Real-time detection of geometric interference: 

application to full-body 5-axis haptics

Stephen Ho

B. S., Mechanical Engineering (1996)

Cornell University

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1999

© Massachusetts Institute of Technology, 1999. All Rights Reserved.

Author .........................................................................................

Department of Mechanical Engineering

May 26, 1999

Certified by .................................................................

Sanjay E. Sarma
Assistant Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .................................................................

Prof. Ain A. Sonin
Chairman, Department Committee on Graduate Students
Interference Detection for Haptic Application in 5-Axis Tool Path Generation
by
Stephen Ho

Submitted to the Department of Mechanical Engineering on May 26, 1999, in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

Abstract

Haptic systems for virtual environment simulations have stringent computation requirements to meet the high bandwidth necessary for operating a haptic device. The underlying computation task of a haptic rendering system is the collision detection and interference analysis of bodies involved in the haptic environment. Faster computer processors and more sophisticated methodologies have enabled haptic systems to model fully three-dimensional objects and their interactions in 6DOF space. A methodology for fast computation of solid interactions as well as its incorporation into a haptic system are discussed. The application of a fully three-dimensional haptic environment to Computer Aided Design and Manufacturing is presented.

Thesis Supervisor: Prof. Sanjay E. Sarma
Acknowledgments

This project benefited greatly from the support and contributions of Suzuki Motor Corporation and Adachi Yoshitaka. I would like to thank Adachi and Suzuki for developing a haptic device for our application and for Adachi’s tireless efforts to interface my program to the Suzuki haptic device. Appreciation is also extended to Masatake Saitoh, Takahiro Kumano, Ikemoto Akio, and the rest of Adachi’s colleagues at Suzuki Motor Corporation.

I give great thanks to Professor Sanjay Sarma whose encouragement and thoughtfulness have made working on this project joyful and enlightening.

To my colleagues Elmer Lee, Laxmiprasad Putta, Mahadevan Balasubramaniam, Taejung Kim, Seung-kil Son, Niranjan Kundapur, Paula Valdivia Y Alvarado, and Yogesh Joshi, your company has filled the past two years with friendship and merriment. I could not have enjoyed MIT more without you.

To my family, whose understanding and strength have given me the freedom to open my heart, your love and support was essential to helping me though the rough times. Jenny, I am proud to have you as my sister. I thank you all.

And finally to Daniel, who has made my life complete, I am truly fortunate to have you in my life. Your devotion and support are indispensable.
# Table of Contents

Chapter 1: Introduction and Background ........................................... 7
  1.1 Haptic Systems ................................................................. 7
  1.2 Collision Detection Methodologies ........................................ 9
    1.2.1 Lin-Canny Algorithm, I-Collide and V-clip .......................... 9
    1.2.2 Triangle Set Methods with Bounding Volume Hierarchies ........ 10
  1.3 Full body collision detection for haptic systems ....................... 11
    1.3.1 Our System ............................................................. 12
    1.3.2 Application ............................................................ 13
  1.4 Thesis Overview ............................................................ 14

Chapter 2: Collision Detection .................................................... 16
  2.1 Solid Modeling Hierarchy .................................................. 16
  2.2 Implicit Equation model for Solid Representation ....................... 18
    2.2.1 Multiple Implicit Equations for Complex Solids .................. 19
    2.2.2 Constructive Solid Geometry ...................................... 21
    2.2.3 Implicit Equation Binary Space Partition Tree ................... 21
  2.3 Point Cloud model for Solid Representation ............................ 23
    2.3.1 Point Cloud Generation ............................................. 25
    2.3.2 Combination Representation ........................................ 26
  2.4 Collision Detection Method ............................................... 26
  2.5 Discussion ................................................................. 27

Chapter 3: Bounding Volume Hierarchy ......................................... 29
  3.1 Bounding Volume Properties .............................................. 29
  3.2 Bounding Volume Examples ................................................. 31
    3.2.1 Axis Aligned Bounding box ...................................... 32
    3.2.2 Oriented Bounding Box ............................................ 32
    3.2.3 k-Discrete Orientation Polytope ................................ 33
  3.3 Hierarchy .......................................................................... 34
    3.3.1 Point Cloud Bounding Volume Hierarchy ......................... 35
    3.3.2 Implicit Equation BSP-Tree Bounding Volume Hierarchy .......... 37
  3.4 Collision Detection Using Bounding Volume Hierarchical Trees .......... 39

Chapter 4: Interference Analysis .................................................. 42
  4.1 Depth Information ........................................................... 42
List of Figures

Figure 2.1: Solid Representation Hierarchy ............................................. 17
Figure 2.2: Constructive Solid Geometry .................................................... 20
Figure 2.3: Binary Space Partition .............................................................. 22
Figure 2.4: Point Cloud Representation ........................................................ 25
Figure 2.5: Grazing Contact Situation .......................................................... 28
Figure 3.1: Non-overlapping bounding volumes ........................................... 30
Figure 3.2: Axis Aligned Bounding Boxes .................................................... 32
Figure 3.3: Oriented Bounding Box .............................................................. 33
Figure 3.4: k-Discrete Orientation Polytope .................................................. 33
Figure 3.5: 2D Axis Aligned Bounding Volume Tree ..................................... 36
Figure 3.6: Bounding Volume Tree for Point Clouds ..................................... 37
Figure 3.7: Bounding Volume tree and implicit equation BSP-tree combination .... 38
Figure 4.1: Solid Representation Hierarchy for interference analysis ............... 44
Figure 4.2: Binary Space Partition describing a cylinder ............................... 47
Figure 4.3: Regionalizing of the solid interior .............................................. 48
Figure 4.4: Profile for an Axially Symmetric Probe ...................................... 49
Figure 4.5: Structure of the probe representation in general form .................. 51
Figure 5.1: Direction of Force Resolution .................................................... 55
Figure 5.2: Equivalent body forces .............................................................. 57
Figure 7.1: Schematic Diagram of System Implementation ............................. 65
Figure 7.2: Suzuki Haptic Device ............................................................... 66
Figure 7.3: Tool Profile Example ................................................................. 67
Figure 7.4: Reduced Discontinuity ............................................................... 70
Figure 7.5: Haptic Simulation Objects .......................................................... 72
Chapter 1: Introduction and Background

1.1 Haptic Systems

A haptic system creates the feeling of touch through robotic means. The goal of a haptic system is to provide the user with a physical means of interacting with a simulated environment. Prior to the availability of haptic devices, human immersion in simulated environments was limited to visual and audio perception and/or interaction. Haptic devices add the sense of touch to augment the information that can be made available to the human user.

Due to the computational complexity of simulating contact situations and the high bandwidth requirements of haptic feedback systems, haptic systems have been limited in their ability to simulate contact interactions. The computation of force feedback must be completed approximately 1000 times per second [Massie 96] in order for the haptic system to realistically re-create the feeling of contact situations. Early haptic demonstrations had been limited to computationally simple tasks such as simulating the contact interaction between a point and a plane. Developments in haptic systems led to more useful contact simulation situations such as point to convex object contact, ray contact with a convex union [Basdogan 97], and point to NURBS surfaces [Hollerbach 97].

The use of a point (or a sphere) as the controlled entity in the virtual simulation greatly simplifies the computational needs of haptic systems. Under this simplified situation, meeting the 1000Hz requirement is easier [Massie 96]. A single point eliminates the need for tracking and computing orientation information for the point entity. As a result, only
three degrees of freedom need to be detected and actuated by the haptic device. This simplification greatly reduces the computations needed within the system. The drawback of limiting the interaction to a single point entity is the lack of true solid representation. Massie states “...when using the device, one's hand can physically pass through the volume occupied by a virtual sphere, while only the finger tip is constrained to remain outside the virtual sphere [Massie 96].” Furthermore, to truly simulate solid interactions, more than three degrees of freedom are needed. Whereas the geometric information of a point in space is unchanged by rotation, further consideration is required for arbitrary solids under rotation.

Members of the MIT Touch Lab have improved upon the single point methods to achieve haptic feedback for a ray interacting with a union of convex objects [Basdogan 97]. This development improves upon the single point model by eliminating the undesirable effect of passing though a “solid” object; however, a ray is still not a true solid and has limitations in solid to solid interaction simulations. Furthermore, the method is limited to interaction with a union of convex objects. In general, decomposition of solids into convex components is difficult, and objects exist that cannot be decomposed into convex sub-units.

Hollerbach et al. have successfully created a system that allows the interaction with a non-uniform rational B-spline (NURBS) surface [Hollerbach 97]. This technique enhances the complexity of objects with which haptic interaction is possible. However, the method limits the interaction to probing the NURBS surface with a point. As with all systems that use a single point entity to interact with the virtual environment, the method
presented by Hollerbach suffers the same drawback described by Massie: non-realistic solid interaction as the user's hand may pass through the object of interaction.

1.2 Collision Detection Methodologies

At the root of the software that runs a haptic system is the collision detection method that determines when entities of the simulation are in a state of interaction. There are many approaches to addressing the collision detection problem. Three main factors contribute to the quality of a particular methodology: speed, accuracy, and wealth of information. Methods that are very good at one factor tend to be less satisfactory in the others. Accordingly, each method has advantages and disadvantages for any particular application. The following sections describe current approaches to collision detection and comments on the applicability of these methods to haptic systems.

1.2.1 Lin-Canny Algorithm, I-Collide and V-clip

Lin and Canny present an efficient method of calculating collisions between convex polyhedra [Lin 93]. The method takes advantage of temporal and spatial coherence and operates especially well when the objects rotate at a speed less than one half a rotation per collision detection cycle. In addition to determining a collision event, the method gives proximity information. Application of the Lin-Canny method is limited to convex objects. Methods to break non-convex objects into a union of convex object could be used, but the additional complexity of these methods limits the practicality of such an approach. I-Collide is a collision detection package written by the computational geometry group at University of North Carolina. I-Collide extends the Lin-Canny approach to an n-body simulation with penetration depth information; however, the objects of the collision
detection are limited to convex polyhedra [Cohen 95], and therefore suffers the same limitations of the Lin-Canny method. Mirtich has improved upon the Lin-Canny approach with the V-Clip collision detection method, which enhances the robustness of the Lin-Canny approach and adds penetration depth calculation [Mirtich 98]. Again, the technique deals with convex objects so that non-convex objects must first be decomposed into the union of convex objects.

The fundamental drawback of these collision detection techniques is the restriction to convex or the union of convex objects. These techniques are not applicable to highly non-convex objects whose complexity can not be easily decomposed into convex subunits. As a result these methods would not be a good basis for creating a haptic system between arbitrary solid objects.

