2010

A hybrid method for haptic feedback to support manual virtual product assembly

Daniela Faas
Iowa State University

A hybrid method for haptic feedback to support manual virtual product assembly

by

Daniela Regine Ursula Faas

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Co-majors: Mechanical Engineering; Human-Computer Interaction

Program of Study Committee:
Judy M. Vance, Major Professor
James E. Bernard
Stephen Gilbert
James H. Oliver
Eliot H. Winer

Iowa State University

Ames, Iowa

2010

TABLE OF CONTENTS

LIST OF FIGURES v

LIST OF TABLES vii

ABSTRACT viii

CHAPTER 1. Introduction 1

1.1 Overview 1

1.2 Scope 2

1.3 Impact and Motivation 4

1.4 Organization 6

CHAPTER 2. Background 7

2.1 Virtual Reality Assembly Simulation 8

2.2. Collision Based Applications 11

2.3 Positional Constraint Based Applications 15

2.4 Geometric Constraint Based Applications 16

2.5 Combined Physics- and Constrained-based Applications 20

2.6 Background Summary 22

CHAPTER 3. Geometric Modeling 24

3.1 Boundary Representation 25

3.2 Geometric Modeling and Haptic Rendering 26

3.3 Voxel-sampling-based Collision Detection 28

3.4 Haptic Rendering 30

3.4.1 Haptic Devices 31

3.4.2 Dual-handed Haptics 32

CHAPTER 4. Methodology 36

4.1. Voxmap PointShell Method 36

4.1.1. Pointshell Shrinking 42

4.2. Geometric Constraint Recognition and Solver 45
4.3 Hybrid Method 48
4.3.1. Performance 50
CHAPTER 5. Voxmap Pointshell-Constraint Hybrid Method 52
5.1. Hybrid Methodology 54
5.2. Tying BREP Info to Voxels 55
5.3. Constraint Recognition 59
5.4. Blending Constraint with Collision Forces 62
CHAPTER 6. Implementation 71
6.1 Hardware 71
6.2. Haptic Devices 73
6.3 Software 73
6.4 Application Structure 76
6.4.1 Initialization 76
6.4.2 Simulation Loops 77
6.4.3 Application Flowchart 78
6.5 Application Features 80
CHAPTER 7. Pointshell Shrinking Assessment 84
7.1. Experimental setup 85
7.1.1 Software 85
7.1.2. Hardware 85
7.2. Procedure 86
7.2.1. Pointshell shrinking affect on assemble-ability study 87
7.2.2. Feature Size Affect on Assemble-ability Study 89
7.3. Results 89
7.4. Conclusion 93
CHAPTER 8. Results 95
8.1 Evaluation of Hybrid Method 96

8.2 Low-Clearance Assembly Evaluation of Hybrid Method 97
8.2.1 Case I: Voxmap PointShell Method Only 97
8.2.2 Case II: Hybrid Method 98
8.5 Refresh Rate Evaluation of Hybrid Method 100
8.4 Summary 102
CHAPTER 9. Conclusion 104
BIBLIOGRAPHY 107
ACKNLOWDEDGEMENTS 117
ACKNLOWDEDGEMENTS 117
BIOGRAPHICAL SKETCH 118

LIST OF FIGURES

Figure 1: User Interaction through Desktop VR Interface 4
Figure 2: Dual PHANToM Configuration with HIDRA [16] 13
Figure 3: VEGAS shown in a CAVE Environment. 14
Figure 4: VADE by Jayaram [28] 15
Figure 5: The Constraint Manager Interface with Industrial Case [30] 17
Figure 6: Dual-Handed Haptic Configuration for SHARP 19
Figure 7: Positioning of a Peg with Guide Planes [37]. 21
Figure 8: B-rep Geometry [40] 24
Figure 9: Geometric B-rep Entities in Peg and Hole (Faces, Edges and Vertices) 25
Figure 10: Desktop based Haptic Systems (left to right): SensAble's PHANTOM Omni, SensAble's PHANTOM Desktop, MPB-technologies Freedom 6S, Force dimension's 3-DOF Omega haptic device 31
Figure 11: Several Glove based Haptic Systems: Immersion's CyberGraspTM, Immersion's Haptic Workstation 31
Figure 12: A User with a PHANTOM Haptic Device in a CAVE [65] 33
Figure 13: (A) Point-Voxel Collision Detection, (B) Tangent Plane Force Model [70] 38
Figure 14: Virtual Coupling Model [70] 38
Figure 15: Normal and Shrunk Pointshell Location 43
Figure 16: D-Cubed Modules. 46
Figure 17: Model Flow Chart 50
Figure 18: Force and Torque Flowchart 53
Figure 19: Binding of Separate Geometry Models in the Hybrid Method 54
Figure 20: Association of BREP Face ID with Voxels 57
Figure 21: Lookup Table for B-rep Face and Edge Data 58
Figure 22: Assembly Sequence for Hybrid Method 59

Figure 23: Positioning of a Cylindrical Peg over Hole Using Coincidence of the Cylinder Axis and the Joint Axis. 60
Figure 24: Constraint Recognition Algorithm 61
Figure 25: Sample Part Insertion Where Force Blending is Critical 65
Figure 26: Peg-Hole Mate 66
Figure 27: Response Surface of Force Blending Factor 68
Figure 28: Desktop VR Setup with 2 Phantom Omnis 72
Figure 29: Application Libraries 74
Figure 30: Simulation Loops 78
Figure 31: Application Flowchart 79
Figure 32: Hardware Setup 86
Figure 33: Peg and Hole 87
Figure 34: Peg and Hole Assembly with Hybrid Method 99
Figure 35: System Performance Comparison 101

LIST OF TABLES

Table 1: Experimental Setup for Pointshell Shrinking 88
Table 2: Experimental Setup for Feature Size..... 89
Table 3: ANOVA Table for Pointshell Shrinkage 90
Table 4: Analysis of Variance 91
Table 5: Table for Effect Tests..... 91
Table 5: CAD Model Statistics 97

ABSTRACT

The purpose of this research is to develop methods to support manual virtual assembly using haptic (force) feedback in a virtual environment. The results of this research will be used in an engineering framework for assembly simulation, training, and maintenance. The key research challenge is to advance the ability of users to assemble complex, low clearance CAD parts as they exist digitally without the need to create expensive physical prototypes. The proposed method consists of a Virtual Reality (VR) system that combines voxel collision detection and boundary representation methods into a hybrid algorithm containing the necessary information for both force feedback and constraint recognition. The key to this approach will be successfully developing the data structure and logic needed to switch between collision detection and constraint recognition while maintaining a haptic refresh rate of 1000 Hz.

VR is a set of unique technologies that support human-centered computer interaction. Experience with current VR systems that simulate low clearance assembly operations with haptic feedback indicate that such systems are highly desirable tools in the evaluation of preliminary designs, as well as virtual training and maintenance processes. This work will result in a novel interface for assembly methods prototyping, and an interface that will allow intuitive interaction with parts based on a powerful combination of analytical, visual and haptic tools.

CHAPTER 1. INTRODUCTION

1.1 Overview

The goal of this research is to develop and evaluate methods to support interactive low-clearance computer-aided design (CAD) part assembly in an immersive, virtual environment. This research will form the foundation of future advances related to product design in Virtual Reality (VR), specifically in the areas of assembly planning, virtual training and maintenance planning.

One of the manufacturing engineer's major responsibilities is the determination of assembly sequences and methods. Given an assembly of parts, the manufacturing engineer constructs the assembly sequence, determines the configuration of sub-assemblies, and attempts to balance the work load of workers along the assembly line to avoid bottlenecks in assembly. Software tools currently in use are based on traditional computer interfaces such as the monitor and mouse. While these tools support some aspects of decision making, they do not realistically account for how humans interact with parts. For example, CAD software can be used to identify interference fits between mating parts, but this software does not account for variables in human decision-making with respect to part manipulation. As a result, human interaction with parts may not correspond with the assembly sequence determined through the use of traditional software tools. Without the consideration of the human interaction, errors can emerge late in the design process which

may result in the need to make costly design changes. These errors could consist of misplaced parts, hard to reach areas, the need for supplemental fixturing and other actions.

Hands-on interaction and manipulation of parts supports human decision making. Incorporating this kind of interaction during the assembly phase of the design process has the potential to identify errors before they become costly setbacks.

Haptic feedback adds the senses of touch and force to the visual components of VR. The feeling of parts as they contact and fit together is a key consideration when evaluating assembly sequencing. Haptic feedback is essential to obtaining realistic model behavior because it acts as a physical guide to part interaction. Operator experience with current VR systems shows that haptic feedback is a highly desirable tool in the evaluation of preliminary designs prior to prototype building.

This work will integrate the concepts of haptic feedback and low clearance assembly into a single, powerful tool. The hybrid methodology combines voxel collision detection and automatic constraint recognition, allowing engineers to rapidly explore assembly scenarios, evaluate design concepts, and train assembly workers. This will result in a novel solution for assembly operations and allow intuitive interaction with parts under realistic premises. Assembly scenarios will be tested as a result of this work.

1.2 Scope

The aim of this thesis is to provide a novel method for VR manual assembly simulations to allow parts with low clearances to be assembled. The voxmap pointshell is a

reliable collision detection method, but is limited by its voxel approximation in low clearance situations. Constraint methods rely on highly accurate geometry information, but cannot provide the required haptic refresh rates. Boundary representation (BREP) models contain the surface and face information of a part and can be subjected to geometric constraint solving algorithms. In this work, automatic constraint recognition based on BREP data representation will be used to provide guidance in the assembly process. Individual voxels carry BREP data for constraint recognition purposes. Collision forces and torques are scaled down in this hybrid method while constraint guided alignment forces and torques are used as guides. This will make haptic feedback possible during low clearance assembly simulations.

The VR interface allows optimal viewing capabilities through stereo viewing and position tracking. Fig. 1 illustrates one potential VR environment consisting of a projection screen, two haptic devices, stereo glasses and position tracking of the head positions. In a fully-immersive VR environment, it will be possible to view, interact with, and assemble components with the help of haptic feedback.

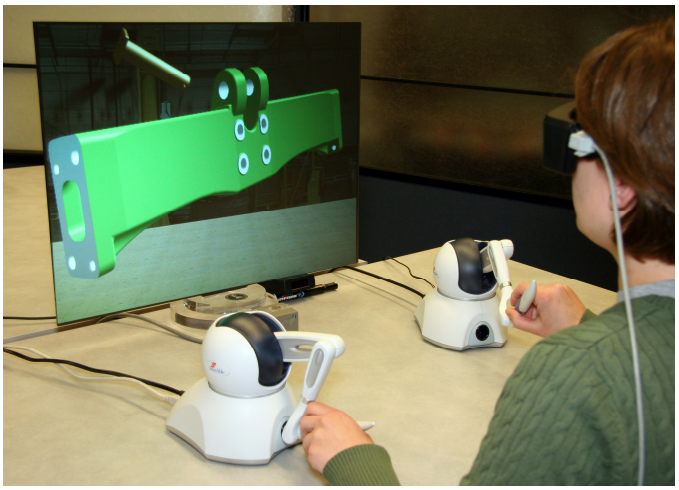

Figure 1: User Interaction through Desktop VR Interface

1.3 Impact and Motivation

Current virtual environments typically rely on visual and audio sensory feedback. However, in reality, the assembly process relies primarily on a person's ability to feel how parts go together. An immersive virtual environment including haptic feedback provides interfaces for interaction with CAD models before physical prototypes are built, thus allowing engineers to anticipate potential problems with physical assembly. Although haptic rendering and VR cannot be used to fully replace interactions on the real manufacturing floor, these methods present the potential to improve the decision making process of the user during CAD model manipulation.

If VR is to be an integral part of the early design process, advances in collision detection and haptic modeling of complex CAD parts are needed to bring realism to immersive virtual assembly. Physical assembly processes engage multiple human sensory systems. Adding sensory input to the virtual environment in the form of force feedback will

aid in the decision making process by bringing the virtual experience closer to what assembly workers will experience during production.

One method to explore human interactions with parts in an assembly is to insert virtual humans into the digital environment. Virtual humans have been used in assembly planning and ergonomic studies, but they do not allow users the first-person experience. Human motions are anticipated and then programmed into the digital environment. The first-person perspective, in combination with haptic feedback, brings a high level of realism to assembly planning. The key feature of VR environments is real-time interactivity. User input produces interactive modification of the virtual world. The combination of instant feedback and interactive use of the human senses permits VR to be an immersive, intuitive experience for users.

