566349	0.1276168	0.0631462	0.0321170	0.0158291	0.0086867	0.0040496	0.0040496
	8	0.2567358	0.1268279	0.0629107	0.0315839	0.0156713	0.0079870	0.0039226	0.0021706
	16	0.2566467	0.1266391	0.0629117	0.0314026	0.0156519	0.0078748	0.0039082	0.0019923
	32	0.2565978	0.1266129	0.0629068	0.0313597	0.0156534	0.0078323	0.0039094	0.0019671
	64	0.2565850	0.1266058	0.0628992	0.0313519	0.0156518	0.0078214	0.0039089	0.0019569
	128	0.2565835	0.1266037	0.0628963	0.0313489	0.0156504	0.0078195	0.0039083	0.0019541

(c) Half-torus

		stock density						
		1	2	4	8	16	32	64
grazing curve density	1	1.3236397	0.6622118	0.3311830	0.1878804	0.1063994	0.0925577	0.0910745
	2	1.3236397	0.6622118	0.3311830	0.1656088	0.0898639	0.0486482	0.0282778
	4	1.3236397	0.6622118	0.3311830	0.1656088	0.0841989	0.0426855	0.0221004
	8	1.3236397	0.6622118	0.3311830	0.1656088	0.0828085	0.0414052	0.0211257
	16	1.3236397	0.6622118	0.3311830	0.1656088	0.0828085	0.0414052	0.0207028
	32	1.3236397	0.6622118	0.3311830	0.1656088	0.0828085	0.0414052	0.0207028
	64	1.3236397	0.6622118	0.3311830	0.1656088	0.0828085	0.0414052	0.0207028
	128	1.3236397	0.6622118	0.3311830	0.1656088	0.0828085	0.0414052	0.0207028

Industry standard machine tolerance for machining metal is 1/1000 of an inch

or 0.0254 mm. Machining wood has higher tolerance, between 4/1000 and 5/1000 of an inch or 0.1016 mm and 0.127 mm. The next two sections look at what stock densities are needed to keep simulation error bellow these machine tolerances.

Simulation Error for Metal Milling

First we will look at the case of continuous surfaces (Table 4.7). The first stock density with less 0.0254 mm simulation error is approximately 16 (Table 4.7(a)). For the swept surfaces a simulation error of less than 0.0254 mm requires a stock with a density between 4 and 32, depending on how much rotation the tool path contains (Tables 4.7(b) and 4.7(c)). If we merge the minimum stock densities for both swept surfaces and stamping then the density should be between 16 and 32, with 16 being likely sufficient for most tool paths.

On the other hand if the design surface is discontinuous then for stamping we achieve a simulation error of less then 0.0254 mm when the stock density is approximately 64 (Table 4.8(a)). For the swept surfaces getting a simulation error of less then 0.0254 mm requires a stock with a density between 16 and 64 (Tables 4.8(b) and 4.8(c)). Therefore overall the stock density should be set to 64.

Simulation Error for Wood Milling

We can derive the stock densities required for wood milling in the same way we did for metal milling. For continuous surfaces and machine tolerance of 0.127 mm the stock densities between 2 and 4 are sufficient. For discontinuous surfaces stock densities between 8 and 16 are recommended.

4.2.3 Direction of Motion

The motion of the tool is required to compute the intersection between the machine and the stock. The simple way to achieve this is to take the difference between the current and previous positions. The correct way to do this is to compute the motion using the machine model, as described in Section 3.2.4. An easy test to check if this method works is to check if it produces less error then simple motion.

I reran the torus test with closest error point and direction of motion (Table 4.9). Then I took all the values computed using simple motion and closest error (Table 4.7(c)) and computed the ratio between corresponding entries (Table 4.10). A ratio less then one means that the accurate direction of motion is more accurate. As we can see in the table this is generally the case. As noted in Section 4.2.1 grazing curve unit density and stock density should be equivalent. This occurs along the centre diagonal of the table. Along this area the ratio varies between 0.9 and 0.5 or anywhere between 10% to 50% improvement in the error.

Table 4.9: Torus Maximum Error (mm) - with closest point error metric and accurate direction of motion

		stock density						
		1	2	4	8	16	32	64
grazing curve density	1	0.2954824	0.2245506	0.170207	0.1106765	0.0992665	0.0990946	0.094927
	2	0.3308003	0.1537302	0.089868	0.0575804	0.0398248	0.0297081	0.027193
	4	0.3481001	0.1673421	0.080817	0.0385323	0.0238633	0.0141007	0.009961
	8	0.3597602	0.1756529	0.086482	0.0426447	0.0210070	0.0103087	0.006084
	16	0.3597406	0.1756070	0.086416	0.0426436	0.0210045	0.0103209	0.005239
	32	0.3630478	0.1779668	0.088112	0.0438105	0.0218271	0.0108933	0.005431
	64	0.3622773	0.1774093	0.087722	0.0435435	0.0216432	0.0107501	0.005328
	128	0.3629832	0.1779107	0.088075	0.0437959	0.0218190	0.0108757	0.005419

Table 4.10: Ratio between accurate direction of motion and simple motion (Tables 4.9 and 4.7(c))

		stock density						
		1	2	4	8	16	32	64
grazing curve density	1	0.85	0.79	1.095	1.04	1.01	1.05	1.00
	2	0.94	0.90	0.840	0.85	1.04	1.20	1.11
	4	0.98	0.97	0.756	0.93	0.72	0.74	0.90
	8	1.00	0.62	0.559	0.82	0.80	0.94	0.83
	16	0.99	0.62	0.991	0.63	0.55	0.80	0.80
	32	0.45	0.62	1.002	0.84	1.00	0.85	0.57
	64	0.45	0.62	0.995	0.84	0.82	0.83	0.74
	128	0.45	0.99	0.824	0.99	0.57	0.57	0.75

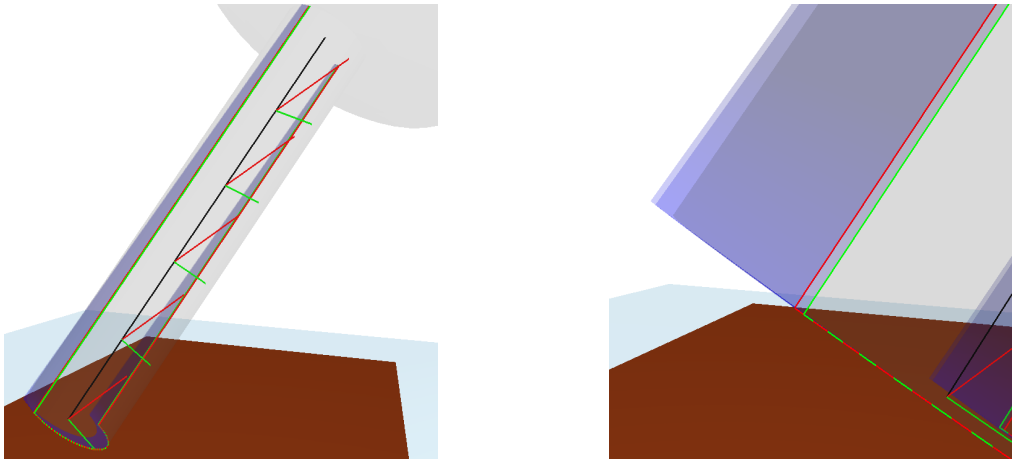


Figure 4.4: The grazing curves computed by the direction of motion algorithms — the curve generated by the approximate algorithm is in red, while the curve generated by the approximate algorithm is shown in green. Also shown are vectors representing the direction of motion of the tool along the axis. The colours of the vectors correspond to the same algorithm as the colours for the grazing curves.