1.2.2 Triangle Set Methods with Bounding Volume Hierarchies

Triangle set methods represent the surface of each object as a set of triangles and report whether or not any triangle of one set intersects any triangle of the other set in each collision detection query. Triangle sets are capable of representing both convex and non-convex objects. Furthermore, the methodologies no not even require that the object be a coherent solid; any set of triangles is applicable. To increase the efficiency of the triangulated collision detection methods, programmers incorporate bounding volume hierarchies into the methodologies. Gottschalk et al. introduce the use of Oriented Bounding Boxes in triangle set methods [Gottschalk 96], and Held et al. propose the use of k-Discrete Oriented Polytopes [Held 96]. These bounding volumes and their associated bounding volume hierarchical trees enable the collision detection technique to compute
the solution more quickly in the average case. Bounding volume hierarchies are discussed further in Chapter 3.

The additional information that these methods may report include a list of the intersecting pairs of triangles and the number of intersecting triangle pairs. Penetration depth is not available from the triangulated method. Furthermore, the method has limitations in its ability to detect triangles that are completely in the interior of the other solid. The same flexibility that allows the triangle set to represent both convex and non-convex solids forces the dismissal of interior and exterior topological information. Triangle set methods are excellent if additional information beyond the detection of collision is not needed.

Triangulated surface methods with bounding volume hierarchies may have the speed necessary to meet the cycle time requirements of a haptic system; however, the inability to detect intersection within the interior of the solid objects and the lack of penetration depth information makes these triangulated model methods unsatisfactory for haptic system collision detection. Penetration depth is a fundamental necessity for haptic rendering.

1.3 Full body collision detection for haptic systems

To enable a haptic system to represent the reaction forces involved between two fully three-dimensional objects, the underlying collision detection algorithm must be able to detect contact between the objects and formulate an appropriate force response. The collision detection method must not be limited in the types of solids that may participate; both convex and non-convex objects should be applicable. The force response must be representative of the reaction force created by the contact. And the collision detection and
force response calculation must be performed quickly so that the one millisecond cycle
time is met.

1.3.1 Our System

To simulate contact between solids, the haptic device used in the system must accommodate degrees of freedom beyond translational coordinates. A general three dimensional solid has orientation information as well as position information, and this information must be addressed in a haptic system attempting to achieve full 3-D contact response. Furthermore, the haptic device should actuate a rotational force response in addition to a translational force response in order to represent the full body nature of the simulated solid interactions most realistically.

To drive full body haptic responses in the device, the governing program must detect when the interacting solids collide, determine the extent of the collision, and determine the proper response to the collision. We have implemented software capable of realistic full body interactions, and linked the software to a Suzuki 5-degree-of-freedom (DOF) haptic device. We present here the implementation and an extension to full 6DOF interactions. In the 5DOF case, the control entity is assumed to be axially symmetric. When an object is axially symmetric, rotation about the axis of symmetry does not effect the relative geometry of the object with respect to its environment. For 6DOF, the controlled entity may be any solid object, and a 6DOF haptic device should be used.

As discussed in Section 1.2, many researchers have investigated the problem of collision detection and have developed various solutions to the problem. Unfortunately, these solutions are either too slow for haptic applications, do not provide sufficient information for haptic force response calculation, or are limited in the geometry of the
solids involved in the calculations.

To address the collision detection and analysis needs of a haptic system, we have designed and implemented a rapid interference detection and analysis method. The data structures representing the solid objects within the haptic simulation environment possess the necessary information required for haptic interfacing. Furthermore, the method exploits some features of previous collision detection methodologies to increase its efficiency and meet the 1000 Hz bandwidth requirement.

1.3.2 Application

The creation of a 5DOF haptic system has an immediate application in the area of CNC Machining tool path planning. Current methods for tool path creation involve Computer Aided Manufacturing (CAM) systems that determine a machining tool path based on the geometry of the target part. These paths are then verified and with additional software. Errors are for complex geometries resulting in an invalid tool path that does not cut enough material or removes material that should have remained. The former is easily corrected by removing additional material; the latter, however, is a much more serious issue since this error would render the part unusable. Correction of the latter involves manipulation of the tool path planning software to produce a correct path plan.

In many cases, the general path characteristics are easily determined by a human user. Qualitatively, a human machinist can describe the path a milling machine should take to achieve the desired final product. However, a human is not able to give exact quantitative instructions to a CNC Machine. We propose a system that combines the wisdom and experience of a machinist with the power of a computer system to create correct tool paths.
for an arbitrary object. A haptic interface will bridge the knowledge of a human with computing power and precision.

A haptic system allows the user to interact with a virtual solid environment by controlling a virtual entity within the simulation. For the purpose of tool path planning, the virtual solid environment is the target part that is to be machined and the virtual entity is the cutting tool (or tools) that will be used to create the part. Following this paradigm, the haptic system allows the user to touch the part by controlling the position and orientation of the tool.

With the haptic system, the user can select appropriate tools for machining the desired part, and indicate how the tool should cut the part by touching the part in the manner that machining is desired. The computer gathers information about how the tool is able to contact the desired part, and from this information tool paths can be derived [Laxmiprasad 98].

1.4 Thesis Overview

Chapter 2 describes the new approach for interference detection as a method for collision detection only. In other words, solution to the “yes or no” question, does object A collide with object B. Chapter 3 explains how bounding volume hierarchies facilitate the computation load for the method described in Chapter 2. Chapter 4 extends the method of Chapter 2 to include interference analysis so that information beyond mere detection of collision may be determined easily from the methodology. Chapter 5 explains how the additional information derived from the interference analysis method can be used to calculate a force response for a collision that can be applied to a haptic system. Chapter 6
explains the application of a 5DOF haptic system to CNC tool path generation. Chapter 7 discusses the results of a 5DOF haptic system, and Chapter 8 concludes with a discussion of the newly created haptic system.
Chapter 2: Collision Detection

A the very least, a collision detection method determines whether or not two geometric entities have a non-empty intersection. The result is a simple yes or no. Due to limitations on participating geometries or insufficient information available from the methodology, previous approaches discussed in section 1.2 are not applicable to a full three dimensional haptic system. To enable a haptic system to operate under full three dimensional conditions we developed a new collision detection method that specifically addresses the needs of haptic systems. This chapter explains the representations selected for this new approach and explain how collision detection is determined from the models.

2.1 Solid Modeling Hierarchy

Any solid object divides space into two regions, the interior and exterior; the boundary between the interior and the exterior is the surface of the solid. A triangulated representation of the surface boundary is an approximation of the actual surface boundary in which the surface is broken into many triangles that collectively represent the surface boundary. When the triangulated surface is composed of very small triangles, the approximation becomes very close to the actual surface. Each triangle in the surface representation is bounded by three edges, which in turn are defined by three vertices. The hierarchy of solid representations is illustrated in Figure 2.1.

As shown in Figure 2.1, determining the intersection between exact representations of the solids would give the most complete information about the collision. From the standpoint of calculating a proper force response, such an operation would yield excellent
information. However, the complexity of the operation would likely exceed the timing constraints of a simulation or virtual reality environment. Using exact solid to exact solid collision query also has the added disadvantage of having to determine the exact representation of an arbitrary solid. In general, exact representations are cumbersome. Performing the collision query on the exact surface to exact surface has the same disadvantages as the exact solid to exact solid method.

Since exact representations are generally too computationally expensive, an approximation of the surface boundary must be used. Several collision detection methods listed in literature use triangulation approximations of the solids’ surface to perform
collision detection between solids. Very fast procedures for triangle representation to triangle representation collision detection have been presented in the literature [Gottschalk 96] [Klosowski 97].

In the following sections, we outline a new method for collision detection. The proposed collision detection method determines intersection between an exact solid representation and a modified vertex representation. The reason for these particular solid model representation methods will not be immediately clear in this chapter. In Chapter 4, the reasons for choosing these particular solid models will be explained as the collision detection procedure outlined here is expanded to include interference analysis that determines more information about the state of collision beyond simply that collision has occurred.

2.2 Implicit Equation model for Solid Representation

An implicit equation is any equation of the form \( f( ) = 0 \). In three dimensions an implicit equation is of the form \( f(x, y, z) = 0 \), where \( x, y, z \) can represent the cartesian coordinates in space. Points \((x_0, y_0, z_0)\) that satisfy the equation \( f(x, y, z) = 0 \) form a surface in space. This surface is called an implicit equation defined surface, or implicit surface. The properties of implicit equations make them a natural choice for representing solids because geometric properties of solids have a analogous algebraic property in implicit equations [Ganter 93].

The boundary of a solid divides space into two distinct regions, the interior and the exterior. The interior is considered part of the solid; the exterior is all of space that does not belong to the solid. The surface that divides these two regions is the boundary of the
solid. Mathematical equations of the form \( f(x, y, z) \) exhibit a similar geometric property since the surface defined by \( f(x, y, z) = 0 \), divides space into two distinct regions. In one of these regions, evaluation of \( f(x, y, z) \) at each point \((x, y, z)\) yields a positive number. A relationship between interior of a solid and exterior of a solid can be made with the two regions created by an implicit equation \( f( ) = 0 \). After arbitrarily equating the negative region with the interior of a solid, evaluation of a point’s coordinates in an expression \( f(x, y, z) \) determines whether or not the point is part of the solid’s interior, exterior, or boundary.

A simple example of an implicit equation’s ability to represent a solid object is the equation for a unit sphere: \( x^2 + y^2 + z^2 = 1 \). For points within the sphere’s interior, the expression \( x^2 + y^2 + z^2 - 1 \) evaluates as a negative number, points in the exterior evaluate to a positive number, and points on the boundary evaluate as zero. In this manner, a mathematical equation effectively represents a solid body.

### 2.2.1 Multiple Implicit Equations for Complex Solids

A sphere is represented simply and compactly by a single implicit equation. However, more complex geometric solids may require multiple implicit equations to fully represent the solid. For example, a simple finite cylinder requires several equations to adequately describe its geometry:

\[
\begin{align*}
  x^2 + y^2 - r^2 &\leq 0 \\
  -h + z &\leq 0 \\
  -h - z &\leq 0
\end{align*}
\]

All three equations must be satisfied to be in the interior of the cylinder. If any one of the equations is not satisfied, then the point is in the exterior of the cylinder. In this
manner, multiple equations may be used together to form more complex solid models through AND (intersection) and OR (union) combinations of equations. Alternatively, implicit equations can be collected in a binary tree to divide space into regions. Each region can then be designated as part of the interior or exterior of the solid. The former method is referred to as Constructive Solid Geometry, and an example is given in section 2.2.2. The latter method, we call an implicit equation binary space partition tree, and an example is given in section 2.2.3.
2.2.2 Constructive Solid Geometry

Figure 2.2 illustrates how Constructive Solid Geometry (CSG) techniques combine several implicit equations through set union and intersection operations to construct the final object. Construction begins with simple solids such as spheres, cylinders, and half-spaces. By taking the intersection between the sphere and a half space, the half sphere can be constructed. The finite cylinder is derived from the equation for an infinite cylinder by intersection with half spaces. Finally the half sphere is combined with the finite cylinder through a union operation to create the final object. For the purpose of determining whether or not a point \((x_0, y_0, z_0)\) is in the interior of the solid, the compound statement Equation 2.2 will indicate whether or not the point is in the interior or exterior. If Equation 2.2 is true, the point is in the interior; if Equation 2.2 is false then the point is in the exterior.

\[
(x \leq 0 \land x^2 + y^2 + z^2 - 4 \leq 0) \lor (x > 0 \land x \leq 6 \land y^2 + z^2 - 1 \leq 0)
\]  

(2.2)

2.2.3 Implicit Equation Binary Space Partition Tree

Figure 2.3 illustrates how the same object created in Figure 2.2 by constructive solid geometry, can be defined by an implicit equation binary space partition tree (BSP-tree). At the root of the tree, all of space has yet to be divided into regions. The implicit equation \(x = 0\) divides space into two halves, one where \(x < 0\) and one where \(x > 0\). Continuing down the tree, the equation \(x^2 + y^2 + z^2 - 4 = 0\) divides the half space \(x < 0\) into two regions. Similarly, the equation \(y^2 + z^2 - 1 \leq 0\) divides the half space \(x > 0\) into two regions. Following the tree, all of space is systematically divided into pieces so that at the leaves of the tree, each leaf corresponds to a unique region of space. Each of these regions is then
designated as either part of the solid interior (1) or part of the exterior (Ø). For simplicity, the surface boundary of the object is considered to be part of the solid interior (1).