Current VR technology allows users to look at the model from different perspectives and move within the virtual environment. Haptic feedback adds the sense of touch to the user experience and allows more natural, intuitive interaction with 3D CAD models. Without force feedback, the user can manipulate the models, but is not able to feel collisions between parts. Because correct part-to-part interaction is the goal of the assembly design process, VR without haptic feedback is considerably less useful. Incorporating haptic feedback into virtual manual assembly is therefore a necessary next step in improving the functionality of the VR environment, and by extension its usefulness as an engineering tool during manual assembly operations.

1.4 Organization

The next chapter presents a review of virtual assembly applications, computational geometry and haptic devices. The chapter organizes previous research attempts based on part interaction in VR. Chapter 3 discusses boundary representations and different types of collision detection algorithms. In addition, haptic rendering and devices are presented. Chapter 4 presents a novel hybrid method and explains the voxmap pointshell method and geometric constraint recognition algorithms used for this research in more detail. Chapter 5 discusses how the BREP data is tied to each voxel and also discusses force blending between voxmap-pointshell and alignment forces and torques. The application structure and features are discussed in Chapter 6. Chapter 7 discusses pointshell shrinking and its effectiveness for assembly tasks. Chapter 8 evaluates the hybrid methodology and presents results. The chapter discusses the challenges involved in the hybrid method. The final chapter outlines the conclusion and identifies directions for future work.

CHAPTER 2. BACKGROUND

VR is a technology that provides users with the ability to interact with a digital environment using multiple senses. The key elements of the VR experience are the virtual world, immersion, sensory feedback and interactivity [1]. The first of these, the virtual world, is a collection of objects, with rules and relationships that control the behavior of these objects in the virtual environment. Immersion, the second component, refers to the objective level of sensory fidelity and is often achieved through multiple sensory stimulations. The most common sensory experience implemented in a virtual environment consists of stereoscopic viewing and head tracking. Position trackers are placed on the users head to track the person's change in view. New views of the digital environment are created based on the new viewpoint and are displayed either on a projection screen or in a head mounted display. This control of the digital view allows a user to view the virtual object by moving his/her head in a natural way.

One of the goals of VR is to provide immediate, interactive sensory feedback to the user. In order for VR to seem real or authentic, the environment must respond to the user's actions. The VR environment allows users to move, select and release objects. VR technology allows the user to be present in a virtual world and to be mentally immersed while the VR system senses the user's position and actions and responds to one or more of the user's senses [1]. VR supports, amongst other things, interaction with 3D objects using natural hand and head motions instead of using more traditional computer terminal inputs.

Research in virtual assemblies faces several challenges. One of the primary challenges is to provide accurate and reliable collision detection that allows parts to be manipulated and does not allow interpenetration. Final positions of the parts can be placed using cues, but the interaction between parts and the user suffers if the cues take over. These cues can be pre-defined final part positions that allow parts to snap into place or geometric constraints that guide parts into the final position by simulating physical constraints computed in real time. In addition, data transfer between CAD and VR systems is very difficult, especially managing and supporting complex CAD parts. In the following section, previous research in virtual assembly methods is outlined.

2.1 Virtual Reality Assembly Simulation

In this section, research in the area of virtual assembly simulation is reviewed. The use of VR has matured beyond simply providing a stereo view with the ability to fly through a scene. Today's VR is becoming a truly interactive 3D design environment. VR has been used in several assembly planning simulations, in which the user can interact with 3D CAD models in a virtual environment. These applications support factory planning, maintenance, assembly planning, visualization, and ergonomic assessment, among other tasks. Other researchers have produced couplings of VR with other analyses modalities, such as finite element, multi-body kinematics/dynamics, fluid flow visualization, and other areas of application [2-5].

VR provides a different way to interact with CAD models than the traditional desktop-based applications. Ye *et* al. [6] concluded that the use of virtual environments helps users perform assembly tasks. They compared a traditional non-immersive desktop virtual environment with an immersive projection screen CAVE environment to assess the user's ability to generate an assembly sequence for an air cylinder assembly consisting of 34 parts. The subjects were presented with these three conditions and were asked to generate an assembly sequence. This study showed that human subjects perform better during assembly operations in virtual environments than in traditional engineering environments.

Virtual Reality has also been used for assembly planning to reduce cost and improve assembly sequence planning. While human factors are an important consideration in engineering design, assembly planning is critical to the success of the product. Assemblies can be very complex and can involve hundreds of parts. Each new product requires an assembly sequence plan. Assembly planning has a large impact on cost and production efficiency and is a critical step in the design process. Typically, a production engineer performs assembly planning. The process includes checking for part collisions and part trappings. Reorientation, directionality, and stability must be considered [6]. A step-by-step plan is developed to tell assembly workers what parts are attached to other parts, how they are attached, and in what order. Input from CAD models is used to develop the plan and requires the assembly to be broken down into manageable subassemblies, which are then broken into individual parts.

As a result for the need to improve assembly planning and developing optimized assembly steps, some researchers have focused on the automatic generation of assembly

sequences [7, 8] by using computational geometry to automatically identify mating surfaces and interferences. However, automatic generation of assembly sequences does not keep the human in the loop as a decision maker. Manual assembly simulations allow human operators to perform the required sequences and be part of the decision making process. As Gomes notes, the final goal of any assembly simulation "is the assertion that a part or component can be assembled by a human worker, and that it can be disassembled later on for service and maintenance" [9]. The following is an overview of research that has investigated issues with assembly operations in a virtual environment.

Current methods follow several different approaches based on the level of part interaction with the environment. The first category consists of systems and applications that use physical properties to simulate the interaction of parts with the environment. These applications typically allow the user to move parts freely in the environment. Collision detection is used to prevent interpenetration of parts. The second category of applications uses constraints to place parts within the virtual environment. There are two types of constraints that are commonly used in virtual assembly simulations: geometric and positional. Geometric constraints are used to place parts precisely when physical constraints are not available. Positional constraints are used to pre-define the final position of parts. The third category of applications uses a combination of physics and geometric constraint based interactions to manipulate parts in a virtual environment.

2.2. Collision Based Applications

Kuehne and Oliver [10] created IVY (Inventor Virtual Assembly) in 1995, which allowed the user to verify and evaluate the assembly characteristics of components. The objective of the application was to aid the design-for-assembly process. The assembly steps could be animated for further examination. Although collision detection was not implemented, objects were selected in the virtual environment using assembly hierarchy information.

VSHOP was developed by Pere *et* al. [11] in 1996. It incorporated collision detection based on boundary boxes to avoid part interpenetration. VSHOP was a PC-based system that integrated a force feedback device, the Rutgers Master II. The Master II is a glove-based haptic device that provides tactile feedback to the user through pneumatic actuators. Hand gesture recognition and gravity were implemented for several tasks.

Bullinger *et* al. [12] at the Fraunhofer Institute of Industrial Engineering in Germany developed a system that created a script file containing the sequence of assembly actions. The application used a head-mounted Display (HMD) for stereo viewing as well as data gloves for gesture recognition. VirtualANTHROPOS [13], which simulates movement of the human body in a virtual environment, was used to view a virtual human during assembly tasks. Collision detection was enabled, and head and hand tracking was implemented with magnetic trackers. Only simple gestures such as reaching, grabbing, bringing, placing, and releasing were implemented. This research primarily focused on generating assembly sequences.

Physics-based applications render simulated physical responses between parts during collision detection. The key to this approach is the use of Newtonian physics to compute the motions of rigid bodies within that simulation. VEDA (Virtual Environment for Design for Assembly) by Gupta *et* al. [14, 15] relies on collision detection to model real-world physical behavior. These physics-based algorithms simulated part trajectories once collision occurred. VEDA used two Sensable PHANTOM haptic devices to interact with models. However, the assembly process was limited to two-dimensional (2D) models.

HIDRA (Haptic Integrated Dis/Re-assembly Analysis), developed by Coutee *et* al. [16, 17], also supported haptic feedback. The user was able to grab models between two fingertips, but had only limited 3D manipulation ability (Fig. 2). Because the user's fingertip was treated as a point and not as a surface, complicated geometries were difficult to handle with HIDRA. The application used polygon soup-based collision detection (SWIFT++). HIDRA switched between collision detection and constraint detection and maintenance. The constraint detection and maintenance occurred only when parts were already assembled and did not aid the user in the assembly step. This constraint algorithm reduces the interpenetration that can occur between parts; however, Coutee *et* al. did not compute any constraint forces since they were not always known. The application did not support immersive stereo.

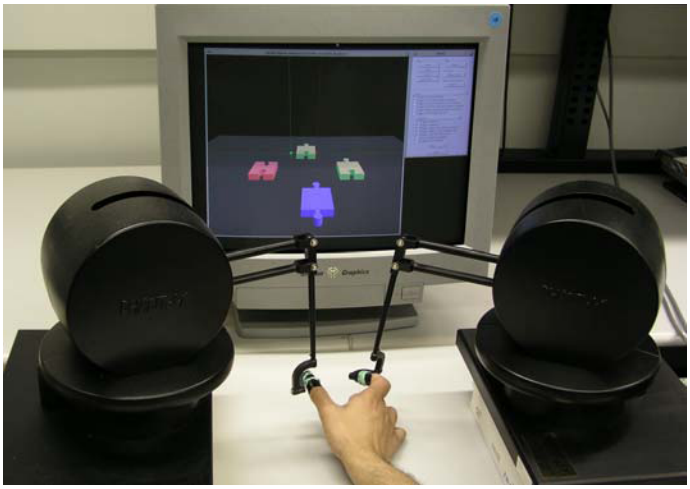

Figure 2: Dual PHANToM Configuration with HIDRA [16]

Kim and Vance [18] examined several collision packages and found the voxmap pointshell method to be best suitable for assembly operations. Virtual Environment for General Assembly (VEGAS) [19], in addition to haptic feedback, allowed data glove interaction and implemented physically-based modeling for parts using the voxmap pointshell method (Fig. 3). The application tracked the user's hand and head position in the virtual environment. The parts were manipulated using a wand in a CAVE environment. The application was able to handle full-scale models with high voxel counts. In later work, an extension to VEGAS implemented a virtual arm model for collision detection between a human operator and parts. A wireless data glove by 5DT Corporation was used to support gesture recognition. Dual-handed assembly operations were also supported.

Figure 3: VEGAS shown in a CAVE Environment.

Out of this research, NHE (Network Haptic Environment) emerged to enable assembly tasks to be evaluated by individuals in geographically distinct locations [20]. The users were able to participate in a network with each other despite being in different locations. A local PC machine was designated as the haptic computer for each environment to ensure a haptic refresh rate high enough to provide smooth interaction between the virtual environments.

Garbaya *et* al. [21] performed an experimental study where user performance was compared in real and virtual environments. A spring-damper model was used to provide collision and contact forces during the mating phase. PhysX, an open source toolkit for physics-based modeling, was incorporated to allow collision detection. Garbaya *et* al. used the CyberGrasp™ haptic device from Immersion Corporation. The experimental study concluded that user performance improved when collision forces were rendered, as compared to when only grasping forces were rendered by the system.

2.3 Positional Constraint Based Applications

Positional constraint based applications use pre-defined positional information to place parts in the virtual environment. Part assembly is accomplished by "snapping" a part to its final position when two parts come within close proximity. One of the first applications that integrated positional constraint based modeling was VADE (Virtual Assembly Design Environment), developed in 1995 by Jayaram *et* al. [22-27] (Fig. 4). Pro/Toolkit was used to import assembly data including geometric constraints and assembly hierarchy to simulate the assembly task. The geometric constraints were activated when parts were in proximity and snapping methods allowed completion of the task. Physics-based methods simulated realistic part behavior such as sliding and swinging. VADE was used in ergonomic evaluations [28]. Only later versions incorporated geometric constraints.

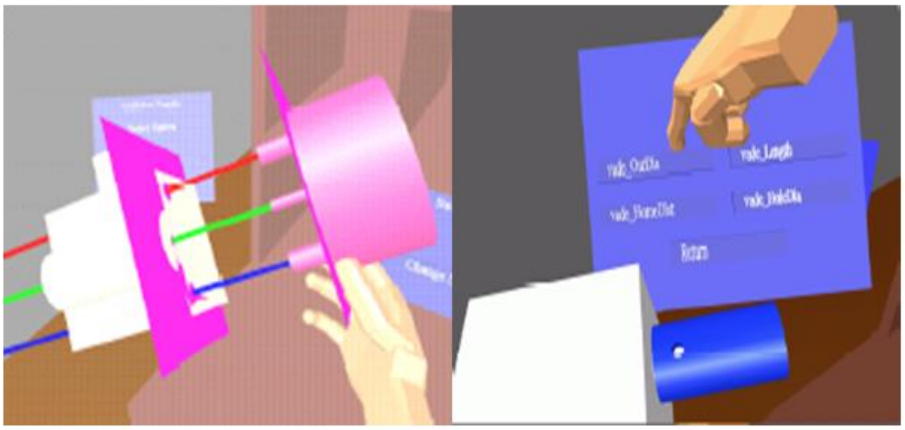

Figure 4: VADE by Jayaram [28]

2.4 Geometric Constraint Based Applications

Geometric constraint based applications render the interaction between parts in a simulated environment using geometric constraints such as parallel, concentric or coplanar.