Figure 4.4 shows a tool with grazing curves generated by both the approximate alogorithm (in red) and the exact algorithm (in green). While the direction vectors computed by the algorithms seem quite different, they are both approximately in a plane that is perpendicular to the normal of the surface of the cylinder where the grazing points are, producing similar grazing curves. Figure 4.4 also contains a closeup of the grazing curves showing that they are not exactly the same.

The improvement for actual machining would depend highly on the toolpath. These tests only measured the error on the continuous area of the surface. In the discontinuous area the errors from the discontinuity dominated and there was no difference between the accurate and approximate direction of motion.

4.3 Summary

The experiments in this chapter have provided a number of useful results. They have verified that the simulation works correctly by showing that the error in the simulation has $O(h^2)$ convergence. They have noted appropriate stock densities given a machining tolerance and surface type. Finally the experiments showed that the accurate direction of motion algorithm produces a 10% to 50% improvement in the error.

The stock densities recommended by this chapter are listed in Table 4.11. The recommended stock density also depends on the amount of angular motion in the tool path, hence the range of densities, with straighter tool paths requiring less density. ToolSim ran most test in less than one second, though the torus test with higher stock densities could take as long as a minute. The memory used up by the tests depends on the stock density. The largest would be the torus test with stock that has a width and hight of 30mm and a density of 64, which used 112.6

Mb of memory. This stock had 1921 by 1921 sample points. For a more reasonable example of memory usage and virtual milling speed see Section 5.3.

Table 4.11: Summery of recommended stock densities

	continuous surfaces	**discontinuous surface**
metal milling	16-32	64
wood milling	2-4	8-16

Chapter 5

Physical Verification of ToolSim

ToolSim is not just a theoretical research project but is also driven by actual applications. A number of times mechanical engineering students would have their surfaces machined virtually on ToolSim before machining it on an actual CNC machine. ToolSim was used by Kaplan et al. in [KBM+04] to assist in tool path planing (Figure 5.1). Many of ToolSim's features, such as M128 machining, back-edge cutting, scallop detection, radial stocks, and the CNC lathe model were added to meet the needs of the users of ToolSim.

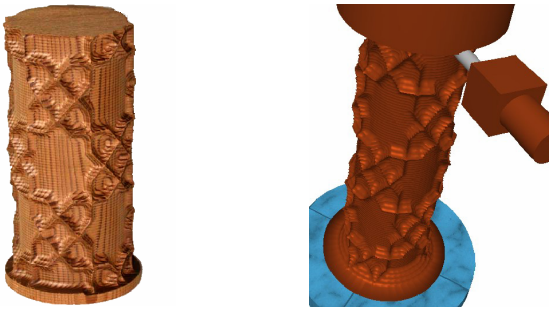


Figure 5.1: The real and virtual surface machined for [KBM+04].

While simulating the CNC machining for the mechanical engineers, occasionally the results would be unexpected. However in many cases the simulation was correct and the fault was somewhere else. There was one particular case of a machine that was being built that had trouble correctly milling a tool path. It was assumed this problem was due to a misalignment in the machine. However, after the simulation produced the same results as the real machine, it was found that the error was actually in the tool path.

In addition to anecdotal evidence of the correlation between the simulated and real machining, I wanted to verify it formally. In the previous chapter I showed that ToolSim was mathematically consistent. In this chapter, I formally verify that the results of the simulation correspond to the results of real machining. To achieve this I machined some surfaces on real machines and compared them to surfaces machined by the simulator. By checking the results I can show that the simulation is a reliable predictor of the outcome of milling.

Gray, et al. [GPI+05] preformed a similar experiment using a single axis lathe. The work here builds on that but uses three and five axis tool paths instead. Also the lathe uses the radial stock, for which we did not verify or develop stock density guidelines, unlike the three and five axis tool path that are machined on stock represent by the rectangular vertical height field, for which we do have guidelines.

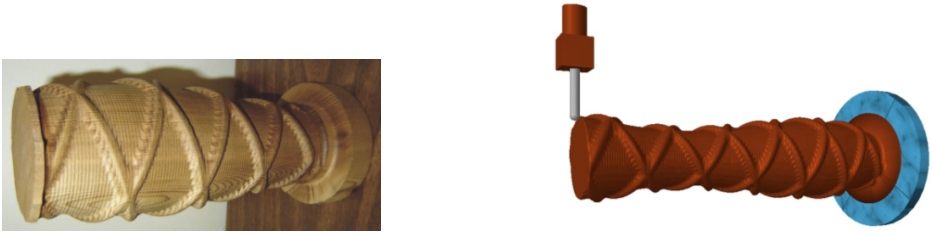


Figure 5.2: The real and virtual surface machined for [GPI+05].

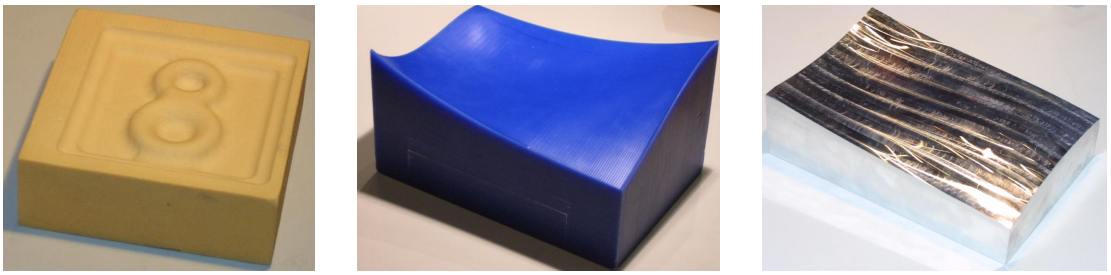
5.1 The Experiments

The experiments consisted of machining a surface on the real machine, laser scanning the results, and comparing the simulated results to the scanned results.