Figure 2.3: Binary Space Partition

Given a point \((x_0, y_0, z_0)\), determining whether or not the point is in the interior simply requires following the appropriate path set by the tree after evaluating the equation at each node. At the root, we evaluate the point \((x_0, y_0, z_0)\) against the statement, \(x = 0\). If \(x_0 \leq 0\) then the \(x \leq 0\) path is selected, otherwise the \(x > 0\) path is followed. The equations are evaluated against the coordinates \((x_0, y_0, z_0)\) at each node and a tree path is established for the point.
At the end of the tree path, the value of the final node indicates whether the point is in the interior or the exterior.

Either a constructive solid geometry representation or an implicit equation binary space partition tree representation could be used to represent a solid so that, given a point \((x_1, y_1, z_1)\) determining whether or not the point is in the interior of the solid is a simple task. For reasons that will be more clear in Chapter 4, we choose the implicit equation binary space partition representation.

### 2.3 Point Cloud model for Solid Representation

As shown in Section 2.2, collision detection between an implicit equation defined solid and a single point is a simple task. To achieve collision detection between the implicit equation defined solid model and some other solid, the second solid must have a structure that can be easily compared to the implicit equation model. By defining the second solid as a collection of points, comparison of the models for overlap becomes simply a task of comparing each point of the second model to the implicit equations of the first model.

A solid represented by means of a collection of points is not nearly exact. However, approximations, even those that reduce a coherent solid to a disjoint set of points, can capture the geometry of the solid within a certain tolerance if number of constitutive elements is sufficiently large. To accommodate collision detection against an implicit equation model described in section 2.2, we select a representation consisting of numerous points that lie on the surface of the solid. Ideally, the point distribution is such that the number of points per unit area is uniform and the gap between points is small.
Section 1.2.2 discussed the use of triangulated models to represent solids, and a collision detection method for these triangulated models. The point cloud representation of solids suffers the same inaccuracy limitations as the triangulated model. Whereas smaller triangles, which necessitate a higher number of triangle elements in the tessellated representation, lead to greater accuracy, more points in the point cloud also lead to better accuracy. The point cloud representation suffers an additional inaccuracy with the unavoidable gaps between points.

A well defined triangulated model can be ‘watertight,’ meaning an infinitesimal body in the exterior of the triangulated representation cannot pass into the interior of the representation without crossing the representation itself. Point clouds do not have this property. Objects that are small enough can easily pass though the cloud undetected. To avoid the situation of grossly missing contact, the gap between points should be less than the smallest feature of the implicit equation defined object. If the implicit object is very small, then with respect to the point cloud model, the implicit object is a point, and point methods [Hollerbach 97] [Salisbury 97] are appropriate.

The advantage of the point cloud representation lies in its simplicity of representation and ease of intersection with implicit equation defined solids. The disadvantages stem from its approximate nature and voids. When selecting a collision detection method, the advantages and disadvantages of each representation must be considered. The following collision detection method works best under conditions where features of the object represented by implicit equations are not significantly smaller than the point cloud point density.
Figure 2.4: Point Cloud Representation

Figure 2.4 illustrates a point cloud representation. While the point cloud is not topologically complete, the implicit equation defined object not only has a fully defined surface, but the interior is fully defined as well. Therefore, points of the point cloud cannot pass into the implicit equation object undetected.

2.3.1 Point Cloud Generation

A point cloud representation can be created in a variety of ways. For efficient use of points, the number of points per unit area should be consistent across the surface of the solid. Nearly even distributions are sufficient. A simple and effective method takes a random sample of points from a surface representation of the solid. Surface representations include tessellated surface approximations and NURBS representations.

In creating a point cloud representation for a particular application two competing factors influence the choice of point cloud density. A large number of points contributes to a more accurate representation of the solid and helps mitigate the possibility of missed collisions. Unfortunately, more points contained in the point cloud representation require added computation time. As a result, the selection of point cloud density must balance the needs of representational accuracy and computation speed.
2.3.2 Combination Representation

The fundamental drawback of point clouds is the fact that it is a discrete representation for a continuous object. Furthermore, point cloud representations lack a clear connection between the individual points and the object that the collection of points represents. In some applications, combining a point cloud representation with a higher level solid representation offers additional support to the point cloud alone. Furthermore, each point of the point cloud can be associated with features of the higher level solid representation. For example, the combination of a triangulated surface mesh and point cloud could connected by associating each point with the triangle to which it belongs.

2.4 Collision Detection Method

The analysis method must determine whether or not any point of the point cloud intersects the implicit equation represented solid. Repeatedly following the procedure described in section 2.2.3 for each point of the point cloud will determine whether intersection occurs. For point clouds with many points, this approach quickly becomes computationally expensive. To meet the speed requirements set for our collision detection method, chapter 4 will explain the use of bounding volume hierarchies and how they apply to collision detection methodologies.

Pseudo code1: Check(A, B) -- method for checking point cloud A against BSP-tree B

```plaintext
Check(A,B)
10  for(i=0; i<NumberofPoints(A); i++)
11     if (AuxCheck(A[i],B) == COLLISION)
12        return COLLISION
13     else
14        skip

AuxCheck(p,B)
20  if B is a leaf
```
if (B == 1) return COLLISION
else return NO COLLISION
else if implicit function of B evaluated at point p > 0
    AuxCheck(p, LeftChild(B))
else
    AuxCheck(p, RightChild(B))

The pseudo-code for Check(A, B) illustrates how a point cloud is checked against a BSP-tree. Each point of point cloud A is verified using AuxCheck(p, B), where p is one point of point cloud A. In line 23, the implicit function associated at the root of B is evaluated at point p. Depending on the sign of the evaluation, either line 24 or line 26 is executed, which calls upon one of the children to continue to the evaluation process. When the procedure reaches a leaf, line 20 is TRUE and either line 21 or line 22 executes which indicates whether the point p is in the interior (COLLISION) or exterior (NO COLLISION).

2.5 Discussion

Because the point cloud representation is an approximation of the continuous surface of the environment, the possibility of undetected contact exists. In general, undetected contact occurs only when the penetration depth is very small. The problem of undetected collision can be minimized by using a high point density for the environment; however, computation time will increase accordingly. Furthermore, missed contacts can not be completely avoided. The approximate nature of a point cloud representation permits the possibility that grazing contact between the exact surface of the implicit object may contact the exact surface of the solid represented by the point cloud without actually contacting a point of the point cloud, see Figure 2.5.

Approximation of the environment is a common drawback to any simulation. The computational power to achieve exact modeling and collision detection between two
general objects is not available. Even with tessellated surface models, the tessellations themselves are approximations of the actual surface. Collision detection methods for tessellated models may be exact for the models, the approximate nature of the models themselves lead the overall result to be subject to the same type of undetected grazing contact between the actual solids.
Chapter 3: Bounding Volume Hierarchy

The collision detection method described in Chapter 2 is correct because it successfully determines the subset of points of the point cloud that are in the interior of the solid determined by the implicit equations. Because accurate representations require a large number of points for the point cloud model, the collision detection method, described in the previous chapter, is inefficient in determining the solution. The procedure, as described so far, asks that we check each point individually for its inclusion within the implicit equation defined solid's interior. As the number of points in the representation increases, the computation time increases proportionally. In this chapter we describe a technique for adapting bounding volume hierarchy methods used in other collision detection procedures to facilitate the collision detection method described in Chapter 2.

3.1 Bounding Volume Properties

A bounding volume is a geometric solid whose interior completely encloses the entity that the volume is designated to bound. For the purposes of facilitating collision detection methods, bounding volumes that most closely approximate the features of the bounded entity perform better. The bounding volume plays the role of a simplified approximation of the actual entity. The simplified form is used in the first steps toward collision detection to focus upon portions of the entities that are most likely in a state of collision.

The geometry of the bounding volume is generally less complex than the geometry of the entity that it bounds. A less complex geometry contributes to making intersection analysis between bounding volumes easier. For example, the intersection test between two
spheres only requires comparing the distance between the center points of the spheres with their radii.

![Diagram of non-overlapping bounding volumes](image)

**Figure 3.1:** Non-overlapping bounding volumes

If two bounding volumes do not overlap, then no entity within one bounding volume can overlap any entity of the other. A single bounding-volume-to-bounding-volume overlap check has the potential to give significant information about state of collision of all entities within the volume. When bounding volumes overlap, then no inference can be made about the state of collision of the bounded entities.

Two main characteristics of a particular bounding volume contribute to its suitability for facilitating collision detection procedures. One is the ease of determining whether two bounding volumes overlap. The second is the ability of the bounding volume to conform to shape of the entity that it bounds. Generally, bounding volumes that have simple overlap tests are less efficient at conforming to the shape of the entity that it bounds, and bounding volumes that have the flexibility to better conform to shapes are harder to check for overlap. In the example of spherical bounding volumes, the intersection test is simple; however, spheres do not have great flexibility to conform to arbitrary shapes. The only variation available with spheres is size (i.e. radius).
Better conformity to the shape of the bounding entity minimizes the occasions when bounding volumes overlap and the entities within the bounding volumes do not intersect. These occurrences are undesirable because when bounding volumes overlap, the collision detection method must assume that the entities within the bounding volumes could potentially intersect until the method can prove otherwise. Conformity between the bounding volume and the actual entity reduces the need for these additional computations. The advantage of low computational cost for bounding volume overlap test is more obvious as it contributes to a faster method by reducing the amount of computation needed for testing bounding volume intersections.