IPSEAM (Interactive Product Simulation Environment for Assessing Assembly and Maintainability) was created by Fernando *et* al. [29] in 1999 and was one of the first applications that used D-Cubed for its geometric constraint recognition algorithm. The approach used by Fernando *et* al. investigated a generic system architecture based on geometric constraint based modeling and not on Pro/Engineer toolkits for geometric constraint import.

The automotive industry has been receptive to integrating VR tools in their production planning. For example, Gomes *et* al. developed a virtual assembly system in conjunction with BMW [9]. The researchers used an HMD for the graphics display. A user study was conducted with a glove for gesture recognition and tactile feedback. Navigation though the virtual world was done through gesture recognition. The user study concluded that the tactile feedback was unrealistic and that realistic virtual assembly operations require force feedback to increase the performance of assembly tasks.

Marcelino *et* al. developed a geometric constraint manager in 2003 to simulate assembly and disassembly tasks in VR [30]. The geometric constraint manager (CM) used direct interaction, automatic geometric constraint recognition, geometric constraint satisfaction and constrained motion (Fig. 5). The CM supported “against”, “collinear” and

"concentric" constraints and was able to validate and enforce existing geometric constraints. The CM recognized broken geometric constraints as well as new constraints. The geometric constraints restricted the object's movement and determined its kinematics. Optimization techniques to handle object transformation and geometric constraint recognition allowed the system to handle industrial cases. The system did not support haptic feedback, but was supported for multiple, immersive virtual environments.

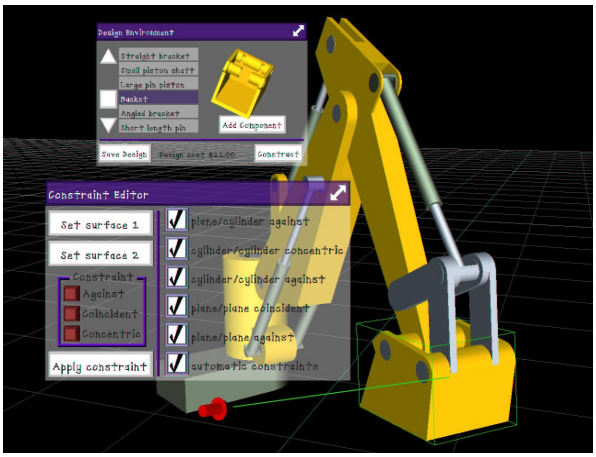

Figure 5: The Constraint Manager Interface with Industrial Case [30]

VAPP (Virtual Assembly Process Planning) was developed by Jun *et* al. in 2005. Part behavior was implemented using automatic geometric constraint recognition and collision detection. Assembly tasks were organized into hierarchical assembly tasks lists. Haptic feedback was not implemented. A discussion of the accuracy of the collision detection algorithm was not included.

Wan [31] developed MIVAS (Multi-Modal Immersive Virtual Assembly System) at Zhejiang University in 2004. MIVAS uses Pro/Toolkit to import CAD geometry and predefined geometric constraints from ProEngineer. VPS was used to perform hand-to-part

collision detection, while RAPID [32] was used for part-to-part collisions. An “escape” direction was used to prevent any unnecessary collision detection during the first steps of disassembly. The authors did not discuss any clearance issues between CAD parts. The user was able to interact with very large scenes (more than 400 objects) with MIVAS.

VECA (Virtual Environment for Collaborative Assembly) was developed by Chen *et* al.. [33] in 2005 and allowed collaborative assembly tasks to be performed at geographically dispersed locations. VECA also used Pro/Engineer to extract geometric constraint data as well as Pro/Toolkit for extracting geometry. VECA does not support haptics.

Seth *et* al. developed a feature-based approach to geometric constraint recognition by taking advantage of dynamically contacting geometric features to predict assembly intent [34]. This approach to representing realistic model behavior is based on physical and geometric constraints. BREP data was used for collision detection, physical constraint simulation and geometric constraint based modeling based on the D-Cubed module. Although this research enables very accurate collision detection and allows feature-based automatic geometric constraint recognition, it lacks haptic feedback (Fig. 6). The user was able to assemble parts with two haptic devices, but no force feedback was rendered to the user. Instead, the haptic devices were used as 6DOF position input.

Figure 6: Dual-Handed Haptic Configuration for SHARP

Iacob *et* al. [35] proposed a new method to manage collisions for contact identification. Polyhedron and kinematic constraints are generated and contact information is created. This approach allowed the use of a six DOF haptic device. The kinematic constraints are used to remove the collision detection between those contacts, therefore allowing assembly and disassembly to occur.

The problem with this approach and others that use pre-defined geometric or positional constraints is that, as the part snaps into position, there is no checking for interference with other parts. These constraint based approaches follow an assembly plan, but they do not allow the user to explore alternate ways to assemble parts. Because these approaches do not allow the user to perform any actions that do not conform to the pre-defined constraints, the potential of the VR environment is not being fully utilized.

2.5 Combined Physics- and Constrained-based Applications

Further developments for VADE allowed pre-defined constraint information to be imported and used during assembly tasks [36]. Wang *et* al. used the geometric constraint information obtained from Pro/Engineer for each part. The assembly tasks were performed using those pre-defined constraints. In addition, VADE was dependent on Pro/Engineer to generate model and constraint information, therefore reducing its compatibility with any other CAD program. This system required pre-programmed constraints and target positions. In addition, the final positions of the parts were also pre-defined. When the parts were close to the final position, they snapped into place.

Loic *et* al. developed a method that uses non-smooth contact dynamics to manipulate objects in the environment and render haptic forces [37]. Geometric constraints were used for low clearance assembly. In a preprocessing step, the geometric constraints were identified by using the same constraints as applied by a CAD program to fully constrain assembled parts. This method uses "guide planes" as visual cues. These consist of geometric planes and surfaces that are created and added to the CAD representation before import into the virtual environment. These guide planes are manually created in the pre-processing step (Fig. 7).

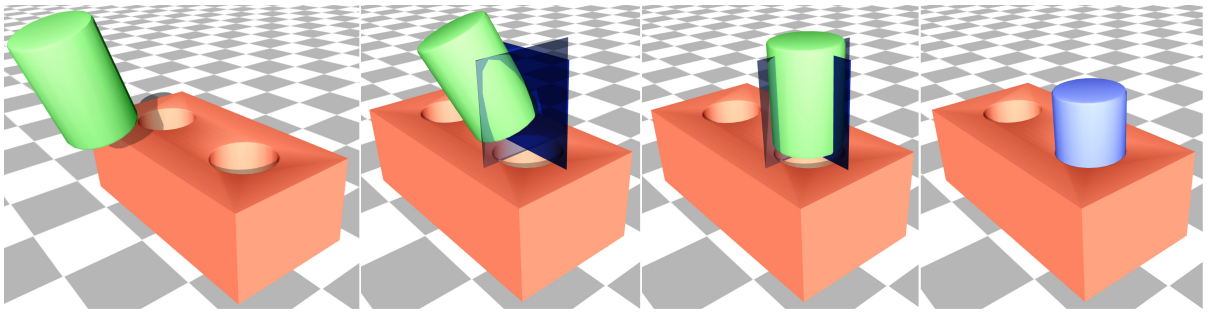

Figure 7: Positioning of a Peg with Guide Planes [37].

The guide planes appear when the moving part approaches another part where a constraint had previously been identified. The user's motion is confined to the guide plane. The geometric constraint is engaged when the moving part approaches the stationary part. Once the geometric constraint is engaged, collision detection and haptic rendering are turned off. There were several potential problems when implementing this approach. One is that turning off collision detection could potentially lead to other collisions occurring but not being detected. Also, in this method, the geometric constraint begins to exert force even before parts touch. This could result in parts being pulled into unwanted constraints. For example, if the user wanted to insert a pin into a specific hole from a line of holes, the neighboring holes would try to pull the pin in as the user moves the pin overhead.

The hybrid approach as it is proposed here, including geometric constraint recognition and haptic rendering, will allow further development of VR as a tool for Product Life Management (PLM). The geometric constraint based algorithm will guide the manual assembly process, while haptic feedback is used to let the user know when parts are colliding. The next section summarizes previous and current research efforts and outlines

some of the challenges in collision detection and haptic rendering for virtual assembly methods.

2.6 Background Summary

This section reviewed and summarized several approaches that have been implemented over the years for performing virtual assembly simulations. Most of the early assembly methods did not use collision detection and depended on pre-defined final positions of parts which allowed them to be snapped into place when they were close to their final positioning. Using geometric constraints allowed for precise placement of parts, but these constraints were typically pre-defined and reduced the opportunity for human decision making.

Pre-defined geometric constraints were activated when parts come close enough to each other and constrained motions were then used to place the parts into final positions. Geometric constraint based applications allow low clearance between mating parts because of the use of highly accurate CAD representations such as BREPs and parametric representations. The downfall of geometric constraint based algorithms is the need for special CAD toolkits to extract the wanted information. This is prohibitive to widespread use of these applications and reduces their acceptance within the user community. Advancements in constraint-guided assembly methods allowed close to real-time validation and application of geometric constraints and also did not require pre-defined constraints or require complex CAD data transfer.

There are many examples of using VR as an interface for product assembly and manufacturing purposes. Few of these systems have haptic force feedback integrated to aid in the assembly process due to the high refresh rate requirements for real-time haptic force computations. For a comprehensive review, see [38].

Several key developments brought VR assembly methods research to their current state. Physics-based assembly simulations used mass, moment of inertia, and center of mass to calculate physical responses such as collision, friction, gravity and other forces. Collision and tactile forces provided the user with an intuitive environment for assembly simulations. Haptic interfaces allowed users to touch and feel virtual models present in the virtual environment. Physics-based assembly simulations typically sacrifice collision accuracy in order to provide real-time collision checks. The high refresh rate requirements for haptic force feedback (~1000 Hz) make low clearance assemblies very difficult to achieve due to reduced collision accuracy. The demand for highly accurate collision detection, while maintaining interactivity with the environment, is critical to the success of virtual assembly methods in industry and academia.

CHAPTER 3. GEOMETRIC MODELING

Computational geometric modeling is used in a wide variety of areas to represent objects. Geometric modeling, defined by Mortensen, is “a computer-aided process” that uses “differential and analytic geometry, vector and matrix methods, tensors, topology, set theory, and an arsenal of numerical computation methods to capture the potentially complex description of an object” [39]. Geometric modeling research is directed toward topics such as realism and faster rendering algorithms. Other research areas include automatic finite element mesh generation, data management, and user interfaces.

There are several geometric representation schemes used in CAD systems. The main schemes are: wire frame representation, various surface modeling schemes, constructive solid geometry (CSG) and boundary representation solid modeling, as well as sweep representation. Based on the representation scheme, different information about the model is derived from the CAD system. BREPs are of particular interest for this research as they allow geometric constraint recognition through surface and face collisions (Fig. 8).

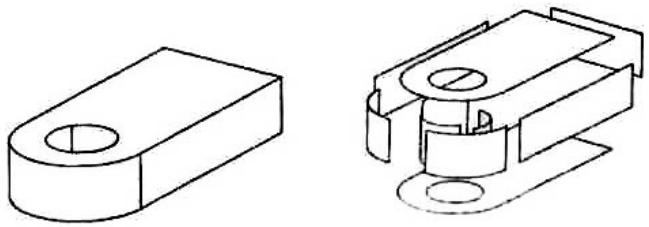

Figure 8: BREP Geometry [40]

3.1 Boundary Representation

The standard BREP geometry model consists of discrete bounding surfaces, curves and point entities (Fig. 9). The BREP shape is represented using the limits of the object. A solid can be modeled as several connected surface elements. The main topological items are faces, edges and vertices. BREP models allow highly accurate collision detection, physical constraint simulation and geometric constraint based modeling.

Consider the peg and block models below (Fig. 9). The peg has one cylindrical face and two planar faces, whereas the block has six planar faces and one cylindrical face. In addition, both models have two cylindrical edges.