5.1.1 The Physical Surfaces

I used three different surfaces in the experiment, which are all shown in Figure 5.3. The first surface (Figure 5.3(a)) was created using a three axis tool path on a machine built in the mechanical engineering lab. It was machined on a foam stock approximately 13cm x 13cm x 5cm. This surface will have higher error since it has sharp edges and because the machine that created the surface was not built with the same engineering tolerances as industry made machines are.

The other two surfaces were machined on the mechanical engineering lab's Decl Maho five axis machine. The surface in Figure 5.3(b) was created on a wax stock using a three half half tool path (which is a five axis tool path with the rotational axis fixed at different positions for different regions of the surface). The last surface (Figure 5.3(c)) was machined on a aluminium stock using a five axis tool path. Both the blue wax stock and the aluminium stock were approximately 15cm x 22.5cm.

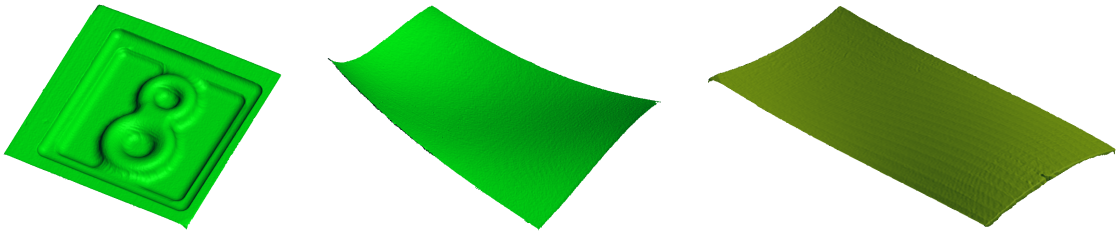


(a) Three axis (b) Three half half (c) Five axis

Figure 5.3: Surfaces used in physical comparison tests.

5.1.2 Scanning

To scan in the real surfaces we used a Minolta VIVID 900 3D digitizer. This uses a colour charge-coupled device (CCD), like the ones in digital cameras, and laser triangulation to create a series of points representing the object [Wik06a]. These points are later triangulated into a surface. The accuracy of the camera is $x : \pm 0.38mm$, $y : \pm 0.31mm$, $z : \pm 0.35mm$ [Inc06]. This resolution is about ten times the machining tolerance for metal milling and results in scanned surfaces that are somewhat bumpy, as can be seen in Figure 5.4 which shows the results. The scanner produces a point cloud that is then used to create a triangulated surface. This is further discussed in Section 5.1.4.

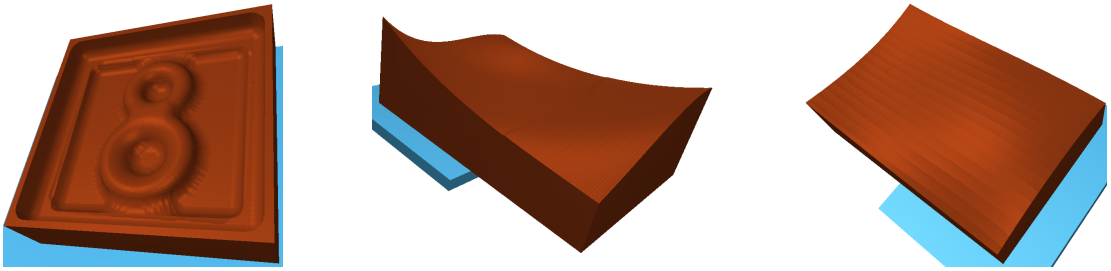


(a) Three axis (b) Three half half (c) Five axis

Figure 5.4: 3D digitized scan of the machined surfaces.

5.1.3 The Virtual Surfaces

The virtual surfaces, seen in Figure 5.5, were machined using the same tool paths as the real surfaces. Table 5.1 contains the parameters used to create each of the virtual surfaces. One thing to note is that I used a stock density of 3 even though the machining tolerance for the three half half and five axis surface was 0.0254 mm. Using the guidelines from the previous chapter this would mean we



(a) Three axis (b) Three half half (c) Five axis

Figure 5.5: ToolSim milled surfaces used in physical comparison tests.

should use a stock density of 16 since these surfaces are continuous. However as noted in Section 5.1.2 the scanning resolution of the camera is less then 3 samples per mm. Therefore I used a similar density for the virtual stock.

Table 5.1: Parameters used to create virtual surfaces

	three axis	**three half half**	**five axis**
width	120 mm	225 mm	225 mm
length	120 mm	150 mm	150 mm
tool	sphere	torus	
tool radius	6.35 mm	major = 6.7 mm, minor = 6 mm	
stock density	3		

5.1.4 Preparing Surfaces for Comparison

To compare the results of scanning with the results of the virtual machining I used the software Geomagic Qualify. However before this could happen, the original point cloud that the scanner produced needed to be converted into a surface, and the

virtual surface and scanned surface needed to be aligned. The program Geomagic Studio was used to perform these operations.

There were some issues we noted with the scanned surfaces. One problem was that sometimes the scanner picked up the edge of the surface even though we only wanted the top. Another issue that was a particular problem was reflection of the laser on the surface creating extra noise in the scanned data. This was a particular problem in the five axis surface since its stock was made of aluminium. To compensate for this the aluminium stock was scanned in two passes, covering different areas of the stock. Even with this adjustment there was a small area with significant noise. A final issue was with the machining of the surfaces itself. The stocks were not the same size as the ideal surfaces, and in the case of the blue stock possibly not even rectangular. Also in the case of the three axis surface the origin of the tool path relative to the stock was arbitrary, leading to the unmachined areas of the stock to have different depths.

To adjust for these sources of error, I edited the scanned surfaces to remove the unwanted parts. In the case of the five axis surface I also smoothed over the part of the stock that had a high error due to reflection (Figure 5.6(a)).

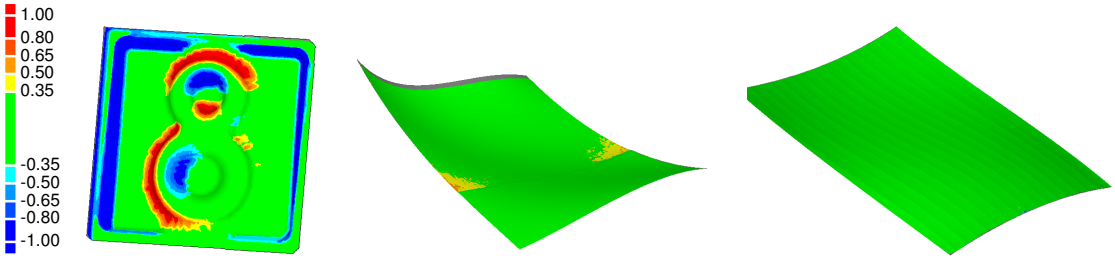
(a) before clean up (b) after clean up

Figure 5.6: Scanning noise clean up on the five axis surface.