3.2 Bounding Volume Examples

The choice of bounding volume can greatly affect the computation time in a collision detection method. The ideal bounding volume possesses high flexibility in geometric shape and a computationally simple method of comparing bounding volumes for overlap. Generally, bounding volume geometries that possess high geometric flexibility have more complicated overlap tests. The following examples of bounding volumes illustrate the properties of bounding volumes and how these properties contribute to collision detection methods.

Several researchers have proposed methods that involve bounding volumes. The two most seriously considered for the accelerating the collision detection method of Chapter 2 included Oriented Bounding Box (OBB), as described by Gottschalk et al [Gottschalk 96], and the k-Discrete Oriented Polytope (k-DOP) construct described by Held et al [Held 96].
3.2.1 Axis Aligned Bounding box

Surfaces that align with the axes are easily bounded by Axis Aligned Bounding Boxes. Surfaces that are not aligned with the axes are bounded less efficiently with AABBs.

Figure 3.2: Axis Aligned Bounding Boxes

An axis aligned bounding box (AABB) is a rectangular prism whose faces are aligned with the x-y-z coordinate axes. Like, the sphere, an AABB has a relatively simple overlap test. If bounding boxes overlap, then no separating plane exists such that the volumes are entirely on opposite sides of the separating plane [Klosowski 97]. For AABB, at most three interval tests are needed to check the overlap status of the bounding boxes. If any one interval test shows no overlap, then the remaining interval tests are not needed. Each axis aligned bounding box is defined by three intervals: the intervals over which the box exists along the x-axis, y-axis, and z-axis. A single interval test compares the interval of one box to the corresponding interval of the other.

3.2.2 Oriented Bounding Box

The Oriented Bounding Box (OBB) extends the capabilities of the AABB by removing the requirement that the faces of the rectangular prism must be aligned with the coordinate axes. The removal of this restriction gives the bounding volume greater flexibility to conform to the shape of the object that it bounds. However, the overlap test becomes considerably more complex. Gottschalk et al. have developed an efficient
An Axis Aligned Bounding Box does not conform well to all objects. An Oriented Bounding Box is better able to conform to the shape of an arbitrary object.

Figure 3.3: Oriented Bounding Box

The method has been implemented in the triangulated object collision detection package RAPID.

3.2.3 k-Discrete Orientation Polytope

Where the OBB adds flexibility to the AABB by allowing arbitrary orientation, the k-Discrete Orientation Polytope (k-DOP) adds flexibility by increasing the number of faces on the bounding volume. A k-DOP is constructed by the intersection of $k$ half spaces. The outward normal of these half spaces are restricted to be in certain critical directions. Hence, the $k$ in k-DOP refers the number of critical directions [Klosowski 97]. Like the AABB, the critical directions are constant. The k-DOP merely adds additional critical directions. In fact, an AABB is a subset of k-DOP where the critical directions are the
coordinate axes and \( k = 6 \). The overlap test for k-DOP bounding volumes is similar to the AABB overlap test, except the number of interval tests increases to \( \frac{k}{2} \).

### 3.3 Hierarchy

As stated in Section 3.1 and illustrated in Figure 3.1, the utility of the bounding volume relies on the fact that non-overlapping bounding volumes yield definitive information about the entities that they bound. When bounding volumes overlap, little information about the bounded entities is directly determined. In an effort to obtain useful information by means of bounding volumes, a hierarchy of bounding volumes is implemented to gather information in stages. Bounding volumes have been used to add efficiency to collision detection queries between tessellated triangle models [Gottschalk 96] [Klosowski 97].

In the case of triangulated models, the collision detection methods must determine which triangles from one object intersect the triangles of the other. At worst, the method must check each triangle of one object to each triangle of the other. The number of triangle pairs is \( n \times m \) where \( n \) and \( m \) are the numbers of triangles in each object representation.

A bounding volume hierarchy increases the efficiency of a collision detection query, on average, by focusing the method on potential colliding components while quickly dismissing portions of the entities that are clearly not in a state of collision. The hierarchical technique divides each object in half and bounds the halves with smaller bounding volumes. The original bounding volume is called the parent of the smaller bounding volumes, which are called the children. Thus, the hierarchy for each object is a
binary tree of bounding volumes. These smaller bounding volumes are then tested for overlap only when the parent has tested positively for overlap.

The root of the tree is a bounding volume that contains all the triangles that compose the object, and each child is a bounding volume that contains half of the object section contained in its parent. The leaves of the binary tree contain exactly one triangle. Figure 3.5 illustrates an example of a bounding volume hierarchy in which the entities bounded are triangles. At the root, all triangles are bounded by a single volume. At each successive level, the children divide the triangles of the parent into two subsets, and each subset is bounded by a bounding volume.

If at any time, a particular bounding volume overlap test reveals there is no intersection, then the children of the bounding volumes need not be checked, and all of the triangles (or whatever entity) in the participating bounding volumes are known to be in a non-intersecting state.

In the worst case, the number of triangle to triangle intersection tests that must be performed is the same as the brute force method. However, in the brute force method, this worst case occurs for all collision queries. With the hierarchical bounding volume method, the worst case will occur only if nearly all triangles are in a state of collision. Such an event is unlikely to occur in typical practices.

3.3.1 Point Cloud Bounding Volume Hierarchy

Application of these methods to our point cloud to implicit equation defined solid models requires only slight modification of these techniques. The hierarchy is very much similar to the hierarchy used in bounding triangulated models. The root bounding volume encloses all points of the point cloud. The children of the root bounding volume enclose a
Figure 3.5: 2D Axis Aligned Bounding Volume Tree

subset of the points of the point cloud such that every point belongs to one of the child bounding volumes. The hierarchy continues to subdivide the points into subsets until the number of points is less than some predetermined value. When the number of points is less than the value specified, the bounding volume that encloses them is considered a leaf of the bounding volume binary tree hierarchy.
3.3.2 Implicit Equation BSP-Tree Bounding Volume Hierarchy

The implicit representation of the solid must also have a bounding volume hierarchy associated with it, so that the bounding volumes of the point cloud representation can be compared to the bounding volumes of the implicit representation. At the very least, the implicit representation may be bounded by a single root bounding volume. Generally, a tree structure such as the one described for triangulated models or point cloud models will increase performance. Establishing a bounding volume hierarchy for implicit equation binary space partition trees is not as straightforward as the triangulated model case.

For the point cloud model, the hierarchy affects the collision detection methodology by dividing and organizing the large number of points into smaller, more manageable subsets. In the implicit equation binary space partition method, a tree structure already
exists that allows the method to focus on the relevant portion of the implicit equation defined object. In this sense, a single root bounding volume may be sufficient since following the BSP-tree very quickly determines whether a point is in the interior or not. However, combining the BSP-tree with the bounding volume tree can be even more beneficial if the bounding volume tree is able to prune non-colliding points more effectively than a single root bounding volume.

![Diagram of BV Tree Root and BSP Root](image)

**Figure 3.7:** Bounding Volume tree and implicit equation BSP-tree combination

In this case, the main contribution of bounding volumes is conformity to the shape of the original object. Objects that conform easily to a single bounding volume will gain less
from a bounding volume hierarchy. Objects with much exterior space contained in a single bounding volume will benefit from a hierarchy since smaller components may avoid enclosing the undesirable exterior space. Furthermore, the implicit equation BSP-Tree that originally defined the object interior from all space, can be broken into sub-trees for each leaf of the bounding volume hierarchy. This process adds some complication to the definition of object representation, but can increase the efficiency of the collision detection procedure. The overall structure of the implicit equation BSP-tree defined object is illustrated in Figure 3.7.

3.4 Collision Detection Using Bounding Volume Hierarchical Trees

Figures 3.6 and 3.7 show the overall organization of the data structures that represent the two objects involved in the collision detection query. Both are rooted with a bounding volume that encloses the entire object. The collision detection query begins with these root bounding volumes.

An intersection test of the root bounding volumes will determine whether the two volumes are disjoint or overlapping. If the volumes are disjoint, then the collision state of the two objects is clearly non-colliding and the procedure is complete. If the volumes are overlapping, then further testing is required.

Under the condition that the two root volumes are overlapping, the children of the root bounding volumes are checked for overlap with each other. If overlap is discovered, then the procedure continues to check bounding-volume-to-bounding-volume by descending down the hierarchical tree. During this process, if two bounding volumes are found to be disjoint, then no further investigation is done for that bounding volume pair. This process
continues until the bounding volumes being tested for overlap are leaves of the hierarchical trees. When two bounding volumes are overlapping and both are leaves of their respective bounding volume tree, then the contents of the leaves must be considered for collision.

When the contents of the bounding volumes are checked for overlap, the procedure follows the method described in Chapter 2 for collision detection between a point cloud and implicit equation BSP-Tree. In this case, however, the point cloud is a subset of the entire object point cloud and the BSP-Tree is a modified BSP-tree that applies to the bounding volume in question. Each point within the leaf bounding volume of the point cloud is checked for collision against the implicit equation BSP-tree of the leaf bounding volume of the implicit equation. When a point is found to be in the interior defined by the BSP-tree, the procedure returns TRUE to the collision detection query. If the procedure completes checking all bounding volume pairs and no point in the interior is found, then the procedure returns FALSE to the collision detection query, indicating that no collision occurs between the two objects.

The described procedure is a recursive method: Let A be an object represented by point cloud with bounding volume hierarchy, and B be an object represented by implicit equation BSP-Tree with bounding volume hierarchy.

**Pseudo code 2: Hierarchical search through Bounding Volume Tree**

```plaintext
BVcheck(A, B)
if BV(A) overlaps BV(B)
    if A != LEAF
        BVcheck(RightChild(A), B);
        BVcheck(LeftChild(A), B);
    else
        if B != LEAF
            BVcheck(A, RightChild(B));
            BVcheck(A, LeftChild(B));
        else
```
check(PointCloud(A), BSPTree(B));

else
    skip

The function BVcheck(A,B) compares the bounding volumes of tree A to tree B to determine which leaves are in a state of overlap. When overlapping leaves are found, the function check(A,B) is called to determine the state of intersection be a set of points A, and a implicit equation binary space partition B as defined in section 2.4.
Chapter 4: Interference Analysis

Chapter 2 introduced the data structures for representing the solid models and a preliminary method for determining collision between these representations. Chapter 3 explained how bounding volume hierarchies can be used to increase the efficiency of collision detection between the two representations. The methodology, as discussed so far, is sufficient to answer the question “Does object A intersect object B?” In this chapter we explain how the methodology can be expanded to address the question “Does object A intersect object B, and if it does by how much?” Furthermore, we will use the same techniques for encapsulating depth information to encapsulate other important auxiliary data desired from the interference analysis. Section 4.1 discusses the importance of depth of interference to simulations and haptic systems. Section 4.2 revisits the solid model hierarchy discussed in Section 2.1 to explain why the method chooses a point cloud representation for one of the objects and an implicit equation model for the other. Section 4.3 explains what additions are needed in the implicit equation BSP-tree model to obtain penetration depth information quickly in the interference analysis.