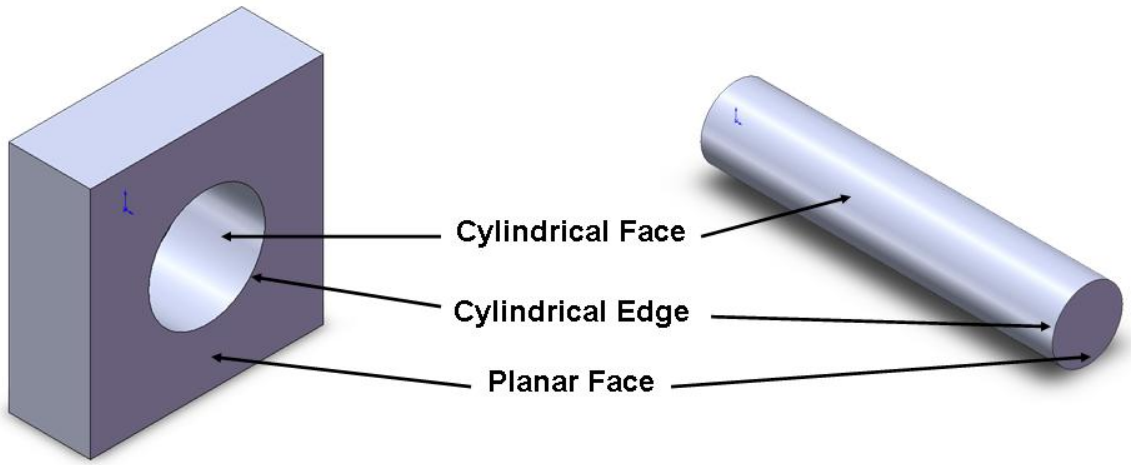

Figure 9: Geometric BREP Entities in Peg and Block (Faces, Edges and Vertices)

Geometric constraints solvers are standard in CAD packages and aid in the assembly process [39]. The advantage of BREP modeling is that the model has high spatial accuracy and facilitates motion constraints by an ad hoc kinematic constraint (virtual kinematics). Liu *et*

al. showed that a BREP model could be changed via haptic feedback by using a BREP model and a haptic geometry model [41]. The user was able to deform the surface model by pushing, pulling or dragging on its surface. This approach supported only simple models.

The advantage of BREPs is that they are widely used in CAD. Because of their data structure, BREPs are used as the geometric basis in geometric constraint solvers. The goal of the geometric constraint solver is to find all placements of the geometric entities which satisfy the given constraints. Geometric constraints are geometric entities such as distance, angle, parallel, perpendicular, concentric, and tangent. Software such as D-Cubed, a collision detection and geometric constraint solver, is needed to deal with BREP data. Because BREP data structure uses lots of memory, haptic development is problematic. D-Cubed does not support 1000 Hz refresh rates, which makes it unsuitable for stand-alone haptic rendering. Because CAD models and haptic models are based on different geometric representations, the use of haptic rendering has thus far not been able to be applied to the modification of CAD designs [41]. The next section talks about haptic modeling and the corresponding collision detection in more detail.

3.2 Geometric Modeling and Haptic Rendering

Three-dimensional objects for haptic rendering can be represented as surface-based or volume-based [50]. While the underlying geometry representation in most CAD software is the BREP, for display and rendering purposes polygonal and triangular meshes are created from the BREPs. Representing a surface geometry with a polygonal mesh is an

approximation process. Haptic rendering for polygonal meshes was introduced in 1995 by Zilles and Salisbury [42, 43]. Their method recorded contact history and provided stable force feedback interaction with polygonal models.

Another method of collision detection is bounding volumes for each object. Collision detection for bounding-volume objects is primarily based on penetration depth computations. Bounding volume hierarchies are made out of tight-fitting oriented bounding boxes (OBBs) [45]. The bounding volumes are limited to spheres, axis-aligned bounding boxes, oriented bounding boxes, and k-dops [42, 46]. These algorithms test if two objects, enclosed in bounding volumes, overlap. The objects are then subdivided and the bounding volumes are again tested for overlap. A drawback to this method is that it may not perform significant culling in close proximity configurations, resulting in a high number of false positives (for collisions) and wasted tests [47]. These algorithms are able to perform continuous collision detection. Most of the existing algorithms can achieve smooth, realistic haptic rendering when limited to a few thousand polygons or a few surfaces [45]. Most scenarios are limited to 10 to 100 pairs of convex primitives [48].

Modeling with polygons can offer a greater spatial accuracy than volume-based modeling, a major drawback to haptic rendering with polygons is the performance barrier when using highly complex models[49]. A simple object requires hundreds or thousands of primitives depending on model accuracy. Highly complex scenarios are therefore not possible without any major degradation of haptic update rates. Another way to generate a haptic model is to use a volume-based approach.

A volume-based model is constructed from voxels which are three-dimensional cubic elements. Volume-based objects are typically used when exact surface representation is not required.

Collision detection in volume-based models is done by sampling one volume and testing the inclusion of sampled points of the object against the voxels of another. Most of the haptic rendering algorithms that use volumetric representations use distance field methods. One, the voxmap pointshell method, will be discussed in more detail later. Cuisenaire gives an overview of the distance field algorithms [51]. Essentially, these algorithms use a uniform or adaptive grid to sample distances between objects. Cuisenaire and Kaufman used a probability map that tests two volumes and then assigns a "surface crossing" probability to each space point. A distance map is then used to increase calculation speed [52, 53]. Non-uniform volume rasters, such as irregular grids, cannot currently be modeled. Volume-based haptic rendering methods such as the voxmap pointshell method have been used to provide haptic rendering and collision detection for complex engineering tasks at haptic refresh rates close to 1000 Hz. Combinations of voxelized and polygonal models have been explored [54] and continue to be of interest.

3.3 Voxel-sampling-based Collision Detection

Three-dimensional objects for haptic rendering can be represented as surface-based or volume-based [50]. A volume-based model is constructed from voxels. The data is converted to voxels and inserted into an occupancy map. The occupancy map is a regularly

spaced grid of cells that tells the model its location on the voxel grid. Volume-based objects are typically used when exact surface representation is not required.

Collision detection in volume-based models is done by sampling one volume and testing the inclusion of sampled points of the object against the voxels of another. Most of the haptic rendering algorithms that use volumetric representations use distance field methods. One, the voxmap pointshell method, will be discussed in more detail later. Cuisenaire gives an overview of the distance field algorithms [51]. Essentially, these algorithms use a uniform or adaptive grid to sample distances between objects. Cuisenaire and Kaufman used a probability map that tests two volumes and then assigns a "surface crossing" probability to each space point. A distance map is then used to increase calculation speed [52, 53]. Volume-based modeling supports the required refresh rate, but non-uniform volume rasters, such as irregular grids, cannot currently be modeled. Combinations of voxelized and polygonal models have been explored [54] and continue to be of interest. Volume-based haptic rendering methods such as the voxmap pointshell method can provide reliable 1000 Hz refresh rates. They have been used in complex engineering tasks.

The different modeling methods have advantages and disadvantages for haptic rendering. In this research, combining volume-based haptic rendering with the precise geometric constraint information of a BREP model will bring the state-of-art in haptic rendering one step closer toward an entirely geometry-driven approach to haptic rendering. Using this hybrid methodology, geometric constraint recognition will be driven with BREP models, while the volumetric haptic representation will enable stable and fast simulation of

the forces and torques as well as collision detection. In addition, CAD models can be assembled with low clearance while precise haptic forces are computed. Geometric inaccuracies of the model will be mitigated because the voxel model will only be used for force computations.

3.4 Haptic Rendering

The method of geometric representation (modeling) drives what collision detection is used. Haptic rendering uses collision detection to generate contact forces to create the illusion of touching virtual objects. To obtain a collision or force response, the forces acting on the model or tool using collision detection as an input are computed. Collision detection is an essential element in haptic rendering because it detects potential violations of environmental constraints [42], but it can be computationally expensive for complex models.

Haptic update rates must be as high as 1000 Hz to display smooth and realistic forces and torques [47, 55]. Currently, real-time graphic applications have a refresh rate of 20 to 30 Hz [45]. However, the human tactile system can detect changes in forces at much higher frequencies of 500 Hz [56]. Therefore, it is critical that the haptic loop performs close to 1000 Hz. Sensitive collision detection via a high haptic refresh rate is critical because of the sensitivity of the human tactile system. Most existing haptic rendering methods can be divided into geometric-primitive-based and voxel-sampling-based haptic rendering. The

next section will discuss commercially available haptic devices and some research in bimanual haptics.

3.4.1 Haptic Devices

Being able to touch or feel a physical part is useful and intuitive for engineers. The sense of touch and kinesthesia can be enabled through haptic devices, which are input/output devices used in VR applications. Haptic feedback can add to the visual and audio feedback generated by VR as touch and force feedback is rendered. Burdea defines force feedback providing "real-time information on virtual object surface compliance, object weight, and inertia" [57]. Haptic devices require a refresh rate of 1000 Hz to produce convincing feedback [58].

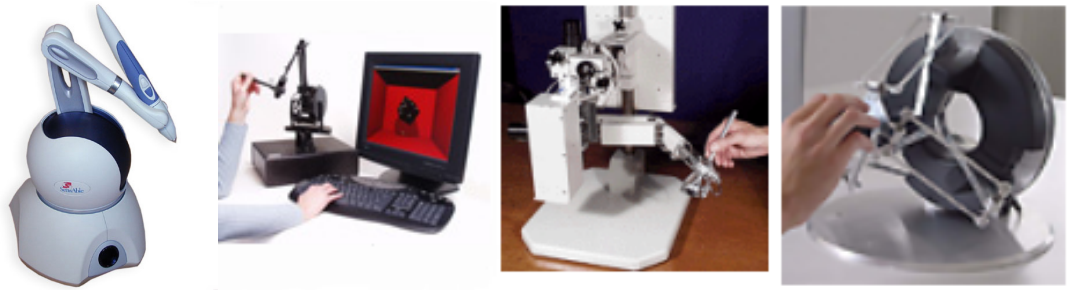

Figure 10: Desktop based Haptic Systems (left to right): SensAble's PHANTOM Omni, SensAble's PHANTOM Desktop, MPB-technologies Freedom 6S, Force dimension's 3-DOF Omega haptic device

Figure 11: Several Glove based Haptic Systems: Immersion's CyberGraspTM, Immersion's Haptic Workstation

Haptic devices can be categorized into ground-based (Fig. 10), glove-based (Fig. 11) and tactile displays. For this research, ground-based haptic devices will be used.

Force feedback devices such as SensAble's PHANToM Haptic Device have been implemented in multiple CAD and gaming environments. Some of the most widely used and commercially available haptic devices for more than three DOF are manufactured by Sensable (Personal HAptic inNTerface Mechanism or PHANToM)[1], Immersion Corporation (CyberForce)[2], Sarcos (Dextrous Arm Master)[3] and MPB Technologies (Freedom 6S)[4]. The PHANToM Omni developed by Massie and Salisbury [59, 60] will be used in this research. This haptic device allows six DOF and produces translational force feedback.

3.4.2 Dual-handed Haptics

Most of the research about two-handed interfaces and asymmetric hand usage can be found in human-computer interaction (HCI) literature. The use of dual-handed input has been of great interest to the mechanical assembly, medical surgery and free-form modeling community amongst others. Research suggests that two handed input provides a more natural interface for the user [61]. Guiard suggests that there are different roles for the dominant and non-dominant hand during tasks in the VE [62]. In his study, the non-dominant hand holds the part while the right hand manipulates tools. Furthermore,

[1] www.sensable.com
[2] www.immersion.com
[3] www.sarcos.com
[4] www.mpb-technologies.ca

complex tasks are divided into different roles, with each hand performing different subtasks [63].

Several researchers have implemented dual-handed haptics in their efforts to simulate assembly tasks in a virtual environment. The dual-handed haptic assembly simulations of particular interest to this research are HIDRA and SHARP. Although HIDRA uses two Phantoms, the user is only using his/her dominant hand to manipulate objects [16]. SHARP, on the other hand, allows the user to use both hands to interact with models. Seth *et* al. noted that dual-handed haptic implementation presents a major challenge to maintaining a haptic refresh rate of 1000 Hz [64].

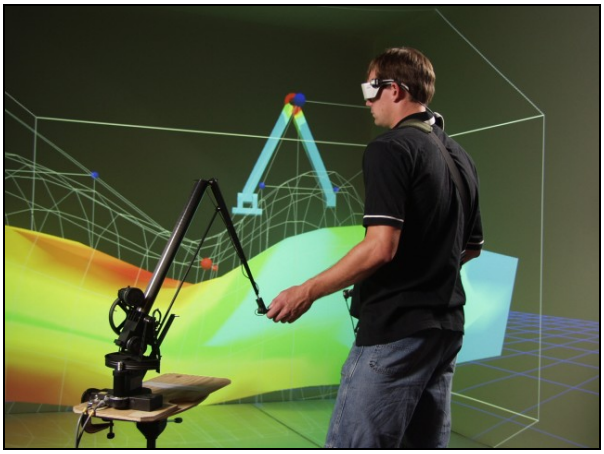

Figure 12: A User with a PHANTOM Haptic Device in a CAVE [65]

Although most haptic devices are used in conjunction with a 2D monitor as part of a desktop system, haptic feedback can be added to a projection screen virtual environment [65] (see Fig. 12). Within the projection area, a movable stand allows the user to move in the virtual environment while interacting with models. Research in the area of networked

haptics has allowed several users to interact simultaneously with 3D models in a collaborative environment [20].