After the scanned surfaces were cleaned up they were aligned with the virtual surface and any unneeded areas, such as the unmachined area of the three axis surface, were removed. Then they were ready to be compared to the virtual surfaces.

5.2 Results and Analysis

I used Geomagic Qualify to compute the difference between the virtual and scanned surfaces. Figure 5.7 shows the virtual surfaces coloured by the difference between each point on that surface and of the closest point on the scanned surface. Table 5.2 contains a summary of the results of comparing the three virtual surfaces with their physical counterparts. We are mostly interested in the maximum error, which we would like to be between $\pm 0.35mm$, since that is the resolution of the 3D scanner. This corresponds to the green areas in Figure 5.7.



(a) Three axis (b) Three half half (c) Five axis

Figure 5.7: Difference between the physical surfaces and the virtual surfaces

As expected the three axis surface has somewhat high error. If we look at Figure 5.7 we see that the source of errors is along the edges in the surface. This was due to errors in the machine. However the flat parts of the surface and even most of the three axis surface are the same within scanning resolution. The three

half half surface has the lowest number maximum error. This is partly due to the fact that it was milled on a machine with industrial machining tolerances and that the stock material was non-reflective. The five axis surface has a slightly higher maximum error then the three half half surface. This is probably due to the scanned surface having more noise due to laser reflection on the stock.

Table 5.2: Error statistics for the difference between surfaces

	Max Error	**Avg Error**	**Std. Dev.**
three axis	2.00mm	0.38mm	0.42mm
three half half	0.62mm	0.09mm	0.08mm
five axis	0.84mm	0.05mm	0.05mm

5.2.1 Sources of Error

There are a number of sources of error for the difference between the virtual and physical surfaces. As noted before there are issues with the scanning process. Another problem is alignment of the stock with the machine in the real machine and in the simulation. Both of these can be mitigated, as noted before.

Another source of error is differences between the machine model and the real machine. However, only a few aspects of the machine construction affect the results of machining. These are the tool shape, the distance between the tool tip, and rotational axis and the distance between the rotational axis if any. Since there are only a few it is easy to adjust for them.

A final source of error is the simplifications built into the simulation. The simulation does not take into account the forces of the stock against the tool, and therefore would miss any effects such as skipping, due to the tool moving too

fast. However these type of effects are undesirable in the machining process. The engineers creating the tool path would want to know about the presence of these unwanted effects, and these errors could alert the engineers that the effects exist.

5.3 ToolSim's Memory and CPU usage

We are looking at connecting ToolSim to a real machine, therefore we should consider the memory usaage and speed of the simulation when it is used to machine a tool path. I will use the five axis surface as an example. Assuming we had used the Chapter 4 recommendations for stock densities and used a stock density of 16 then the stock for the five axis surface would have 2401 by 3601 sample points and use 263.855 Mb of memory. On a computer with 1 Gb of memory and a 1.7 GHz Pentium M processor the five axis surface on this stock was machined in 2 minutes and 22.7 seconds. In general I found that as long as the stock size did not exceed more than half of system memory then the stock size did not have a significant impact on machining speed.

5.4 Summary

These experiments have shown that the results of the simulation generally match the results of the real machine, within the tolerance of the digital laser scanner. The three axis surface had many points outside of the range of the range of scanner tolerance, but this is due to a number of known problems with the machine. The five axis surface was generally the same as the virtual surface and has very few points outside of the scanner range of tolerance. These few regions of greater difference are probably due the surface material reflections interacting with the

scanning equipment. The three axis surface was largely the same as the virtual surface, except for two areas. The error in these areas are likely due to some of the simplifications in ToolSim, such as ignoring the forces of the stock on the tool, and merits further investigation.

Chapter 6

Conclusion

This thesis has provided a guide on constructing CNC machining simulators and how to show confidence in their correctness. Here I will summarize the important ideas and results and their limitations. I will also discuss future work that may help remove some of the limitations.

6.1 Important Highlights

The main highlights of the design of ToolSim is it use of height fields, swept surfaces, hierarchal models, and accurate direction of motion algorithm. The height fields allow the stock to be represented with a reasonable amount of memory, while keeping intersection and rendering simple. Swept surfaces allow much higher accuracy when computing the results of machining without sacrificing speed. Using the hierarchal model of the machine we can construct a function that gives us the direction of motion of any point on the tool. This simplifies the code and gives us increased flexibility in computing results and testing the machine. The hierarchal models of the machines allow the representation of a variety of CNC machines and

support the computation of results.

Some of the other important contributions of the thesis came from the numerical tests performed on ToolSim. I verified that ToolSim is correct by showing that the error in the simulation has $O(h^2)$ convergence. I also determined the stock densities required to keep simulation error below the error tolerance for metal and wood milling. The process for deriving these requirements could also be used for other machining tolerance levels. I also noted that the accurate direction of motion algorithm increased accuracy by 10% to 50%. The physical tests showed that the results were by and far the same between the virtual and actual machining.

6.2 Limitations

There are a number of limitations on the results presented in this thesis. As the chapter on design noted, ToolSim can only machine functional surfaces. Also the results of Chapter 4 only apply to the vertical height field stock, but not the radial height field stock. The physical tests were limited by the resolution of the scanner which was, at least for two of the surfaces, ten times lower then the machining tolerance used to create the surface.

6.3 Future Work

The work presented in this thesis can be expanded on in many ways. Some simple issues to deal with would be to address the limitations noted in the previous section. While there are good reasons for restricting ToolSim to functional surfaces, it would be a good idea to do the numerical verification on the radial height field. This would also provide the error analysis for this type of stock.

Currently stocks are limited by the amount of physical memory. Until 64-bit machines become more popular it may be useful to modify the stock to use data structures that store data off-line, such as b-trees. Related to this would be changing stock rendering to show a lower density version of the stock. Even if the stock fits in memory this is still useful since it can be used to make sure that ToolSim remains interactive.

The current simulation has many approximations and simplifications. One important factor that it ignores is the forces involved in machining. Modeling forces would allow the simulation to generate more accurate results and let it detect possible damage to the machine.

As was noted in the introduction that one of the goals of the thesis was to allow the data to be analyzed more easily. The current error analysis in ToolSim is somewhat primitive, and could be expanded to include gouge detection. The current scallop detection algorithm is not entirely automated and needs some user intervention. With further investigation this could be expanded to be fully automatic. Another issue with the scallop detection algorithm is that it uses the first derivative of the surface to detect the approximate location of the scallops, which is used to compute the height. If the second derivate was used instead then the algorithm would be able to make a better estimation of the scallop height.

A final longer term goal would be to get the simulator to interact with the CNC machine. By predicting what will happen the simulator will be able to act as an intermediary between the machine and machinist, and allow for greater productivity by reducing the possible errors and decreasing the designed part to machine part time.