4.1 Depth Information

The need for additional information describing the state of collisions drives the development of an interpenetration analysis method. Many applications require or benefit from information that describes the extent of interference between two objects. For example, physical simulations of interacting solid objects benefit from penetration depth information because the simulation can scale the physical response to the extent of
penetration. Traditional collision detection methods that simply determine whether or not collision has occurred limits the quality of a response to a basic yes or no response. With depth information the response may be scaled accordingly leading to a smoother, more continuous response.

In some situations, the step response of collision detection is sufficient or can be compensated with additional calculations. However, in many situations, the availability of depth information improves the quality of the system. In haptic systems, a virtual object interacts with a virtual environment with force feedback to indicate when contact is made. To maintain stability and realistic effects, haptic devices rely on force responses that scale with penetration depth. The step response of traditional collision detection is not sufficient for haptic devices. For proper force response feedback penetration depth information is essential.

4.2 Solid Representation Hierarchy

Section 2.1 listed a hierarchy of solid modeling methodologies. Of the representations presented, only the exact solid and exact surface models have full interior/exterior information available for use within an interference analysis method. Triangulated surface approximations, edge representations, vertex representations so not have interior/exterior information, and therefore possess insufficient information to reveal penetration depth, see Figure 4.1. However, if at least one of the two objects in the interpenetration analysis method is an exact solid representation (or surface representation), then the penetration depth information can be derived from this information even if the other representation
has no depth information. This observation relies on the fact that the amount by which object A penetrates object B is equal to the amount by which object B penetrates object A.

In a triangulated surface model, the solid is defined by the collective; individual triangles do not contain information about the interior. Even with an outward normal associated with each triangle, the interior information is limited. Surface equations, however, provide information about the boundary as well as data regarding the interior region.

The interference analysis method determines collision and penetration depth information by representing one object, by a mathematical equation representation of a solid, and the other object by a modified vertex representation. The implicit equation BSP-
Tree is used since it has the ability to encapsulate interior depth information where the triangulated method cannot. Furthermore, the structure of a BSP facilitates the calculations. The point set representation (point cloud) is selected for its simplicity and compatibility with the equations of the BSP-tree.

4.3 Binary Space Partition Model

For interference analysis, the task of determining penetration depth is added to the task of detecting collision. In analysis of the solid model hierarchy, we observed that the amount that object A penetrates object B is equal to the amount that the object B penetrates object A. In our interference analysis procedure object A has been defined to be represented by a set of points and object B is represented by a set of equations. To achieve the goal of quickly determining the extent by which a point of A penetrates the interior of B, an auxiliary equation is associated with the interior of object B for the purpose of penetration depth calculation. By evaluating this additional equation at a given point, the penetration depth of the given point is determined. The form of the equation must be designed to give the appropriate depth for a given probe solid. For example, the associated equation for the unit sphere located at the origin is:

$$d = 1 - \sqrt{x^2 + y^2 + z^2}$$ \hspace{1cm} (4.1)

According to Equation 4.1, points on the surface of the unit sphere have a depth of zero. Points in the interior of the unit sphere have positive depth, and points in the exterior of the sphere have negative depth. For more complex probe geometries, a collection of auxiliary equations may be needed to adequately describe the probe. Furthermore, since
penetration depth on the exterior of the object B is not needed, (collision has not occurred) these additional equations need only be defined for the interior sections of object B.

As noted in section 2.2, the BSP-tree includes equations that separate the interior of the solid from the exterior. Determining interference depth from a solid described by a collection of equations has the added complexity of determining which equation to apply for depth information. In other words, a solid defined by multiple surfaces has the added difficulty of determining which surface is closest to the point that is intersecting.

The example of a single sphere is simple; one equation describes the interior of the sphere, and one equation describes the depth information for the interior. As an added bonus, the same equation could be used to describe the “negative interference depth” (distance to the sphere) of points in the exterior of the sphere. In general, solids are not as simple as the sphere. For example, a relatively simple solid that illustrates added complexity is a finite cylinder. A cylinder has three surfaces of interest, the body of the cylinder and the two circular ends. Given a point in the interior, to find the interference depth, one must first know which of these surfaces is closest to the point in question.

Fortunately, the set of points in the cylinder that are closest to the one end of the cylinder surface reside in a contiguous region that can be easily defined.

Furthermore, each surface of the cylinder and its associated region of points that have that surface as its closest surface all have easily defined regions. The division of these regions can be described by implicit equation surfaces. These equations can be incorporated into the implicit equation BSP-tree. The compatibility with BSP-trees further justifies our selection of the BSP-tree over CSG methods. The BSP-tree structure can easily incorporate our need to uniquely identify a region within the interior of the solid so
that the corresponding equation describing the interference depth can be evaluated. Figure 4.2 illustrates the use of implicit equations to define a cylinder and divide it into appropriate regions. The cylinder and regions are shown in Figure 4.3.

4.3.1 Beyond Depth Information

In addition to depth information, the simulation may also need the direction indicating the shortest path to the solid exterior. At the surface of the solid, this direction is the surface normal. In the interior of the solid, this direction is the shortest distance to the probe surface. Just as an auxiliary equation provides depth information, a second auxiliary equation, in this case a vector equation, provides the direction information. Since the direction to the closest surface depends on which surface is closest, just as with the depth information case, a particular vector equation describing the direction to the surface is defined for each region. For the unit sphere located at the origin, the unit direction vector is

\[
\begin{bmatrix}
\frac{x}{\sqrt{x^2 + y^2 + z^2}} \\
\frac{y}{\sqrt{x^2 + y^2 + z^2}} \\
\frac{z}{\sqrt{x^2 + y^2 + z^2}}
\end{bmatrix}
\]  

(4.2)
This vector is the unit vector in the same direction as the ray originating at the center of the sphere and passing through the point \((x, y, z)\). Figure 4.3 illustrates the regionalizing of a cylinder and the auxiliary equations describing depth and direction.

\[
\begin{align*}
\text{Direction} & : [0, 0, 1] \\
\text{Depth} & : h - z \\
\text{Direction} & : \left[ \frac{x}{\sqrt{x^2 + y^2}}, \frac{y}{\sqrt{x^2 + y^2}}, 0 \right] \\
\text{Depth} & : r - \sqrt{x^2 + y^2} \\
\text{Direction} & : [0, 0, -1] \\
\text{Depth} & : h + z
\end{align*}
\]

**Figure 4.3:** Regionalizing of the solid interior

Together, the penetration depth and direction vector provide interior information that is not usually available in conventional collision detection methods. This information is important to physical simulations and virtual reality environments. Additional auxiliary equations can be used if the simulation needs more information about the solid's interior.

Standard CSG methods are adequate for defining the probe for basic collision detection. However, the auxiliary equation(s) required for interpenetration analysis make standard CSG inadequate for most probe solids. To simplify the development of equations for use as a probe in a virtual environment, the probe solid should be divided into regions. For this reason, the BSP-tree representation is preferred. Within a region, each quantity needing auxiliary functions should be describable by a single equation. If a region of the probe solid has a quantity that can not be described by a single equation, then that region should be divided into two or more regions. After dividing the solid into regions and
assigning appropriate auxiliary functions to each region, a BSP tree is constructed to allow
efficient determination of which region a point \((x, y, z)\) resides

With this construction: given a point \((x, y, z)\), the BSP tree is traversed to determine in
which region the point resides. If the region is part of the interior of the probe, then
collision has occurred and evaluation of the auxiliary functions associated with the region
determines penetration depth and direction.

4.3.2 Profile revolution

In many cases the probe solid possesses an axis of symmetry that can be exploited to
reduce the complexity of the BSP tree and the calculations needed to traverse the BSP-
tree. When the probe solid is axially symmetric the solid can be defined as a volume of
revolution. Without loss of generality, let the axis of symmetry coincide with the \(z\) axis. In

![Figure 4.4: Profile for an Axially Symmetric Probe](image)

49
this case, the solid can be represented by equations of the form \( f(d, z) = 0 \), where 
\[ d = \sqrt{x^2 + y^2} \]. Figure 4.4 illustrates an example. The profile definition simplifies the BSP-tree of probe by representing the three dimensional solid as a two dimensional profile. The BSP-tree then has a simplified task of dividing two-dimensional space into regions. Detecting collision between the solid and a point becomes a two dimensional problem, and locating the region to which a point belongs is less complex. Since the axially symmetric object will interact with a non-axially symmetric environment, contact situations break the symmetry of the BSP-tree object. Even though two dimensions are sufficient to identify a region, the associated auxiliary functions for each region must continue to address the full three-dimensionality of the solid for evaluation of direction within the identified region.

4.3.3 Bounding Volumes

Chapter 3 explained the use of bounding volumes to increase the efficiency of collision detection procedures. To gain the benefits from bounding volume hierarchical trees, both the point cloud representation and the implicit equation BSP-tree must have a bounding volume tree. Therefore, a bounding volume tree should be incorporated along with the BSP and auxiliary functions. Every portion of the solid representation must be accountable within the bounding volume tree, not just the surface boundary. In some cases, using a single bounding volume that encompasses the entire solid may be acceptable rather than developing subdivisions of the solid and building a bounding volume tree.

In either case, the overall structure of the probe representation begins at its highest level with a bounding volume that contains the entire solid. The leaves of the bounding
volume tree (or the single bounding volume if no tree is used) each have a BSP tree that divides the leaf bounding volume into regions. Finally, each region has auxiliary functions associated with it. Figure 4.5 illustrates the structure of the probe representation.

4.3.4 Advantage and Disadvantage

The probe representation has many advantages. Any finite computer model of a solid can be represented using the BSP approach outlined in section 4.3. The equation model is capable of representing solids exactly and maintains interior and exterior information. The auxiliary functions associated with the interior regions of the probe give flexibility to
allow various information to be conveyed depending on the requirements of an application. Furthermore, the equation representation is simple to evaluate for a given point in space. Geometric complexity is the main disadvantage of the probe representation method. While any solid model could be represented, the complexity of the method makes the representation less attractive for very detailed solids.

4.4 Interference Analysis Procedure

The interference analysis follows the same procedure as the collision detection method described in Chapters 2 and 3. Since the implicit equation BSP-tree incorporates region information about the solid interior, after a point is identified as in the interior, the associated auxiliary functions are evaluated for the point's location, and the auxiliary information for the given point is determined for use in the system.

The auxiliary functions can be used in many ways depending on the application. Each interpenetrating point of the point cloud is evaluated according to the probe representation criterion. In any collision, multiple points of the point cloud will intersect the probe. The results of the auxiliary function can be treated in many ways such as individual, summation, average, maximum, or minimum. The composite result applies to the simulation.