3.5 CAD Data Exchange

CAD-VR data exchange is an important issue facing the VR community. CAD systems used by industry and academia are typically not suitable for use in a VR environment. Most VR applications use scene-graphs (OpenSceneGraph, OpenGL Performer, OpenSG, etc.) for visualization, which require simplified polygonal geometry to ensure interactive frame rates. Obtaining high fidelity graphical representations is difficult when dealing with real-time interactivity where graphical rendering must be real-time as well. Currently, there are few standardized and non-proprietary ways to convert CAD data into a representation that is suitable for VR.

During this CAD data translation process, parametric information or BREP data is not contained in the graphical representation. This requires VR applications that rely on geometric constraints to deal with two representations: one graphical and one with BREP information. Similarly, physical-based methods also require two separate representations of the CAD model in order to enable VR assembly simulations. One model is required for the graphical representation and the other representation is typically much coarser and used for interactive physical simulation.

Particular problems present the pre-definition of geometric constraint data for virtual assembly simulations. Special CAD toolkits are required to determine physical properties, geometric constraints, final transformation matrices, or other assembly data,

which ties such virtual assembly methods to those particular CAD systems. In addition, manual pre-processing is often required. Because of the additional required work to pre-process the models, this hinders widespread acceptance of virtual assembly methods. Several CAD software companies, such as UGS and Dassault System, have taken up the challenge and are providing interfaces for immersive, stereo visualization. However, most of them lack haptic feedback and provide only limited interaction with parts

CHAPTER 4. METHODOLOGY

Combining assembly operations with VR has brought about numerous approaches to deal with part interactions. One of the setbacks is that some virtual assembly applications cannot handle scenarios where the final position of parts in the assembly is unknown. In addition, model pre-processing also hinder the use of virtual assembly tools in industry.

This research tries to answer the challenges outlined above by combining haptic feedback with geometric constraint-based part guiding in order to provide a more realistic approach to assembly simulations in VR. Low clearance assemblies will be possible with this approach, while giving the user the freedom to manipulate parts without any pre-existing final positions or pre-defined geometric constraints. Human decision making is critical to identifying potential errors in the assembly and design processes. This chapter discusses the hybrid methodology in detail.

4.1. Voxmap PointShell Method

The voxmap pointshell method is a volume-based collision detection package developed as a fast collision detection method for complex models [66, 67]. The voxmap pointshell method was developed by McNeely, Puterbaugh and Troy for Boeing Company and was created specifically for large assemblies. The voxmap pointshell method can sustain the necessary 1000 Hz haptic refresh rate to render haptic forces smoothly. Kim and Vance

examined several collision packages and compared them with Voxmap PointShell (VPS) [18]. The voxmap pointshell method was found to provide realistic collision detection and physically-based interaction modeling with good performance [20, 68].

The voxmap pointshell method utilizes a collision detection method based on probing a voxelized part with surface point samples [67]. The surface of a scene object is voxelized, and the grasped or dynamic object is represented by a set of surface point samples called a pointshell. In addition, an inward pointing surface normal is associated with the pointshell. The state of the object is computed at every frame by solving for the position of quasi-static equilibrium.

In the voxmap pointshell method, models in the scene are voxelized in a pre-processing step to a voxel size specified by the user. The model is partitioned into regions of free space, object surface, and object interior by using a volume occupancy map or voxmap. The pointshell is then created by using all the center points of all surface voxels. This space is partitioned using a volume occupancy map or voxmap to create the voxels. Static objects are represented by voxels, while dynamic objects (moving objects) are represented by a pointshell set based on the voxelized model. The pointshell of the dynamic object is created by identifying all of the center points of all surface voxels of the dynamic object. Surface normals are also a part of the pointshell data. Distance fields are implemented to give advance warnings of potential pointshell and voxel collisions [69] (Fig. 13A).

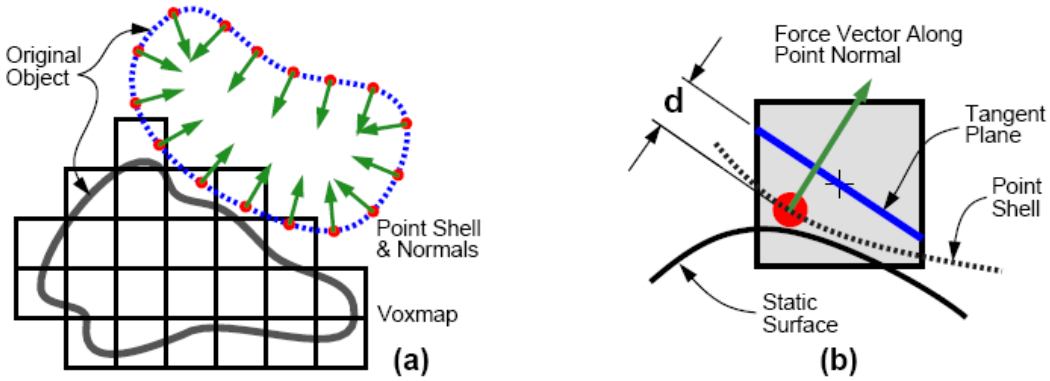

Figure 13: (A) Point-Voxel Collision Detection, (B) Tangent Plane Force Model [70]

The intersection of the pointshell with the voxel model is used as the basis for the collision detection and the haptic rendering. The collision detection module checks for inclusion of the sample points in the scene voxels, and then a local force model is applied (Fig. 13B). A depth of interpenetration, d, is calculated when a point intersects a voxel. Based on this depth, a force is calculated based on Hooke's Law ($F = kd$), where the force field stiffness is k_{ff}. The spring stiffness is set to a high value to reduce moments of inertia calculations. A mass must be assigned to the dynamic object equal to the apparent mass for that object (Fig. 14).

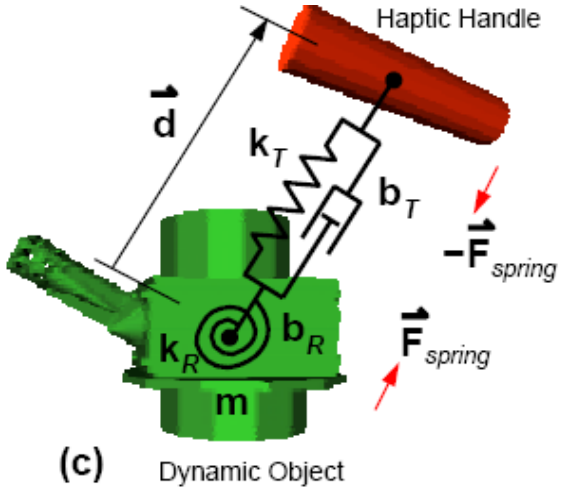

Figure 14: Virtual Coupling Model [70]

In order to avoid instability, a coupling force is applied between the virtual manipulator, or haptic handle as shown in Fig. 14, and the dynamic object. This coupling force and torque are composed of a spring force and torque. Damping terms are included in both equations. The spring force, F_S, and torque, τ_S, are computed as:

$$\vec{F}_{Spring} = k_T \vec{d} - b_T \vec{v} \quad \textbf{(1)}$$

$$\vec{\tau}_{Spring} = k_R \vec{\theta} - b_R \vec{\omega} \quad \textbf{(2)}$$

where k_T and k_R are spring constants, b_T and b_R are damping coefficients, d and θ are distance measures and v and ω are relative velocities.

The collision force and torque are calculated by the voxmap pointshell method is proportional to the amount of inter-object penetration d and a force field constant k_{ff} (Eqn. 3 and 4). The collision force, $F_{Contact}$, and torque, $\tau_{Contact}$, are computed as:

$$\vec{F}_{Contact} = k_{ff} \vec{d} \quad \textbf{(3)}$$

$$\vec{\tau}_{Contact} = k_{tt} \vec{\theta} \quad \textbf{(4)}$$

where k_{ff} and k_{tt} are spring constants, d and θ are distance measures. The maximum distance a part can move is defined as maximum travel. To reduce the processing load, a speed limit is imposed by the voxmap pointshell method. This is also prohibits deep penetrations arising from collisions at high velocities. A pre-contact braking force is calculated when deep point-voxel penetrations occur. This reduces the point velocity when approaching a voxel. The breaking force, $F_{Breaking}$, and torque, $\tau_{Breaking}$, is based on the spring and collision forces:

$$\vec{F}_{Breaking} = -b_{ff} \vec{v} \quad \textbf{(5)}$$

$$\vec{\tau}_{Breaking} = -b_{tt}\vec{\omega} \qquad \textbf{(6)}$$

where b_{ff} and b_{tt} are damping coefficients. The total collision force, $F_{Collision}$, and torque, $\tau_{Collision}$, is based on the breaking and contact forces (Eqns. 7 and 8).

$$\vec{F}_{Collision} = k_{ff}\vec{d} - b_{ff}\vec{v} \qquad \textbf{(7)}$$

$$\vec{\tau}_{Collision} = k_{ff}\vec{\theta} - b_{ff}\vec{\omega} \qquad \textbf{(8)}$$

The total force, F_{Net}, and torque, τ_{Net}, is based on the spring and collision forces (Eqns. 9 and 10).

$$\vec{F}_{Net} = \vec{F}_{Spring} + \vec{F}_{Collision} \qquad \textbf{(9)}$$

$$\vec{\tau}_{Net} = \vec{\tau}_{Spring} + \vec{\tau}_{Collision} \qquad \textbf{(10)}$$

In addition to rendering forces from the virtual coupling (spring-damper system) between the dynamic object and the virtual hand, the voxmap pointshell method models part interaction using a module called Physically-Based Modeling (PBM). These interactions use rigid body dynamics to describe the dynamic state of a rigid body at time t. The basic equation of motion must satisfy the Newton-Euler equations of motion:

$$\vec{F}(t) = M\ddot{\vec{x}} \qquad \textbf{(11)}$$

$$\vec{N}(t) = I\dot{\vec{\omega}} + \vec{\omega} \times I \bullet \vec{\omega} \qquad \textbf{(12)}$$

where x is the linear displacement, F(t) is a linear force along time t, M is the object's mass, w is the angular velocity, I is the moment of Inertia, and N(t) is the angular force. The voxmap pointshell method solves these equations linearly using finite difference approximations. The resulting position and velocity offsets between the hand proxy and the dynamic object change the coupling force and the torque, which is then fed back to the haptic device.

Since objects are voxelized only on the surface, deep penetrations, which can occur if objects collide at high velocities, cannot be handled. This eliminates potential instability problems that can arise due to high contact stiffness. Deep penetrations are avoided by formulating the coupling force as a non-linear spring. Because the pointshell may contain too many points to be tested in a single haptic frame, an estimate of the minimum distance from any contact is calculated based on the distance field.

Parts need to be moved at high enough speeds to follow the natural motion of the human hand. If the application refresh rate drops, the object appears lagging behind the hand movement. Voxel size and the voxmap pointshell method update rate affect the part's speed. A maximum time period can be defined for the part interaction calculation. The CPU will try to perform part interaction calculations until they are solved within the maximum time or the maximum distance the object can move per frame to reduce calculation time less than the maximum time.

Hierarchical culling of sample points can improve processing speed, but ultimately the computational cost depends on the number of contact points. The voxmap pointshell method will allow up to a 32 bit memory allocation for each voxel and can store additional information such as surface, edge and vertex information of the BREP is closest to. Each voxel has 26 neighbors that share a vertex, edge or face with the neighboring voxel. In addition, the voxmap pointshell method has proximity detection. For any point on the pointshell, the voxmap pointshell method knows how far that point is from any object. The closer a point approaches a chosen object, the more often the voxmap pointshell method

samples it. During the sampling process, a tree approach is used to determine how close the parts are [67].

Because the voxmap pointshell method is a volumetric-based approach to haptic rendering, the accuracy of the collision detection and haptic rendering is inversely proportional to the voxel size. Smaller voxels will allow higher collision accuracy, but at small sizes, more voxels are required to represent the part, which increases the memory requirements of the application. Smaller voxel sizes may also increase the computational load because more voxel-pointshell collisions occur. When a pointshell penetrates a tangent plane that passes through the voxel's center point, a depth penetration is computed. Voxel size plays a crucial part during part collision and interpenetration. Due to the nature of the voxelization process, low clearance parts cannot be assembled because of this accuracy limitation. The limitation of voxel-based collision detection is its voxel-scale accuracy [67].