Bibliography

[BLW92] D. Blackmore, M.C. Leu, and L.P. Wang. Applications of flows and envelopes to NC machining. *Annals of the CIRP*, 41(1):493–496, 1992.

[Gla89] Andrew S. Glassner, editor. *An Introduction to Ray Tracing*. Academic Press LTD., San Diego, CA 92101, 1989.

[GPI+05] P. Gray, G. Poon, G. Israeli, S. Bedi, S. Mann, and D. Miller. Model to Part: A roadmap for the CNC machine of the future. In *AMST 2005 7th International Conference on Advanced Manufacturing Systems and Technology*, number 486, pages 247–256. Springer Wien New York, 2005.

[Gra02] P. Gray. Graphics-Assisted Tool Path Verification in 5-Axis Surface Machining. MASc. Thesis, University of Waterloo, Canada, 2002.

[HB94] Donald Hearn and M. Pauline Baker. *Computer Graphics*. Prentice Hall, Englewood Cliffs, New Jersey 07632, second edition, 1994.

[Hoo86] Tim Van Hook. Real-time shaded NC milling display. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 15–20, New York, NY, USA, 1986. ACM Press.

[Inc06] Canadian Measurement-Metrology Inc. Minolta Vivid 3d Digitizer, 2006. [Online; accessed 12-June-2006; `http://www.cmmxyz.com/236CC/238.html`].

[KBM+04] Craig Kaplan, Sanjeev Bedi, Stephen Mann, Gilad Israeli, and Gilbert Poon. A New Paradigm for Woodworking with NC Machines. *Computer-Aided Design*, 1:217–222, 2004.

[LMB05] Chenggang Li, Stephen Mann, and Sanjeev Bedi. Error Measurement for Flank Milling. *Computer-Aided Design*, 37(14), December 2005.

[MB02] S. Mann and S. Bedi. Generalization of the imprint method to general surfaces of revolution for nc machining. *Computer Aided Design*, 34(5):373–378, April 2002.

[MBIZ] S. Mann, S. Bedi, G. Israeli, and X. Zhou. Tool motions for 5-axis machining. Unpublished.

[Pre75] P.M. Prenter. *Splines and Variational Methods*. John Wiley & Sons, June 1975.

[RBIM01] D. Roth, S. Bedi, F. Ismail, and S. Mann. Surfaces swept by a toroidal cutter during 5-axis machining. *Computer Aided Design*, 33:57–63, January 2001.

[Wik06a] Wikipedia. 3d scanner — Wikipedia, the free encyclopedia, 2006. [Online; accessed 26-August-2006; `http://en.wikipedia.org/w/index.php?title=3D_scanner&oldid=71748138`].

[Wik06b] Wikipedia. CNC — Wikipedia, the free encyclopedia, 2006. [Online; accessed 12-June-2006; `http://en.wikipedia.org/w/index.php?title=CNC&oldid=57863303`].

[Wik06c] Wikipedia. G-code — Wikipedia, the free encyclopedia, 2006. [Online; accessed 18-July-2006; `http://en.wikipedia.org/w/index.php?title=G-code&oldid=62732365`].

[WLB97] Liping Wang, Ming C. Leu, and Denis Blackmore. Generating swept solids for NC verification using the SEDE method. In *SMA '97: Proceedings of the fourth ACM symposium on Solid modeling and applications*,

pages 364–375, New York, NY, USA, 1997. ACM Press.

[WNDS00] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL Programing Guid*. Addison-Wesley, Englewood Cliffs, New jersey 07632, third edition, February 2000.

Glossary

NC Numerically controlled.

CNC Computer Numerically controlled.

tool The part of the machine that cuts away pieces of the stock during machining.

design surface The shape of the part we want to machine.

stock The material the CNC machine is creating a part from.

axes Translation or rotation axes that the machine can move about.

machine coordinate A position of a machine axis.

tool path A set of tuples specifying machine coordinates that the machine move the tool along to create the design surface from the stock.

surface of revolution The idealized shape of the tool, used in computation.

g-codes Tool path commands understood by most commercial CNC machines.

stock density The number of samples per unit area on the stock.

in-between steps The extra steps the simulated machine between tool positions.

in-between frames Synonym for in-between steps.

stamping Intersection of the tool with the stock at a particular position.

swept surface The surface of the volume of space that the tool moves through.

grazing curves The area of contact the tool makes with the swept surface.

scallop Extra material left behind on the stock after machining, but does not exist on the design surface.

Simulation error The difference between the simulation results and actual results, due to the approximations in the simulation.

Machine tolerance Machine tolerance is the maximum amount of error a machine will introduce when machining.

GPU Graphics Processing Unit.

Appendix A

Affine Transformations

These are the scale, transpose, and rotation transformation matrices, which are used to create the machine geometry. Normally the functions $f(t), g(t), h(t)$ are constants. However in ToolSim the linear interpolation of the machine coordinates are also used as parameters in the machine transformations.

$$T_x(f(t)) = T(f(t), 0, 0) \tag{A.1}$$

$$T_y(f(t)) = T(0, f(t), 0) \tag{A.2}$$

$$T_z(f(t)) = T(0, 0, f(t)) \tag{A.3}$$

$$T(f(t), g(t), h(t)) = \begin{bmatrix} 1 & 0 & 0 & f(t) \\ 0 & 1 & 0 & g(t) \\ 0 & 0 & 1 & h(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

$$S(f(t), g(t), h(t)) = \begin{bmatrix} f(t) & 0 & 0 & 0 \\ 0 & g(t) & 0 & 0 \\ 0 & 0 & h(t) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

$$R_x(f(t)) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(f(t)) & -\sin(f(t)) & 0 \\ 0 & \sin(f(t)) & \cos(f(t)) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

$$R_y(f(t)) = \begin{bmatrix} \cos(f(t)) & 0 & \sin(f(t)) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(f(t)) & 0 & \cos(f(t)) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.7}$$

$$R_z(f(t)) = \begin{bmatrix} \cos(f(t)) & -\sin(f(t)) & 0 & 0 \\ \sin(f(t)) & \cos(f(t)) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.8}$$

The derivatives of the transformation matrices are used to compute the accurate direction of motion of the tool relative to the stock (Section 3.2.4).