Other auxiliary functions may also be used. Where the depth and direction information represent geometric information intrinsic to the solid's geometry, additional auxiliary functions may be created to encapsulate information that is user defined. In Chapter 6, an additional auxiliary function will be introduced for use in a computer aided design and manufacturing system.
Chapter 5: Force Response Calculation

From interference analysis, a set of intersecting points from the point cloud are determined. Furthermore, the interference depth and direction of each point is known from the analysis. In this chapter, we describe the technique of consolidating the information for each point interference into one composite force/torque response. Section 5.1 addresses how we resolve a force response direction from the surface normal of the point cloud point with the surface normal determined from the auxiliary equation associated with the implicit equation BSP-tree. Section 5.2 explains how several point force responses are combined into a single force/torque.

5.1 Point Force Resolution

After a point has been identified as being in the interior of the solid and the appropriate auxiliary functions have been evaluated, the state of collision between that specific point and the implicit equation object is defined by the location of the point, the direction of the point outward normal vector, the depth value determined by the auxiliary function, and the direction defined by the vector auxiliary function. These factors must be combined to determine a force response appropriate for that point.

Two quantities are defined for the particular point. A force magnitude and a force direction. For convenience we will define the force acting upon the point cloud defined object. The force acting upon the implicit equation defined object would be equal in magnitude, but opposite in direction.
The force magnitude scales with the penetration depth. The magnitude must be defined so that the force penalty for deeper interference is greater than that for more shallow interference. Typically, the force magnitude is proportional to the depth of the interference. In this way, when the penetration depth is near zero, the force magnitude is small. As the penetration depth increases, the force magnitude increases. Other relationships may be used as well. The force magnitude should increase with increasing penetration depth; therefore magnitude functions such as \( \text{magnitude} = kd \) and \( \text{magnitude} = kd^n \), where \( d \) is the penetration depth and \( k \) is a constant, are some examples of possible force response magnitude functions.

Alternatively, the force magnitude can be incorporated directly into the auxiliary function for each region. Therefore, a newly defined magnitude auxiliary function could be used in lieu of the depth auxiliary function.

In addition to the force magnitude, the direction of force acting at the point in question must also be resolved. From interpenetration analysis, two direction values are determined -- the surface normal of the point in the point cloud and the direction of shortest distance to the surface of the implicit equation defined object. Ideally, the direction of one is the exact opposite of the other. In other words, consistency occurs when the outward normal of object A is anti-parallel to the outward normal of object B. In this case, the resolution of direction is simple. Since we are determining the force response acting on the implicit equation defined object, the direction of force would be the outward normal of the point cloud point.

Unfortunately, the two directions are not always anti-parallel. To some degree of accuracy, the method of strictly using the outward normal of the point cloud is adequate.
However, to utilize the most geometric information available, some composite of the information should be used to determine the direction of force. A simple method to incorporate both pieces of information is to let the direction of force be the average of the two directions. Again, since we are defining the force response upon the implicit equation defined object:

\[ v_{df} = \frac{1}{2}(\hat{n}_{pc} - \hat{n}_{eq}) \]  

(5.1)

The direction of force, as defined in equation 5.1, is not necessarily unit length. While the composite direction could be made unit by dividing by its length, the range of lengths is beneficial to the determination of an accurate representation of a force response. Noting that the full force response is the product of the force magnitude and the force direction, cases where the direction of force is less than one correspond to situations where the directions are in disagreement. When the directions agree or nearly agree, the direction of force will be close to unity. Additionally, dividing by the resultant length may not be counterproductive since this practice may lead to division by a small number, causing a calculated direction that is arbitrary and subject to numerical inaccuracies. We choose not
to normalize the force direction length and take advantage of the attenuation to the force magnitude when the directions are not consistent. Furthermore, we can drop the factor of $\frac{1}{2}$ in equation 5.1 and address the scaling factor by the depth auxiliary function.

5.2 Composite Force/Torque Determination

After an appropriate force response has been determined for each point participating in the interference, the effect of each force must be collated into one force that represents the effect of all the point forces. To achieve this result, the list of forces are transformed into their equivalent force and torque acting on a single point of the body. For convenience we select the desired compliance center of the object.

5.2.1 Compliance Center

The compliance center for a particular body is the point in space such that any translational force acting on the body that acts through this point results in pure translation (no rotation). Furthermore, any rotational force (torque) acting on the compliance center results in rotation only [Asada 86]. The compliance center is a natural choice for the location of resolving the many forces into one force/torque. We know that the force component acting on the compliance center will result only in translation of the object, and the torque acting on the body will be the only source of rotation. Furthermore, for many systems for which interpenetration analysis is applicable, the convention of returning force and torque at the compliance center will be most useful.
5.2.2 Force at Compliance Center

To determine the equivalent force and torque of a particular point's force at the compliance center, the force magnitude, direction and point location and compliance center location contribute to the final result. In calculating the force and torque at the compliance center, the methodology must be careful to assure that all vectors and point coordinates are in consistent coordinate frames. The following formulae assume that all values have been converted to world space coordinates.

Given the point coordinates \((p_x, p_y, p_z)\), compliance center coordinates \((cc_x, cc_y, cc_z)\), force direction \(F = [f_x, f_y, f_z]\), and force magnitude \(f_m\). The equivalent of a force \(f_m \cdot F\) acting upon point \(p\) is a force \(F_{cc}\) and torque \(T_{cc}\) acting upon the compliance center point \(cc\), where \(F_{cc}\) and \(T_{cc}\) are defined to be:

\[
F_{cc} = [f_{m_x}, f_{m_y}, f_{m_z}] = f_m \cdot [f_x, f_y, f_z] \quad (5.2)
\]

\[
T_{cc} = [r_yf_mf_z - r_zf_mf_y, r_zf_mf_x - r_xf_mf_z, r_xf_mf_y - r_yf_mf_x] = f_m(R \times F) \quad (5.3)
\]

Equation 5.2 shows that the translational force is always equal to the force acting upon the point. Equation 5.3 show that the rotational force (torque) is proportional to the force
magnitude and the distance between the point of action and the compliance center. When the distance is zero, the torque $T_{cc}$ is also zero.

The composite force and torque for all points is simply the summation of each of these calculated force and torques.

$$F_{cc_{total}} = \sum F_{cc_i} \quad (5.4)$$
$$T_{cc_{total}} = \sum T_{cc_i} \quad (5.5)$$

In our model, a single point of the point cloud exerts only a translational force upon the object at the location of the point. This translational force may resolve to be the combination of the translational force and a torque at the compliance center, but at the point of contact, only a translational force is modeled. These force interactions represent the reaction forces between the two objects when they are in contact. This force can be used for physical simulations where the bodies interact in space. Alternatively, these forces can be implemented with a haptic device to create a haptic system.
Chapter 6: Tool Path Generation

The interference analysis and force response determination methods described in Chapters 4 and 5, can be applied to a variety of haptic systems. In this chapter, we present a method for creating a system that allows a user to model the contact interactions between a milling tool and any solid object. The solid object represents a target part that the user wishes to create in physical reality through a milling machining process. By simulating contact forces between a computer model of the milling tool and the target part, the user will be able to feel the surface of the target part through the milling tool. During this simulation, data that describes the geometric relationship between a milling tool and a target part is collected then used to create instructions for machining the desired part.

6.1 Computer Aided Design

In industry, the need for a certain part geometry rises, and a designer creates the needed part using a Computer Aided Design (CAD) system. At this point, the part is fully defined geometrically, but it does not exist in physical reality; the part only exists as a computer representation. To create an actual physical object, Computer Aided Manufacturing (CAM) tools are applied to the CAD model to create a set of instructions that allow a milling machine to cut the part from stock material. Unfortunately, the creation of machining instructions can be complex and error prone. In rapid prototyping, where designers desire to have an actual physical part of their new designs as quickly as possible, the complications with CAM creation of machining instructions are especially problematic.
The process of milling simply removes the unwanted material from a stock, to leave the desired part. The difficulty for milling comes from the need to know how the milling tool can remove a particular volume of material without removing material that should be left alone (gouging) and not having non-cutting portions of the machine hit the stock or its fixture (collision).

At the root of the problem is the question, how can the cutting tool access the each portion of the target part surface such that the cutting surface of the tool contacts the surface of the part without any portion of the tool violating the space of the part. Hypothetically if the part physically existed, then a person could take an un-powered milling tool and press it against the part to see what orientations of the tool will be able to reach the part surface without interference with the solid. Non-interference is enforced by the physical presence of the part body. Of course, this hypothetical situation is not an option since the part does not physically exist. The part only exists as a computer representation.

By using a haptic system that implements the interference analysis method, the physical interaction between the geometry of a milling tool and the geometry of the target part can be achieved in a virtual space. In our implementation, the target part is represented by a point cloud. The implicit equation BSP-tree represents the milling tool. We choose the implicit equation representation for the milling tool for two reasons. First, milling tools tends to have a more simple geometry and therefore have implicit representations that are easier to create. And second, a milling tool is most likely to be used many times with different target parts. As a result, the investment of time and effort
to create the implicit representation of a milling tool may be used many times with different simulation environments or different target part solid objects.

In this mode of use, a haptic device is in the position to gather data that describes the possible positions and orientations of the cutting tool with respect to the target part that will contribute to the machining of that part. In this haptic system for tool path generation, the tool object (implicit equation BSP-tree) position is controlled by the haptic device, and forces acting upon the tool as a result of contact with the part are transmitted to the user through the haptic device. The part object (point cloud) is fixed in space as the tool object moves with respect to it.

6.2 Collision Avoidance Based Tool Path Generation

The haptic system functions as an instrument for enforcing collision avoidance between the two objects and recording information regarding their contact situations. To record the contact position information in an organized and meaningful manner, a composite definition, described in section 2.3.2, is used to represent the target part object. Within this composite definition, the part has a point cloud representation and triangulated surface representation. The point cloud definition is used for interference analysis and force response calculation. The triangulated model is used to organize the position and orientation information. Each point of the point cloud references the triangle to which it belongs by means of a unique triangle identification number. Therefore, when contact is made and information is to be recorded, the information is stored by triangle id number. To encapsulate all the data involved in describing the geometric information needed, we group the information into postures that contain information about the contact. A
collection of postures for each of the triangles of the triangulated representation constitute a posture map for the object.

The system has two goals that it attempts to achieve through the haptic environment. At first glance, these goals seem to have conflicting interests. However, the haptic system provides an environment that assists in meeting both goals simultaneously. During the investigation, we desire to avoid collision between the tool and the part. More accurately, we wish to avoid any situation in which gouging or physical interference occurs. At the same time we are interested in moments when the cutting surface of the tool touches the part surface. In other words, we are interested in collisions. While this goal may seem in conflict with our goal to avoid collision, what we are actually interested in is a certain type of collision. We must be able to distinguish between desirable collisions which give useful information towards creating a tool path, and undesirable collisions that represent a state of gouging or hardware interference. To resolve these conflicting view of a collision, we introduce the concept of a niceness factor to describe a particular collision’s suitability towards yielding useful information for tool path generation.