4.1.1. Pointshell Shrinking

A pointshell is a collection of points on the moving object that represent the centers of each surface voxel. Within the VPS software, there exists the functionality to specify an offset distance. This offset distance moves the point of each pointshell along its outward-pointing normal by a distance defined by the user, in order to obtain finer-grain control over surface offsetting. Setting a positive offset moves the point outward which is sometimes implemented to guarantee no penetration occurs between colliding objects. However, when low clearance assembly is required, moving these points out from their initial positions further prevents assembly. The approach here is to use pointshell shrinking which

pulls points closer to the interior of the object along the surface normal direction. One advantage of using pointshell shrinking is that this method does not require re-voxelization.

Because voxels are created on the boundaries with center points on the surfaces, they project beyond the surface of the model. The problem with this approach is that this creates potential collisions where technically there should be none. The pointshell can be shrunk when parts are on the verge of being successfully mated. Boeing has demonstrated this approach by inserting a bolt into a hole.

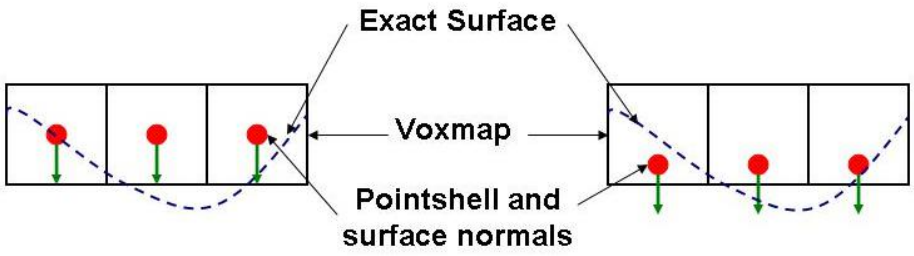

Figure 15: Normal and Shrunk Pointshell Location

Pointshell shrinkage is defined as a percentage of the voxel size. Because voxels are created using a volume partitioning approach, the points in the pointshell sometime exist beyond the surface of the model (Fig. 15). Pointshell shrinking has the potential to move these points to the surface or to the interior of the dynamic object, which could result in potential collisions where technically there should be none. It is possible to shrink the pointshell more than the voxel distance. This may cause the pointshell to do a local spatial inversion leading to erroneous forces. Unfortunately, there is no internal limit to curtail the amount of shrinkage on a point-by-point basis to prevent inversion. However, there is

virtually no run-time cost for shrinking the pointshells, which is important to maintaining the haptic refresh rate.

The major limitation to this approach is that the user needs to tell the voxmap pointshell method when to shrink the pointshell. This can be done in advance, but may lead to detecting collisions that are not actual collisions between parts, but instead collisions between random boundary voxels. Therefore, it would be advantageous to be able to dynamically shrink or expand the pointshell.

As a recommendation by McNeely, 50% smaller point-shells should be used for the voxelized model. The voxmap pointshell method will provide the collision detection and haptic feedback while D-Cubed will be used for geometric constraint management. While pointshell shrinking allows tighter fits between parts, it will not be used in this research because geometric constraint based and collision-based forces will be blended.

The voxmap pointshell method uses a tetrahedral mesh to produce a voxelized model. Because of the cubic voxelization process that the voxmap pointshell method uses, hole sizes are reduced in diameter compared to the graphical representation of the part. This becomes problematic during low clearance assemblies when a pin is inserted into a hole. Multiple collisions occur with tight clearances, and forces are sometimes calculated before the CAD surfaces actually touch each other. This approach works well for most assembly scenarios, but due to the geometry approximation, the voxmap pointshell method does not allow low clearance assembly due to the maximum offset discussed above. Reducing the voxel size to produce a better geometry approximation is sometimes not feasible due to the increased memory requirements for small voxels.

When low clearance objects are assembled, due to the approximation of their geometry, collisions will occur when they should not. Knowing which voxels are colliding with each other allows the methodology to search for the corresponding boundary representations. D-Cubed will then be used to determine if geometric constraints are applicable to these BREPs. If they are, the voxmap pointshell model is then turned off. If a geometric constraint can be applied to the identified BREPs, the models are reoriented to match the constrained and can then be reassembled while the voxelized model is dormant. For example, when a pin is inserted into a low clearance hole, the BREP geometry will guide the pin while haptic rendering is disabled at that instance. Collision and contact forces will be calculated based on pointshell shrinking at that time.

4.2. Geometric Constraint Recognition and Solver

Geometric constraint based modeling will allow low clearance parts to be assembled while maintaining realistic part behavior. The voxmap pointshell method provides haptic rendering and collision detection, but the approximation of the CAD geometry prevents the assembly of low clearance parts because collision of voxels will occur where they shouldn't. Geometric constraint recognition will provide a solution by providing geometric constraints when parts are close enough for geometric constraints to be applied. The geometric constraints will be solved, aligning the parts based on the geometric constraint requirements while haptic rendering is paused during the geometric constraint based, highly accurate final assembly step. D-Cubed, a family of software components from

Siemens UGS [71] is used for the geometric constraint recognition part of the test bed. There are four D-Cubed modules as shown in Fig. 16.

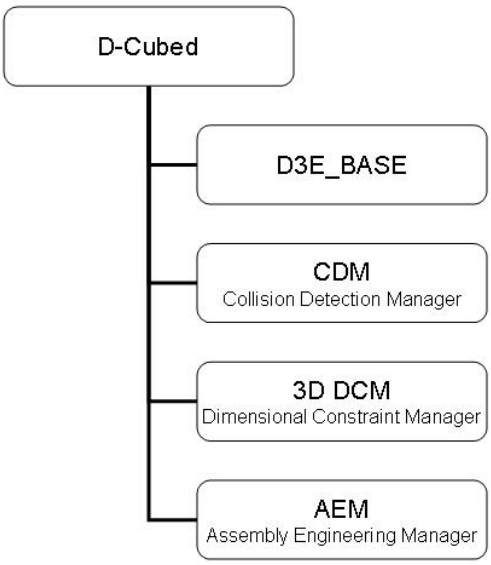

Figure 16: D-Cubed Modules.

The D3E_BASE module holds the model data structure definitions and specifies which 3D geometry model format is used for the other D-Cubed components. The BREP model will be loaded using the D3E_BASE module, which makes this information available for the other components of D-Cubed.

D-Cubed's Collision Detection Manager (CDM) is responsible for managing collisions and interferences. The CDM uses the BREP data loaded by the D3E_BASE to query for collisions. This allows for highly accurate collision detection. Collisions between part surfaces can be detected with an accuracy of 0.0001 mm [34] based on parts modeled by Seth *et* al.

The Assembly Engineering Manager (AEM) performs physical simulation [72]. BREP data is used by both the CDM and the AEM, which allows a model's physical constraints to be highly accurate. The AEM module treats collisions between the BREP surfaces as contacts and calculates part trajectories based on these collisions during assembly. It integrates mass, inertia properties and gravity to implement realistic model behavior. The AEM also handles movement of the models in the environment. The physical dragging algorithm allows the user to move selected parts in a realistic way. The AEM also handles contact and permanent geometric constraints. The contact constraints are used to check for interpenetrations and for simulating realistic part behavior. The geometric constraints restrict the degrees of freedom of the part; for example, fixing the part in space so that it cannot move in any direction. Part behavior is based on the part's mass and inertia properties during AEM movements. For the hybrid method, the AEM will not be used because the voxmap pointshell method handles the physical behavior of the parts.

The 3D dimensional constraint manager (DCM) is the constraint solver for D-Cubed. It uses dimensions and geometric constraints to position parts in the assembly, and is responsible for positioning parts based on their geometric constraints. Concentric, coincident, tangent and parallel constraints can be defined through this module. This module also handles over- and under-constrained parts. The constraint manager will also choose a solution from among several possible solutions so that the geometry arrangement is maintained. In addition to the geometric constraint handling, the DCM also is able to modify part shapes and can be used to generate 3D sketches. Distance, angles, curves and surfaces can be implemented with this module.

SHARP by Seth *et* al. investigated an approach that combined physical and geometric constraints [34, 73]. While the physical constraint simulation enabled realistic part behavior during collisions, the geometric constraint based simulation allowed precise part manipulation and assembly. Seth *et* al. used D-Cubed as their constraint solver [74].

Any geometry or rigid set in three dimensional Euclidean space has six degrees of freedom (three translations and three rotations), minus the number of translation symmetries and rotational symmetries possessed by the geometry, plus any internal degrees of freedom. Using this it can be shown that a line has four degrees of freedom, a plane has three, a circle has six (including one internal degree of freedom), a cylinder has five (including one internal degree of freedom), a sphere has four, a swept surface has five and an evaluated parametric curve or surface has six. The model would be solved by calling the evaluate function. The DCM will calculate the resulting transformations to the model, and the application can read these and supply the new transformation matrix to calculate geometric constraint forces and torques for the hybrid method. There are several different types of geometric constraints and dimensions that can be applied to a parametric geometry. In particular, tangent, coincident, distance dimensions, alignments, and surface orientations are possible.

4.3 Hybrid Method

This section describes the implementation of the hybrid haptic-constraint algorithm. Previous research by this group has yielded a reliable geometric constraint based test bed

that will be integrated with the haptic feedback algorithm of the voxmap pointshell method. This method relies on the voxmap pointshell method to provide better refresh rates than other collision detection methods. Tying boundary representation to the voxels will allow the geometric constraint recognition algorithm to “piggy-back” on the voxmap pointshell collision detection.

This hybrid approach will be using three separate geometry models: a tessellated model for visualization and display, a voxelized model for the voxmap pointshell method collision detection and force calculation, and a BREP model for geometric constraint recognition. The three models will be bound together in the virtual environment and must be moved synchronously. The user will be able to select and move parts in the environment. The voxmap pointshell method will be responsible for collision detection based on the voxelized model. Once collisions occur, geometric constraints will be checked by identifying the colliding BREPs. If valid geometric constraints are found, the geometric constraint algorithm will apply those geometric constraints to help guide the user in the assembly step. To model realistic model behavior during insertion of models, collision forces are calculated based on the voxmap pointshell method collisions and are blended with geometric constraint based forces.

Additional information about the model can be tied to voxels through expansion of the voxel data definition within the voxmap pointshell method. The voxmap pointshell method can, in turn, access this additional information during pointshell-voxel collision and then decide if a geometric constraint needs to be applied. This method would create a way of tessellating BREP data prior to using the virtual environment. The BREP information

would be saved in the tessellation, and software will then convert a tessellation to a voxelized model while preserving the BREP information from the previous step (Fig. 17).

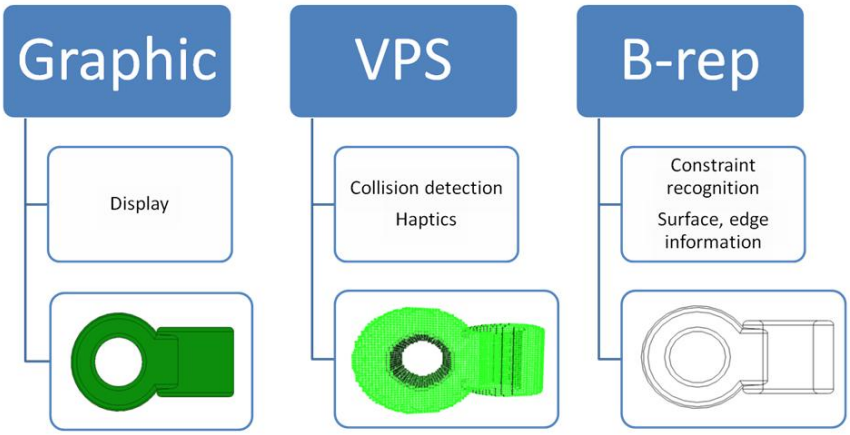

Figure 17: Model Flow Chart

Once the voxelized model carries BREP information, the next step is to develop an algorithm that can decide the likelihood of needing to apply a geometric constraint, and how a geometric constraint can be phased in. Timing is important because if the geometric constraint is applied too soon, the haptic force will jump unnaturally as the graphics model is pushed toward the constraint. Conversely, if the geometric constraint is applied too late, the system may not recognize it, which would cause errors.