$$D_t^1 T_x(f(t)) = D_t^1 T(f(t), 0, 0) \tag{A.9}$$

$$D_t^1 T_y(f(t)) = D_t^1 T(0, f(t), 0) \tag{A.10}$$

$$D_t^1 T_z(f(t)) = D_t^1 T(0, 0, f(t)) \tag{A.11}$$

$$D_t^1 T(f(t), g(t), h(t)) = \begin{bmatrix} 0 & 0 & 0 & D_t^1 f(t) \\ 0 & 0 & 0 & D_t^1 g(t) \\ 0 & 0 & 0 & D_t^1 h(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.12}$$

$$D_t^1 R_x(f(t)) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\sin(f(t)) \times D_t^1 f(t) & -\cos(f(t)) \times D_t^1 f(t) & 0 \\ 0 & \cos(f(t)) \times D_t^1 f(t) & -\sin(f(t)) \times D_t^1 f(t) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.13}$$

$$D_t^1 R_y(f(t)) = \begin{bmatrix} -\sin(f(t)) \times D_t^1 f(t) & 0 & \cos(f(t)) \times D_t^1 f(t) & 0 \\ 0 & 0 & 0 & 0 \\ -\cos(f(t)) \times D_t^1 f(t) & 0 & -\sin(f(t)) \times D_t^1 f(t) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.14}$$

$$D_t^1 R_z(f(t)) = \begin{bmatrix} -\sin(f(t)) \times D_t^1 f(t) & -\cos(f(t)) \times D_t^1 f(t) & 0 & 0 \\ \cos(f(t)) \times D_t^1 f(t) & -\sin(f(t)) \times D_t^1 f(t) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.15}$$

Appendix B

ToolSim File Specifications

The machine and tool path files contain one command per line. The '#' character begins a comment which is text which ends at the end of the line. Commands may have a certain number of arguments. In the command specification the arguments are represented by `<type>`, where `type` is the type of the argument listed in Table B.1. Optional arguments appear between '[' ']'.

`<num>`	A floating point or integer number
`<str>`	A string without spaces, or a string with spaces beginning and terminating with double or single quotes
`<char>`	A single character
`<unit>`	A unit of length such as cm, mm, m, inches or feet
`<val>`	A numeric expression in the form of one of `<num>`, [-]`<char>`, `<num>`*`<char>`
`<shape>`	See table of shape commands

Table B.1: file command argument types

none	Placeholder for no shape
cube [<num$_1$> <num$_2$> <num$_3$>]	A cube, from $(0,0,0)$ to (num$_1$, num$_2$, num$_3$). Defaults to $(0,0,0)$ to $(1,1,1)$
cylinder [<num$_1$> <num$_2$>]	A cylinder with height num_1 and radius num_2. Defaults to height 1, radius 1.
sphere [<num>]	A sphere with radius num. Defaults to radius 1.
cone	A cone with radius 0.5 and height 0.5
conic <num$_1$> <num$_2$> <num$_3$>	A conic section with height num_1, top radius num_2 and bottom radius num_3
torus [<num$_1$> <num$_2$>]	A torus with major radius num_1 and minor radius num_2. Defaults to 0.5 and 0.2

Table B.2: shape commands — used to create a shape

B.1 Machine File

Table B.3: machine specification file commands

Material <str> [(<num> <num> <num>) (<num> <num> <num>) <num>]	If all the parameters are present then it creates an OpenGL material with diffuse and specular highlightes from the 'num' parmaters. The material is named 'str'. If only the string is present then it sets the material prevoisly create with that name as the current OpenGL material.
push [<str>]	Create a new hierarchal node, optionally with the name 'str'.
pop	End a hierarchal node

Table B.3: machine specification file commands (cont.)

clone <str>	Create a new hierarchal node that is a copy of another node named 'str'. It will be the current node and will need to be ended with a pop command.
baseunit [<num>] <unit>	Reads in a unit and optional number and converts it to `mm`. Then it multiples the parameters of `Translate, Scale, Rotate, stock`, and the shape commands, by this factor.
enableAxis <char> <num_1> [<num_2> <num_3>]	Enable the UI for the axis named 'char' and set it's initial value to be num_1. Optionally set the UI's maximum and minimum range to be num_2 and num_3.
Setting <str>	A string changing the program setting, as it appears in the program's setting file.
Name <str>	The name of the machine. It will appeare on TooISim's title bar.
Translate <val> <val> <val>	Append a translation to the current node.
Scale <val> <val> <val>	Append a scale to the current node.
Rotate <val> <val> <val>	Append a rotation to the current node.
Texture [<str>]	If `str` is supplied then subsequent shapes will be rendered with the texture specified by the file '`str`'. Otherwise disables rendering with textures for subsequent shapes.
toolCyl <shape>	The tool cylinder shaft; `shape` should be a `cylinder` command.
tool <shape>	The tool. Can be any command in Table B.2.

Table B.3: machine specification file commands (cont.)

stock [**char**] <num$_1$> <num$_2$> <num$_3$> <num$_4$>	If `char` is 'v' or absent then it creates a vertical height field stock with num$_1$ width, num$_2$ depth, num$_3$ length and num$_4$ density. If `char` is 'r' then it create a radial height field with radius num$_3$ and height num$_2$. The vertical density is num$_4$ while the radial density will be num$_4$*num$_1$.

B.2 Machine File Examples

There where a number of machines created for ToolSim. The hierarchal models of the ones referenced in this thesis are presented here. The single axis lathe is shown in Section B.2.1. The five axis table spindle machine, shown in Section B.2.2, is the machine seen in all the screen shots of ToolSim and was the machine used in the experiments of Chapter 4 and 5.

B.2.1 The Lathe

```
Name Lathe

baseunit cm

enableAxis x 5 0 15
enableAxis y 4 0 50
enableAxis C 0 norange

Material blue ( 0.3 0.7 0.9 ) (0.7 0.7 0.7) 100
Material copper ( 0.550800 0.211800 0.066000 ) ( 0.580594 0.223257 0.069570 ) 51.200001
Material grey ( 0.7 0.7 0.7 ) ( 0.5 0.5 0.5 ) 90

# no need to intersect the cylinder
Setting Intersect Cylinder;0

Material copper

push
        Rotate C 0 0 1
        # the base
        # blue thing
```

```
        push
                Translate 0 0 -1 # move table below stock
                Scale 1.2 1.2 1
                Scale d d 1
                Material blue
                Texture marble.png
                cylinder 1 0.1
                Texture
        pop

        Material copper
        stock r 4 4 4 500
pop

# drill unit
push
        Translate x 0 0
        Rotate 90 0 1 0

        Translate -y 0 0
        push
                Translate 0 0 4.5
                push
                        Scale 0.75 0.75 2
                        cylinder
                pop
                push
                        Translate -1 -1 -2
                        Scale 2 2 2
                        cube
                pop
        pop

        Material grey

        Translate 0.0  0.0  0.0  # so we can change tool cylinder lengths
        toolCyl cylinder 3.0 0.5
        tool sphere 0.5
pop
```