6.2.1 Posture Map

At any moment in time within the simulation, the tool object has some position and orientation relative to the part object. This position and orientation information is the minimum data required to constitute a posture. Additional information helps describe the relationship between the tool object and part object. Since a posture and a posture map are generally associated with the part object, recording the unique tool identification number in each posture is helpful. Also, the niceness factor, which describes the quality of the contact between two object can also be recorded for easier reference.
A posture map is a collection of postures that describes the accessibility of the part object. The utility of a posture map depends on postures for which the quality of contact is designated as acceptable. In other words, postures with low niceness values are not useful in the posture map. A full posture map includes at least one acceptable posture for each triangle of the triangulated surface definition of the part object.

In the case where multiple tools are used in the creation of a posture map, then the each posture within the map would include the tool identification along with the position and orientation information.

6.2.2 Niceness

Niceness is a measurement of the acceptability and desirability of a particular contact. If contact occurs such that a non-cutting surface of the tool contacts the part, then the contact is not acceptable and the niceness value is very low. If certain regions of the cutting surface are more desirable than others, then higher niceness values are associated with these more desirable surfaces. Furthermore, very deep penetration into a cutting surface is also less desirable, therefore, niceness decreases with higher penetration depth.

Niceness values are defined and determined in the same method as penetration depth (or force magnitude) is defined. Various functions are associated with each region of the tool to define niceness. If the tool surface of the region is a non-cutting surface, then the associated function for niceness returns zero. If the tool surface of the region is a cutting surface then the associated function returns a positive value. The positive value decreases for increasing penetration depth. Multiple point interpenetration is resolved by taking the minimum of all individual niceness values.
6.2.3 Data Collection

To facilitate data collection and data organization, a composite representation of the part object is used that combines the point cloud representation with a triangulated surface model. During each cycle, the tool object may be in a position that is desirable for machining the part object. The niceness value of the current collision situation is used to determine the quality of the contact for machining purposes. For data collection, the position and orientation of the tool with respect to the part is recorded as a posture. To assure that the entire surface of the tool is adequately mapped with a collection of postures, the data collection process attempts to assign one, or at least one, posture to each triangle of the triangulated surface representation of the part object. When each triangle has been assigned at least one posture, the data collection is complete.
Chapter 7: Implementation and Results

7.1 System Overview

The interference analysis method has been implemented with a Suzuki Haptic device on a Silicon Graphics Inc. OCTANE workstation. The program utilizes the dual 250MHz MIPS R10000 processors by allocating one processor to handle operating system needs and graphic display while the other processor handles interference analysis and force response calculations. Figure 7.1 illustrates the components of the haptic system and their interrelationship.

![Schematic Diagram of System Implementation](image)

The user manipulates the handle of the haptic device, shown in Figure 7.2, which can move in all directions. Joint encoders on the haptic device measure the movements and can sense motion in all three translational directions as well as two rotational directions.
The rotational axis coinciding with the axis of the handle is not measured. This data constitutes the position and orientation of the implicit equation BSP-tree defined object in the simulation. The data is conveyed to the Silicon Graphics OCTANE computer via PIO shared memory cards in the haptic device computer and OCTANE computer.

![Suzuki Haptic Device](image)

**Figure 7.2: Suzuki Haptic Device**

Based on the position and orientation information from the haptic device, the program running on processor 2 of the OCTANE performs interpenetration analysis between the two objects whose data structures are stored in the main memory of the OCTANE. The process also relays the position information to the graphics process running on processor 1. Based on the interpenetration analysis, the process on processor 2formulates the force response and sends the force response to the haptic device via the PIO shared memory cards. The computer of the haptic device reads the force response and calculates the appropriate motor torque needed to create the desired force response. These signals are sent to the power amplifier which powers the motors at each joint of the haptic robot. Processor 1 of the OCTANE reads the position and orientation from the SGI main memory and uses that information to update the graphic display.
7.2 Implicit Equation BSP-tree Example

Figure 7.3: Tool Profile Example

Figure 7.3 illustrates an axially symmetric ball end mill. The section where $z$ is between 0 and 11 is the housing for the mill; where $z$ is between 11 and 15 is the spindle; and 15 to 20 is the milling tool. Since the tool is axially symmetric, a profile of the tool is sufficient information to describe the tool geometry. The dotted lines on the tool profile divide the tool interior into regions. These regions separate the interior based on which feature of the surface each point is closest. For example, points in region I are closest to surface $\alpha$, region II is closest to surface $\beta$, and region IV is closest to point $\delta$. Auxiliary functions describing the force response and niceness are assigned to each region. For this example the force response magnitude auxiliary functions are:

\[
\begin{align*}
I. & \quad z \\
II. & \quad 5 - d \\
III. & \quad 11 - z \\
IV. & \quad 14 - d - z \\
V. & \quad 3 - d \\
VI. & \quad 16 - d - z \\
VII. & \quad 17 - 2d - z \\
VIII. & \quad 1 - d \\
IX. & \quad 1 - \sqrt{d^2 + (z - 19)^2}
\end{align*}
\]
The explicit expressions in equation 7.1 indicate the penetration depth of a point that penetrates the tool object. Each expression is only valid for the region to which it is assigned. These expressions along with the region definitions, encapsulate the fact that the closest surface to a penetrating point depends on where the point resides within the tool body. Similarly, the force response directions are dependent on region location:

I. \[ \begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \]

II. \[ \begin{bmatrix} \frac{x}{d} & \frac{y}{d} & 0 \end{bmatrix} \]

III. \[ \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \]

IV. \[ \begin{bmatrix} \frac{x}{\sqrt{2}d} & \frac{y}{\sqrt{2}d} & \frac{1}{\sqrt{2}} \end{bmatrix} \]

V. \[ \begin{bmatrix} \frac{x}{d} & \frac{y}{d} & 0 \end{bmatrix} \]

VI. \[ \begin{bmatrix} \frac{x}{\sqrt{2}d} & \frac{y}{\sqrt{2}d} & \frac{1}{\sqrt{2}} \end{bmatrix} \]

VII. \[ \begin{bmatrix} \frac{2x}{\sqrt{3}d} & \frac{2y}{\sqrt{3}d} & \frac{1}{\sqrt{3}} \end{bmatrix} \]

VIII. \[ \begin{bmatrix} \frac{x}{d} & \frac{y}{d} & 0 \end{bmatrix} \]

IX. \[ \begin{bmatrix} \frac{x}{\sqrt{d^2 + (z-19)^2}} & \frac{y}{\sqrt{d^2 + (z-19)^2}} & \frac{z}{\sqrt{d^2 + (z-19)^2}} \end{bmatrix} \]

The vector expressions in equation 7.2 describe the direction towards the closest surface. In determining the force magnitude, which is equal to penetration depth in this case, the expressions depend only on \( d \) and \( z \), where \( d = \sqrt{x^2 + y^2} \). However, because contact is not axially symmetric, determining force direction requires explicit use of \( x, y, z \) coordinates. The direction, combined with the magnitude from equation 7.1, yield the point force response.
The niceness functions for each region are:

\[
\begin{align*}
\text{I.} & \quad 0 \\
\text{II.} & \quad 0 \\
\text{III.} & \quad 0 \\
\text{IV.} & \quad 0 \\
\text{V.} & \quad 0 \\
\text{VI.} & \quad 0 \\
\text{VII.} & \quad 0 \\
\text{VIII.} & \quad 2(x^2 + y^2) \\
\text{IX.} & \quad 2(x^2 + y^2) + (z - 19)^2
\end{align*}
\] 

The niceness value is used to distinguish between valid contacts and invalid contacts. The niceness value can also be used to distinguish between multiple valid contact postures that contact the same area of the part. In the event of multiple valid contacts in the same part area, the system can either record only the posture with better niceness value, or record both postures. In the preferred embodiment, the former is used.

In the example, the regions where cutting surfaces exist are regions VIII and IX. Since all other regions are non-cutting surfaces, their niceness functions return 0 always. The niceness function for region VIII returns 2 at the surface of the tool and zero at the tool axis. Region IX has a slightly more complicated niceness function. Since it is less desirable to cut with the center of the tip of the tool (the velocity of the tip center is zero), higher niceness value is given to portions of the region that are further from the tool axis. In region IX, surface point \((0,1,19)\) has niceness of 2 while surface point \((0,0,20)\) has niceness of 1. As with region VIII, niceness decreases as penetration depth increases.

Inspection of the auxiliary functions shows that the dotted lines in Figure 7.3 are also locations of discontinuity in the force response. In other words, a point moving from region I to region II will have a sudden change in its point force response as it crosses the
boundary. In practice, this discontinuity is smoothed by the averaging effect resulting for many points of the point cloud being in a state of collision. However, the definition of the tool based on Figure 7.3 is not the only possible definition for the given geometry. A more complex regionalizing of the tool profile is illustrated in Figure 7.4. Along the surface of the tool the force direction does not have any sudden jump in direction. However, some portions of the tool still have discontinuities and the locations designated by a circle in Figure 7.4 represent singularities in the definition of direction. These singularities represent areas where the force direction definition changes very fast in a short distance. Such rapid change in direction is not desirable in the haptic system. Fortunately, the most important portion of the tool definition is the surface, which is continuous and singularity free. Of the two definitions Figures 7.3 and 7.4, the former does not have as many singularities as the latter, but has discontinuities, even at the object surface. Either definition will work, and depending on the application, one may yield better performance than the other.
7.3 Coordinate Systems

One of the most significant drawbacks of the $k$-DOP bounding volume is its inability to support rotations of the object that it bounds. In order to deal with rotations, the $k$-DOP must be re-calculated when a rotation occurs. Computing a new $k$-DOP from the base parameters is often too computationally expensive. Held et al suggest that the rotated $k$-DOP be computed from the vertices of the base $k$-DOP [Held 96]. The result is a $k$-DOP that bounds the volume as desired but is not the optimal $k$-DOP bounding volume for the given entities.

To make the best of an undesirable situation, during the process of traversing the bounding volume hierarchy, the point cloud object coordinate system is considered static. In other words, all motion is considered to be the implicit equation BSP-tree object moving with respect to the point cloud. Since the point cloud generally has a larger bounding volume hierarchy, this protocol saves some computation time by re-computing the bounding volumes for the implicit equation object only.

In the portion of the interference analysis where the points of the point cloud are compared to the implicit equations, the relative motion is viewed as the point cloud moving with respect to the implicit equation defined object. This reversal of view accommodates the fact that rotating point coordinates is far easier than attempting to rotate equation definitions. Therefore, bounding volume tree traversal is accomplished in the point cloud coordinate system while point to equation evaluation is performed in the implicit equation BSP-tree coordinate system.
7.4 Results

The haptic system was run several times with the tool object outlined in section 7.2 and illustrated in figure 7.5. The part object is also illustrated in the figure. The number of points in the point cloud were varied for each run of the system to compare the effect of point cloud density. Two trials were run for each point cloud density. In the first, the tool traces a path around the outside surface of the part object. In the second, the path traced is on the inside of the part object. The minimum feature size of the tool object is a radius of one unit. The part object has an inner radius of three units and a surface area of 759.8 square units.