4.3.1. Performance

The key to this approach is both the data structure and the logic related to switching between the voxmap pointshell method and geometric constraint recognition. Performance is a key issue to be explored. A 1000 Hz haptic refresh rate must be maintained at any cost. Performance-enhancing measures such as cluster computing may be necessary to complete

this project. Ultimately, it is the goal of this research to generate an immersive environment where the user can interact naturally with the objects. A decrease in collision detection and haptic performance will impact the goal negatively and cloud the evaluation of the hybrid approach. An evaluation of the hybrid method to test an assembly sequence planning task should be performed to assess the method's effectiveness.

CHAPTER 5. VOXMAP POINTSHELL-CONSTRAINT HYBRID METHOD

The goal of this research is to combine haptic feedback with geometric constraint-guided assembly. The research explores and develops methods to combine VPS-based haptic feedback with BREP constraints to allow realistic part interaction. Direct CAD input, low cost and precise collision detection are emphasized to produce a virtual assembly simulation that is capable of assembling low clearance parts. This research provides a methodology that allows low clearance collision detection for virtual manual assembly with haptic force feedback. This methodology uses a novel algorithm to simulate natural part interactions in a virtual environment that is based on a hybrid voxel/boundary representation. A data structure to support voxel, BREP and tessellated models was developed to support collision detection, BREP identification, automatic geometric constraint identification and haptic force and torque generation (Fig. 18). In addition, contact forces and torques related to geometric constraints must be generated to provide convincing haptic feedback to the user.

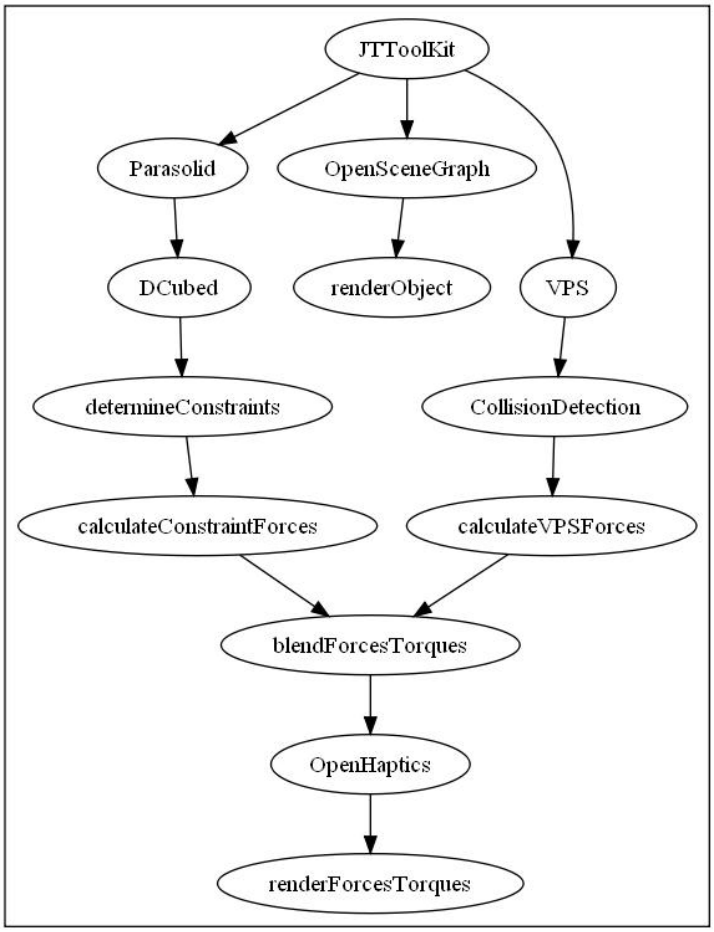

Figure 18: Force and Torque Flowchart

Current geometric constraint based methods cannot provide reliable 1000 Hz refresh rates for haptic feedback, but they do allow very accurate collision detection by using boundary representation of CAD parts. The use of the exact geometric representation of a given part allows low clearance and geometric constraint based assemblies, whereas geometry approximation methods such as the voxmap pointshell method do not support exact geometry collisions or geometric constraint based movements. However, the voxmap

pointshell method allows reliable 1000 Hz haptic refresh rates and can calculate collision forces and torques within the refresh rate.

5.1. Hybrid Methodology

This section describes the implementation of the hybrid haptic-constraint algorithm. Previous research by this group has yielded a reliable geometric constraint based test bed that will be integrated with the haptic feedback algorithm of the voxmap pointshell method.

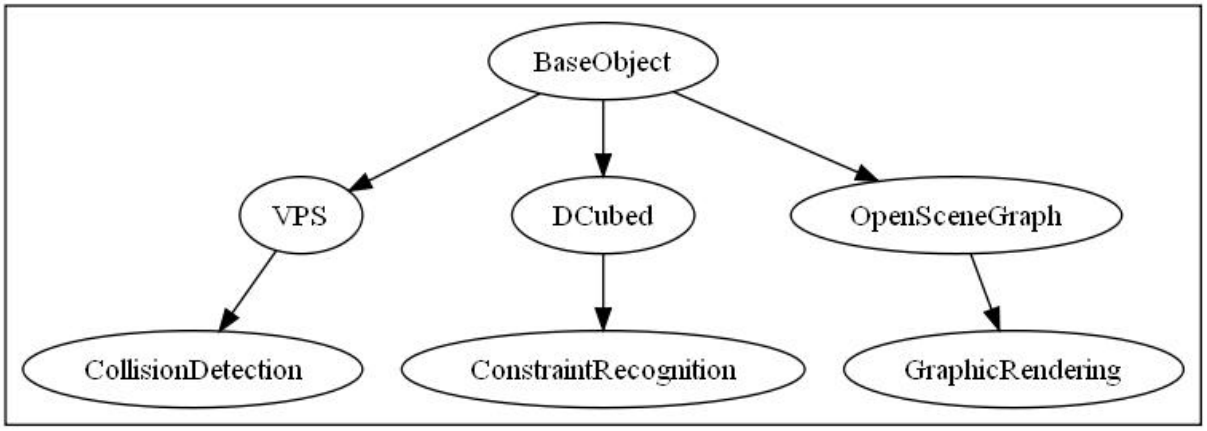

Figure 19: Binding of Separate Geometry Models in the Hybrid Method

This hybrid approach will be using three separate geometry models: a tessellated model for visualization and display, a voxelized model for the voxmap pointshell collision detection and force calculation, and a BREP model for geometric constraint recognition. The three models will be bound together in the virtual environment (Fig. 19). The user will be able to select and move parts in the environment. Internally, the application switches on and off a geometric constraint recognition thread while the collision detection thread constantly runs.

The voxmap pointshell method will be responsible for collision detection based on the voxelized model. Once collisions occur, geometric constraints will be checked by identifying the colliding BREPs. The voxmap pointshell method will be used to determine which faces are colliding with each other through the use of stored data in the voxels and return a pointer to the colliding faces. This pointer then is supplied to the geometric constraint solver method to determine if geometric constraints can be applied to the parts in contact based on the type of face supplied. If valid geometric constraints are found, D-Cubed will apply those geometric constraints and calculate a new position matrix based on the constraint for the dynamic part. A constraint force and torque is calculated and supplied to the voxmap pointshell method. The methodology then determines if a voxel colliding is part of the constrained face. In order to prevent any geometric constraint based forces and torques must be phased. To model realistic model behavior during insertion of models, contact forces and torques are calculated when voxel to pointshell collision occurs. Constraint based forces and torques are calculated based on constrained position matrices, but must be removed when voxels that are colliding are determined to be in a constrained contact.

5.2. Tying BREP Info to Voxels

To switch between voxel-based collision detection and geometric constraint-guided low clearance assembly, boundary information must be tied to voxels. This approach combines the best capabilities of two advanced methods and should allow for more

intuitive virtual manual assemblies through the use of haptic force feedback and geometric constraint-guided low clearance parts. At this point, tessellated data cannot be associated with boundary representation data. They both contain unique information about the part. Triangle data from CAD parts cannot be directly associated with face or edge BREP info at this point. While there are methods that can tessellate boundary representations, they are not open source and would require preprocessing of CAD data, which was not desired for this methodology.

Additional information about the model can be tied to voxels through expansion of the voxel data definition within the voxmap pointshell method. The voxmap pointshell method can, in turn, access this additional information during pointshell-voxel collision and then decide if a geometric constraint needs to be applied. This method would create a way of tessellating BREP data prior to using the virtual environment. The BREP information would be saved in the tessellation, and software will then convert a tessellation to a voxelized model while preserving the BREP information from the previous step.

JtOpenToolKit allows access to BREP and triangle data. However, there is no connection between BREP data and level of details (triangle data) within JT. Although there may be methods that can tessellate BREP data, they are currently not open source. Therefore, this methodology needs to reconstruct the connection between voxels and BREP data. The association between voxels and the BREP must be done in a way that the geometric constraint solver understands.

The *.jt file contains BREP data. This BREP data can be accessed and saved to a Parasolid (*.x_t) file. The Parasolid file is then read into D-Cubed. D-Cubed then adds the

part to its own geometric constraint scene and creates a face and edge array. At the same time, the voxmap pointshell method is supplied tessellated data and then in return voxelizes the part. Within the voxmap pointshell method, a voxel traversal can occur. This traversal allows the user to gain information about the individual voxels such as the position and add data through the use of the voxel expansion. Through this traversal, a BREP face or edge can be directly associated with any given voxel. Each voxel can be traversed once the BREP data is loaded into D-Cubed. A D-Cubed test cube, same size as a voxel, will be collided with the BREP (D3 representation of the model).

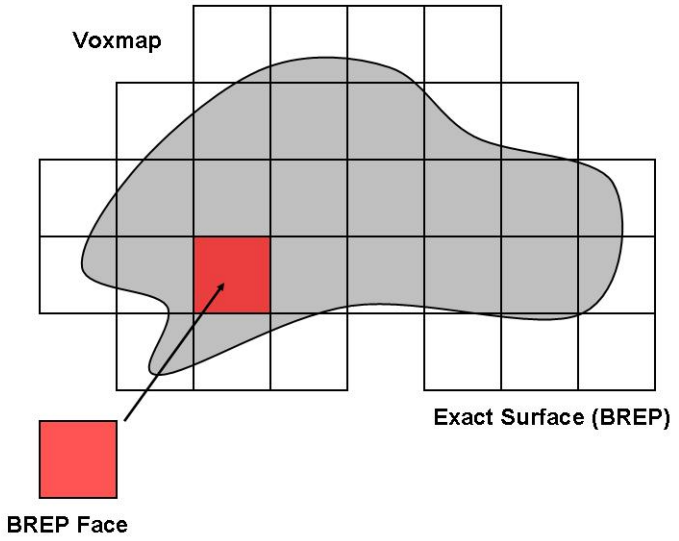

Figure 20: Association of BREP Face ID with Voxels

The test cube will be moved based on the voxel position determined in the traversal step. Next, the collision detection will determine which face or edge the voxel belongs to. A look up list is created for each voxel (Fig. 20). The face and edge data can then be stored in the list. A pointer to the list can be stored in the private data of the voxel.

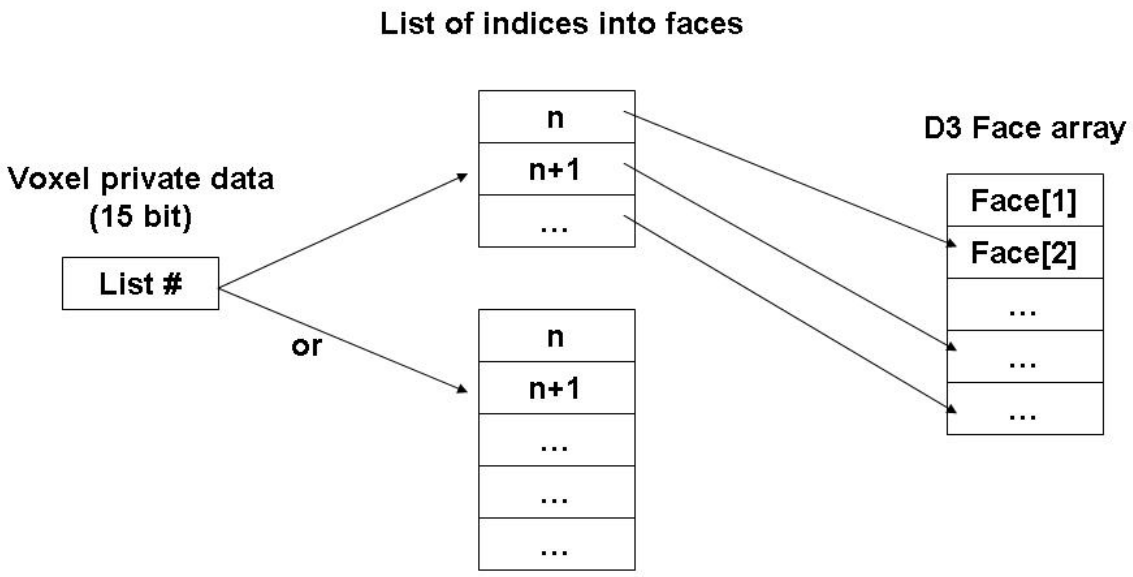

Figure 21: Lookup Table for BREP Face and Edge Data

During the voxmap pointshell collision detection, the private data stored in each voxel can be accessed to reveal which faces or edges the voxels are colliding with. The voxmap pointshell collision detection returns each voxel that is in collision with another pointshell (Fig. 21). Based on the collision results, the lookup table is used to determine which faces and edges the colliding voxels belong to. This in return is supplied to D3. Our logic will test if geometric constraints can be applied. Additionally, the voxmap pointshell method knows if a voxel is on the surface or the inside. The voxmap pointshell method will only return the private data of voxels that are located on the surface and ignore any other voxels.