B.2.2 The Table Spindle

```
Name "Table Spindle - cm"

Material copper ( 0.550800 0.211800 0.066000 ) ( 0.580594 0.223257 0.069570 ) 51.200001
Material blue ( 0.3 0.7 0.9 ) ( 0.7 0.7 0.7 ) 100
Material grey ( 0.7 0.7 0.7 ) ( 0.5 0.5 0.5 ) 90

# default
baseunit cm

# set values and ranges for axis
enableAxis x 0 -3 15
enableAxis y 0 -3 15
enableAxis z 0 -2 10
enableAxis A 0 -45 45
enableAxis C 0 norange

# center machine
Translate -10 -7.5 0.0

# draw table
```

```
Material blue
push
        Translate -x -y 0.0
        push
                Scale 20.0 15.0 1.0
                cube
        pop

        push
                Translate 3.0  3.0  1.0
                # polished copper
                Material copper
                stock z 9 6 1 500
        pop
pop
Material copper

# Drill unit
Translate 0.  0.0  z
Translate 0   0    d
Translate 0.  0.  14.

push
        Scale 6.  6.  6.
        cube
pop

Translate 3.  3.  -1.
push
        Scale 2.5  2.5  1.
        cylinder
pop

Rotate C  0.0  0.0  1.0

push
        Translate -3.  -3.  -3.
        push
                Scale 6.  6.  3.
                cube
        pop
        Translate 0.  0.  -4.
        push
                Scale 1.  6.  4.
                cube
        pop
        Translate 5.  0.  0.
        Scale 1.  6.  4.
        cube
pop

Translate 0.  0.  -7.
Rotate A  1.0  0.0  0.0

# tool cylinder
Material grey
push
        Translate 0.0  0.0  -5.0
        toolCyl cylinder 3.0 0.5

        #tool tip
        tool sphere 0.5
pop
#

push
```

```
        Translate 0.0  0.0  -2.0
        cylinder
pop

push
        Translate 0.0  0.0  -1.0
        Scale 2.0  2.0  4.0
        cylinder
pop
```

B.3 Tool Path File

Table B.4: tool path specification file – ToolSim commands

scale <num$_1$> <num$_2$> <num$_3$>	scale all tool path data in the x direction by num$_1$, y direction by num$_2$, and z direction by num$_3$
origin <num$_1$> <num$_2$> <num$_3$>	translate all tool path data by (num$_1$, num$_2$, num$_3$)
tool <shape>	same as the machine file command
stock [**char**] <num$_1$> <num$_2$> <num$_3$> <num$_4$>	same as the machine file command
depth <num>	change only the depth of the stock
nbf <num>	The suggested number of in between steps to use. For steps with rotation this values is actually multiplied with the number of degrees rotates divided by 20.
dC <num>	The amount to move the C-axis for each machine coordinate in the file. For use with one axis lathes.
dy <num>	The amount to move the y-axis for each machine coordinate in the file. For use with one axis lathes.
pitch <num>	For a full rotation of the C-axis, how much to move the y-axis. For use with one axis lathes.

B.3.1 Tool Path Data Formats

There an a number of different formats for the tool path data. A file will tipical contain many of these entries. A file can only contain either M128 machining data or regular machining data, though the behavior resulting from mixing different type of formats is undefined.

[g01] <num> [<num> [<num> [<num> <num>]]] One or two coordinates for single axis tool path, there coordinates for a three axis tool path, or five coordinates for a five axis tool path. The three or five axis tool path can also optionally start with "g01", which is the g-code for linear interpolation.

[g01] **X**<num> **Y**<num> **Z**<num> Three axes tool path.

[g01] **X**<num> **Y**<num> **Z**<num> **A**<num> **C**<num> Five Axes tool path. The xyz-position specified is the contact point.

LN X<num> **Y**<num> **Z**<num> **NX**<num> **NY**<num> **NZ**<num> **TX**<num> **TY**<num> **TZ**<num> M128 machining. The x, y, z-tuple is the contact point, the nx, ny, nz-tuple is the surface normal at the contact point after machining, and the tx, ty, tz-tuple is the tool axis orientation.

Appendix C

User Interface

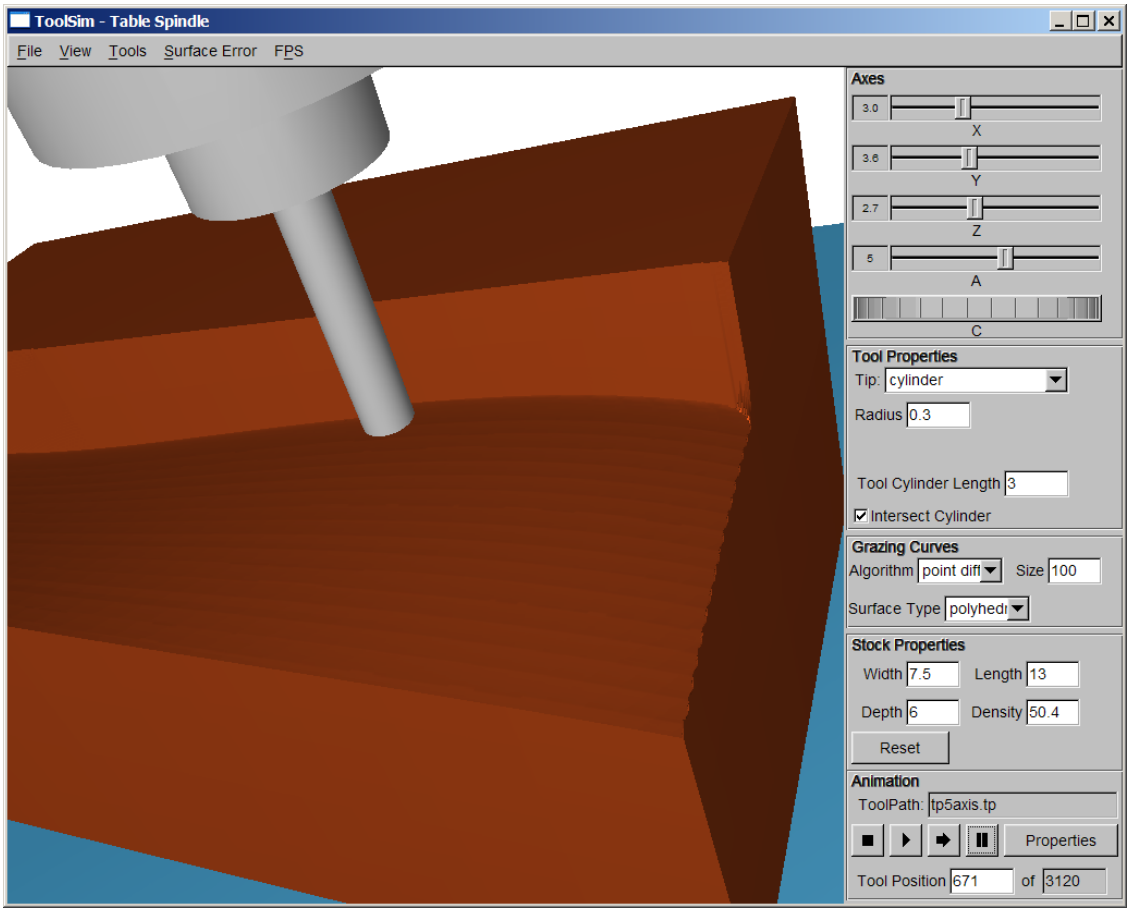
This appendix describes how to use ToolSim. It details all the on screen user interface including the menus and dialog boxes. It also describes the command line parameters that can be passed to ToolSim at startup.