The system achieves the desired effect of providing an appropriate force response to the user when contact between the two objects is made. The quality of the feel varies during the simulation based on the contact situation. When contact is made that does not result in very deep penetration, the response is generally good and convincing. Occasionally, the quality of the response degrades and the haptic device exhibits unwanted vibrations. When the tool interferes with the part object, the force response generated is appropriate for the contact situation. If the penetration depth is extremely large so that
over thirty percent of the tool object is in a state of collision, the response will not necessarily be smooth; however at such a large interference, the concept of appropriate force response breaks down. Overall, the effect is convincing, especially for the desired task of determining contact situations for CNC tool path generation where the concern is mostly focused on low penetration depth contacts.

Table 7.1 reports the quantitative data for the haptic system. Eight trials were run with 10000 pieces of data collected for each trial. The results show that for the specified tool object and a point cloud density of less than 13 points per unit area, the interference analysis and force response calculations run in less than one millisecond in the average case. These cycle times are appropriate for a haptic system. The results also show that the number of points that are in a state of interference tend to be less than 15. From these numbers we infer, that the methodology is able to achieve the necessary cycle time when the number of colliding points does not exceed 15-25 points at a time.

<table>
<thead>
<tr>
<th></th>
<th>Outside Path</th>
<th>Inside Path</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Colliding Points : Collision Time</td>
<td>Colliding Points : Collision Time</td>
</tr>
<tr>
<td>1000 points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.31 pts/area</td>
<td>Average - 1.86 : 453.7 μsec</td>
<td>Average - 3.981 : 466.3 μsec</td>
</tr>
<tr>
<td></td>
<td>Minimum - 1 : 418 μsec</td>
<td>Minimum - 1 : 387 μsec</td>
</tr>
<tr>
<td></td>
<td>Maximum - 7 : 637 μsec</td>
<td>Maximum - 8 : 683 μsec</td>
</tr>
<tr>
<td>5000 points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.58 pts/area</td>
<td>Average - 4.50 : 544.4 μsec</td>
<td>Average - 4.32 : 618.4 μsec</td>
</tr>
<tr>
<td></td>
<td>Minimum - 1 : 467 μsec</td>
<td>Minimum - 1 : 508 μsec</td>
</tr>
<tr>
<td></td>
<td>Maximum - 15 : 1066 μsec</td>
<td>Maximum - 14 : 860 μsec</td>
</tr>
<tr>
<td>10000 points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.2 pts/area</td>
<td>Average - 7.83 : 694.674 μsec</td>
<td>Average - 13.42 : 887.3 μsec</td>
</tr>
<tr>
<td></td>
<td>Minimum - 1 : 567 μsec</td>
<td>Minimum - 1 : 627 μsec</td>
</tr>
<tr>
<td></td>
<td>Maximum - 25 : 973 μsec</td>
<td>Maximum - 41 : 1243 μsec</td>
</tr>
<tr>
<td>20000 points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.3 pts/area</td>
<td>Average - 11.8 : 954.4 μsec</td>
<td>Average - 23.5 : 1382.3 μsec</td>
</tr>
<tr>
<td></td>
<td>Minimum - 1 : 674 μsec</td>
<td>Minimum - 3 : 931 μsec</td>
</tr>
<tr>
<td></td>
<td>Maximum - 88 : 1468 μsec</td>
<td>Maximum - 65 : 2150 μsec</td>
</tr>
</tbody>
</table>

Table 7.1: Interference Analysis and Force Response Calculation Times
7.5 Influential Factors

The quality of the haptic system depends on the quality of the force response signals sent to the haptic device. As stated before, rapid update (1000 Hz) of the force response is required to achieve a smooth and convincing haptic effect. As the update rate decreases, the quality of the system degrades. Several factors affect the rate at which the software is able to update the force response signal. Some of the factors are internal to the program while other factors include the characteristics of entities involved in the haptic system, which are external to the program structure.

7.5.1 Internal Factors

Internal factors include parameters used in establishing the bounding volume hierarchy for the point cloud representation. An obvious parameter includes the type of bounding volume used in the hierarchical tree; other more subtle factors affect the computation time as well such as the number of points that constitute a leaf in the bounding volume hierarchy, the method used to divide a parent into two children, and the type of tree are examples.

The maximum number of points that can constitute a leaf in the bounding volume hierarchy should be set so that the computation time for an additional bounding volume overlap test is comparable to the computation time for checking each individual point against the BSP-tree. In other words, the optimal number of points in a leaf is related to the average time required to analyze each point and the average time to perform a bounding volume overlap test, which also depends on the type of bounding volume used in the hierarchy.
The bounding volume hierarchical tree is built by grouping points of the point cloud within bounding volumes. How these groupings are determined can affect the efficacy of the bounding volume tree structure. Klosowski et al. discuss various methods of creating bounding volume trees for triangulated collision detection methods [Klosowski 97]. These techniques may also affect the efficiency of the bounding volume for a point cloud. Specifically, they consider factors such as choice of axis and choice of split point.

An additional internal parameter that affects the computation time of a interference analysis query is the type of tree used in the bounding volume hierarchy. In the implementation described in chapter 3, the bounding volume hierarchy was a binary tree, where each parent had two children. The hierarchical trees can also be set up so that parents have three or more children. This added parameter which describes the branching of the tree alters the depth of the tree in exchange for changing the width. Trees with higher branching will have smaller depth, but larger width. Changing the branching of the bounding volume hierarchy can change the performance of the interference analysis. At one extreme, the tree has very high branching and no depth, which equates to checking each point of the point cloud if the root is in a state of overlap. At the other extreme, the tree has no branching and consists of a single root node. Again, overlap of the root node leads to checking every point of the point cloud. Somewhere between no branching and high branching is an optimal branching amount, which may or may not be two children per parent. In the implementation presented in this chapter, the branching was set to two so that all trees in the analysis are binary trees. Changing the branching of the point cloud bounding volume hierarchies may improve the interference analysis time.
Klosowski et al discuss the degree of trees and conclude that binary trees are a good choice since they are simpler to compute and have some evidence that binary tree are better than trees with greater branching [Klosowski 97]. In the case of implicit equation space partition trees, the parent always has two children because an implicit equation always has two halves, which is naturally compatible with a binary tree.

7.5.2 External Factors

In addition to the internal factors, the features of the interacting entities also affect the computation time for interpenetration analysis and force response calculation. Clearly, larger and more complex solid entities participating in the simulation will require more computation than smaller and more simple solids. These external attributes of the data members that influence computation time are discussed in this section.

Physical dimension does not have a direct effect on the computation time. The main components that determine the computation time, include the number of points of the point cloud that are in a state of collision or near collision. For example, a very complex object with thousands of points in its point cloud will evaluate just as quickly as an object with only three points in its point cloud, if all points are clearly not in a state of collision. This result is due to the bounding volume hierarchy. Despite the disparity in number of points in the point clouds, the analysis is completed with a single bounding volume overlap test for both cases. Alternatively, if the model with more points was in a state of collision where several hundred of its points are in a state of interference, we would expect the larger model to have a longer computation time.

Observation of the actual procedure illustrates why number of colliding points is the most important external factor to computation time. From the bounding volume hierarchy,
points that are not in a state of collision are quickly ruled out the analysis, leaving points that are colliding or near colliding. The total number of points in the point cloud does have an effect on the computation time. Larger number of points will lead to deeper bounding volume trees, however, this effect is not as substantial compared to the computation time for each colliding point.

Since the number of colliding points is variable, a good metric for quantifying an external factor on computation time is the point density (number of points per unit area). Entities with dense point clouds are likely to have more points in a state of collision than the same entity with a more sparse point cloud representation.
Chapter 8: Conclusion

As described in Chapter 7, the interference analysis methodology with force response is capable of driving a haptic system to enable force feedback in virtual reality environments. In context with other haptic systems past, present, and future, this method distinguishes itself from previous haptic systems by its ability to haptically render a fully three dimensional object against another fully three dimensional object. How this method will stand against other procedures and future procedures will depend on the application and overall effect desired.

8.1 Implicit Equation Representation

The selection of an implicit equation binary space partition tree representation for the probe object restricts the use of the methodology to systems in which the probing tool is well known in advance. Creation of an implicit equation model for a given solid requires time and manual effort to design an appropriate representation. In many applications the probing tool that is under the user’s control is well known prior to running the simulation. In applications such as the CNC machining tool path generation scheme, not only are the tools known in advance, but the tools are used again and again. In this situation, all the tools used on a particular CNC machine need to be modeled once, then these models can be used whenever needed.

However, this interference analysis technique is not as advantageous to use in applications where the probing object is not well known in advance, changes often, or
does not get used many times. In these situations, the time and effort required to create the model may not be worthwhile.

8.2 Point Cloud Representation

While the implicit equation model is not easily changed, the point cloud model is appropriately flexible to allow for a wide variety of virtual environments. Therefore, given an environment, a point cloud model can be created to represent it easily and automatically. As long as the tool is known in advance, the system can simulate the situation where a known body explores an unknown (arbitrary) environment.

As presented, the point cloud representation, must be static. In other words, a point cloud is flexible in its ability to represent any arbitrary object, but is inflexible in its ability to change definition on-line. The rigidity of the point cloud model relates directly to the bounding volume hierarchy that organizes the points of the point cloud. Creation of the bounding volume hierarchy requires sufficiently long time to prevent such calculation between frames of the haptic rendering. Some modification of the point cloud may be possible as long as the underlying bounding volume hierarchy does not change. However, such changes are restricted to be small and will alter the efficiency of computing interference. Essentially, the interference analysis method is best suited for point cloud environments that remain static for the duration of the simulation.

8.3 Bounding Volume Type Selection

Several other measures may be taken to increase the performance of the haptic system. A fundamental aspect of the interference analysis method is the choice of bounding volume
type. In the implementation, $k$-discrete oriented polytopes were selected based on the relative ease of programming as well as the flexibility in choosing $k$. However, the more intricate oriented bounding box approach may yield better results depending on the application. The $k$-DOP bounding volume employs a faster bounding volume overlap test than the OBB; however, the $k$-DOP suffers the need for redefinition under rotational transformations. The OBB method is well suited for arbitrary rotations, but has a more complex overlap test. The best choice for bounding volume has yet to be determined and is likely to be a matter of specific application rather than clear superiority.
References

Adachi 95

Asada 86

Basdogan 97

Cohen 94

Cohen 95

Ganter 93

Gottschalk 96

Held 96

Hollerbach 97

Hou 96
Klosowski 97

Latimer 97

Lin 98

Lin 93

Massie 96

McNeely 99

Mirtich 98

Mulmuley 94

Ponamgi 97

Preparata 85

Laxmiprasad 98

Salisbury 97
Storti 92

Swarup 95

Thompson 97