C.1 On Screen Interface

Figure C.1 shows the main user interface of ToolSim. At the top right are sliders and rollers that control the position of a machine axis. Bellow that is the tool properties area where the user can select the tool shape and set its properties, such as radius and height. They can also set the cylinder shaft length, and whether ToolSim intersects the cylinder shaft with the stock.

Next is the Grazing curves area. It lets the user select the algorithm used to generate grazing curves, including none, which disables them. It also contains the selection for the swept surface type and the number of points on the grazing curve.

Figure C.1: ToolSim user interface



After that is the stock properties area, which lets the user enter the stock width, length, depth and density. Since changing stock properties may take a few seconds and pause the program, ToolSim requires the user to press the stock 'reset' button to make the changes effective. The reset of the settings in the ToolSim interface take effect immediately.

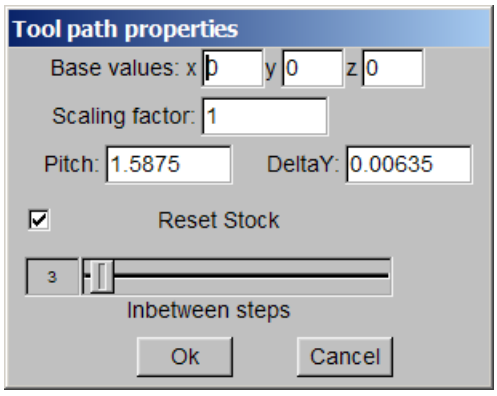
Finally the tool path area displays which tool path is currently loaded and what is the current tool position the machine is on out of the total tool positions. It also allows the user to start, stop, pause and reset the machine's running of the tool

path. The properties button will bring up and extra dialog box which lets the user modify tool path properties such as, origin, scale and number of in between steps.

Tool Path Properties Dialog Box

Figure C.2 shows the tool path properties dialog box. The base values fields offset the x, y, z coordinates of the tool path by the amount specified, while the program will multiply the coordinates in the tool path by the scaling factor. If the tool path being loaded is for single axis lathes then the pitch and delta-Y fields will appear. They will contain the values of the `pitch` and/or `dy` commands in the tool path file but these fields allow the user to modify them before the tool path is loaded. The reset stock option controls whether the stock is reset if the stock command exists in the tool path file and the in-between steps slider controls the number of in-between steps that will be used when milling the tool path.

Figure C.2: Tool Path Properties Dialog Box



C.2 Menus

ToolSim's top level menu has five items: the file, view, tools, surface error, and fps menus.

file/open machine file Load a machine file.

file/recent machine files/ A sub-menu that contains a list of recently opened machine files.

file/open tool path file Load a tool path file.

file/recent tool path files/ A sub-menu that contains a list of recently opened tool path files.

file/open stock Load a stock file.

file/save stock Save the current stock to a file.

file/save stock as obj file Save the current stock to a file in 'obj' format.

file/compare stock Allows the user load another stock to compare with the current stock in the program. This command will destroy the current stock and put in its place the difference between the height fields of the two stocks. Obviously, the stocks need to be the same size.

file/save screen Save a screen shot of the machine and stock to a file.

file/save animation Save a screen shot of the machine and stock to a file every time they change. This sequence of files can then be composed into a movie by an external program.

file/save grazing curves Save grazing curve data that is generated by the program to a file.

view/reset view Resets the view of the machine to the default starting view.

view/show shadows Enable shadows for all object but the stock.

view/show particles Enable particles that show when the stock is cut.

view/show stock shadows Enable shadows for the stock.

view/show stock Enable rendering of the stock.

view/ghost machine Render the machine semi-transparently to enable better view of the stock.

view/draw stock nicer Render the stock with a better algorithm that shows edges better.

view/show curvature Colour the stock using the local curvature of the stock.

view/lock camera to stock Keep the stock centred in the view.

view/show grazing curve Enable drawing of the current grazing curve and swept surface.

view/show grazing curve/show direction vector Show the direction of motion of the points on the grazing curve.

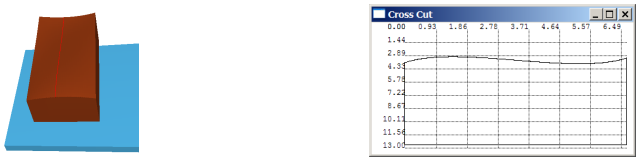
view/show grazing curve/show all grazing curves Show all the swept surface.

view/show grazing curve/clear grazing curves Clear all currently showing swept surfaces.

view/show error surface Show the currently loaded error surface.

view/draw stock as grid Draw the stock as a wire frame, showing the individual height fields.

The tools menu contains the view crosscut option. It allows the user to select a crosscut aligned with the x or y axis by moving the mouse in the x or y direction. Once the user clicks a mouse button a new dialog box will pop up showing the crosscut that was selected.



(a) Selecting a crosscut

(b) viewing the crosscut

Figure C.3: Crosscut tool

The surface error menu contains items for computing the surface error using various methods listed in Section 3.5. Save the difference computed by the error comparisons methods to a file. The FPS menu controls the speed that machining is rendered at. Natural speed is the speed where the program re-renders the machine every time it cuts the stock. The other options set a fixed number of renders per second.

C.3 Command Line Parameters

Many of the commands available through ToolSim's menu can also be run through the command line. This enables the automation of ToolSim. Parameter arguments in square brackets are optional, while arguments in angle brackets are required.

-stereo [eye separation] [focal length] Enable stereoscopic viewing. [eye sepa-

ration] defaults to 5. [focal length] defaults to [eye separation] * 30.

-fps `<fps>` Sets the frame rate throttle to the floating point number `<fps>`.

-mac `<file>` Loads the machine from `<file>`.

-tp `<file>` Loads the tool path from `<file>`.

-load `<file>` Loads a stock and/or error surface from `<file>`.

-autosave `<file>` Automatically start running the current tool path and save the resulting stock to `<file>`.

-tool `<shape>` Set the current tool to `<shape>`. The format of `<shape>` is specified in Table B.2.

-noshadow Disable rendering with shadows.

-imprint `<grazing type>` Set the grazing curve generation algorithm to "none", "diff" for point difference, or "dt" for derivative.

-imprintsurface `<grazing surface type>` Set the swept surface type. Either "poly" for polyhedron or "spline" for B-spline.

-stockcomp $\texttt{<file}_1\texttt{>}$ $\texttt{<file}_2\texttt{>}$ Load two stocks, compute their difference and display the result.

-setting `<string>` Apply any setting as it appears in ToolSim's settings file.