Fig. 22 shows a typical assembly with a peg and a hole. Once collision occurs, faces and edges that belong to colliding voxels are identified on both parts. These face and edge identities are then supplied to the geometric constraint algorithm to access face or edge

arrays and then to determine if geometric constraints can be applied. The geometric constraint recognition algorithm will then calculate a new position matrix. Using the new and the previous position matrices, constraint forces and torques are calculated.

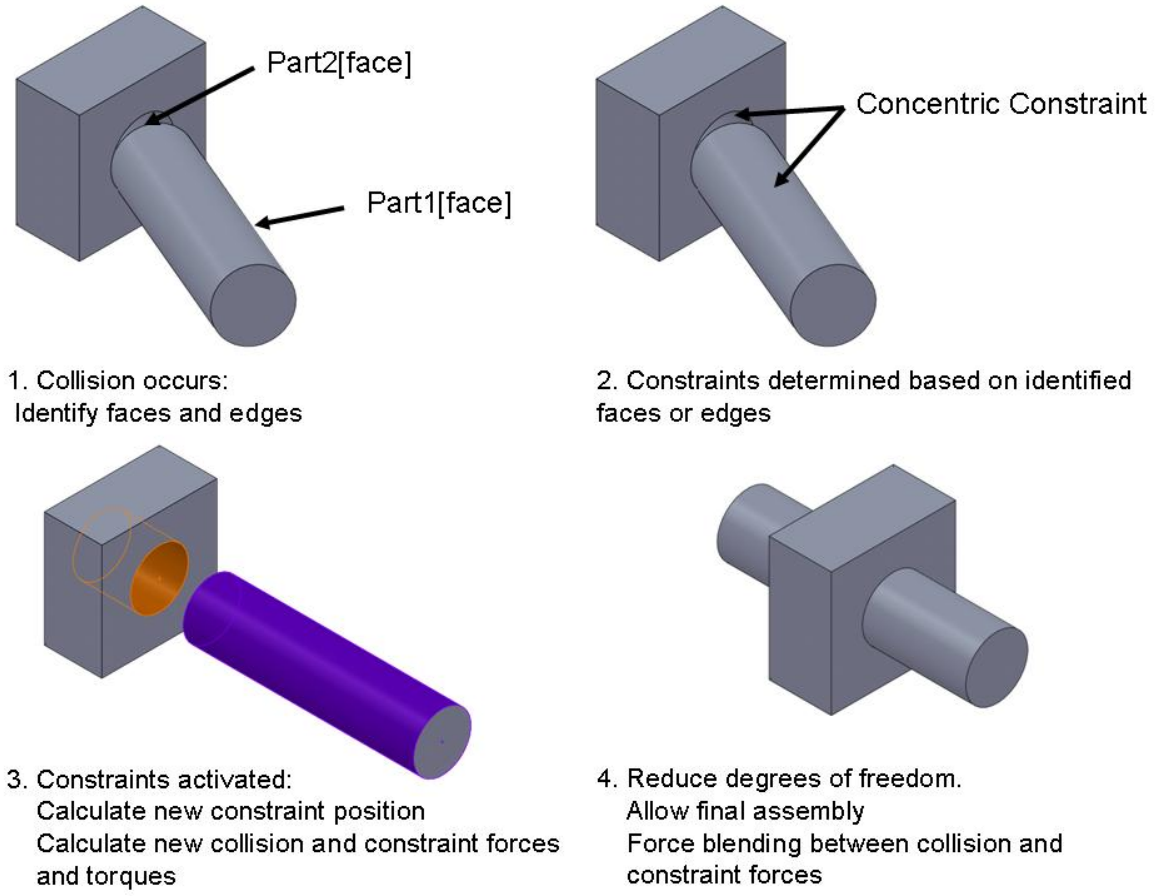

Figure 22: Assembly Sequence for Hybrid Method

5.3. Constraint Recognition

Geometric constraints are constant, non-numerical relationships between the geometric entities of a CAD part. Examples of geometric constraints include parallelism, perpendicularity, and concentricity. A planar constraint can be applied to planar surfaces with a non empty common area. A cylindrical constraint or cylindrical joint is defined

between cylindrical surfaces. Unidirectional cylindrical constraints are defined between conical surfaces. Spherical constraints can be applied between spherical surfaces with the same center, radius and opposite normals. Linear annular constraints are defined as a spherical (convex) surface lying inside a cylindrical (concave) surface.

For this particular method, concentric, planar and spherical constraints were chosen. These particular constraints are used most often in assembly tasks. The geometric constraint recognition algorithm will decide whether two faces are cylindrical or if they are planar. If both faces are determined by the algorithm to be cylindrical, a cylindrical constraint will be applied to both parts (Fig. 23).

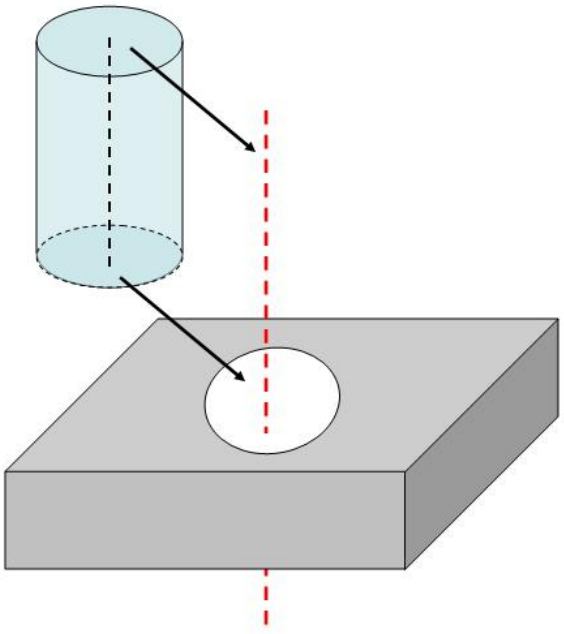

Figure 23: Positioning of a Cylindrical Peg over Hole Using Coincidence of the Cylinder Axis and the Joint Axis.

For the concentric constraint, a center axis is applied to both parts and the movement of these parts is restrained to movement along that center axis. If both faces are

determined to be planes, a planar constraint will be applied. The planar constraint allows the parts to move parallel to each other on a plane. Fig. 24 shows the geometric constraint algorithm loop. If no geometric constraint can be applied, the geometric constraint loop will simply return to the collision detection loop (Fig. 24).

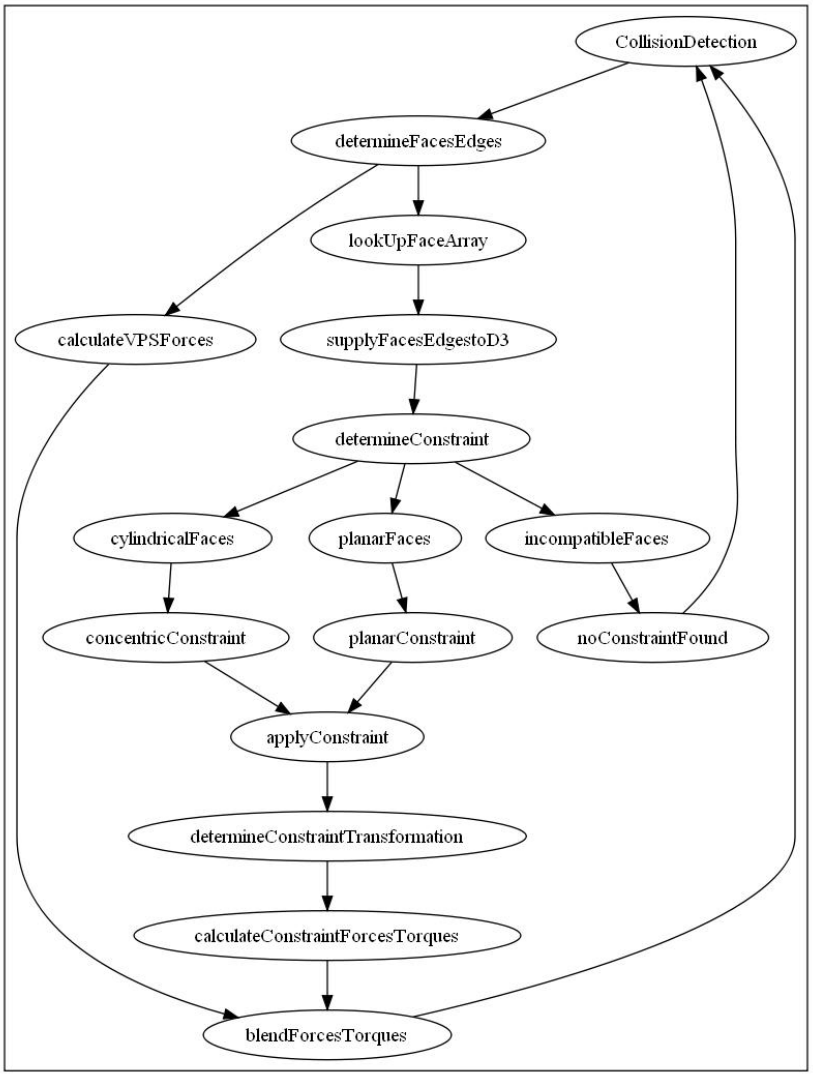

Figure 24: Constraint Recognition Algorithm

The geometric constraint recognition algorithm can be expanded to allow for other constraints. At this point, only concentric and planar constraints were tested during the hybrid method evaluation.

5.4. Blending Constraint with Collision Forces

Once the voxelized model carries BREP information, the next step is to develop an algorithm that can decide the likelihood of needing to apply a geometric constraint, and how a geometric constraint can be phased in. Timing is important because if the geometric constraint is applied too soon, the haptic force will jump unnaturally as the graphics model is pushed toward the geometric constraint. Conversely, if the geometric constraint is applied too late, the system may not recognize it, which would cause errors. The voxmap pointshell method only detects collisions between vertices-to-surfaces, vertices-to-faces and-edge-to-edge collisions. The geometric constraint recognition algorithm using D3 will be able to complement the voxmap pointshell method because it can handle face-to-face collision detection and determine appropriate geometric constraints for such collisions, while allowing low clearance parts to be assembled. “Alignment forces and torques" means the forces (and torques) that are generated to represent the geometric constraint, e.g., for a cylinder-on-cylinder constraint, it is a torque that aligns the axes plus a force that pulls axes into congruency. Collision forces and torques are calculated by the voxmap pointshell method collision detection.

The force blending algorithm will use two calls to the voxmap method collision detection. The first call will be a general collision test against all static objects. No scale factor will be applied to the collision forces and torques. At this point, the alignment forces and torques are null. This first collision detection will return only voxels that are colliding with each other. Those voxels are then queried for face and edge identifiers through the voxel data expansion.

The second call to the voxmap pointshell method collision detection will only test bodies that are colliding. Scale factors for force and torque blending are set. This step may require a capacity or some sort of filter to smoothen out scale factor. The voxmap pointshell collision detection then adds all contact forces when voxel to pointshell collision occurs. The algorithm will not add forces when voxels are determined to be in a constrained contact. This can be done through a list of all collision forces and torques. The list is then queried to determine what forces and torques need to be added. This can be done through the voxel data expansion where edge and face array is known. When the geometric constraint recognition algorithm determines a geometric constraint, those faces and edge arrays must be cross-referenced with the voxmap pointshell method. The collision detection then can blend forces and torques with known geometric constraints. When the geometric constraint recognition algorithm determines a new position, an angle between the two objects can be determined. When the angle is large, a weak force must be applied to align the parts. A smaller angle between the parts will require a larger force. The following is a more detailed description of the force blending algorithm.

A floating-point variable "x" is defined to express how fully the geometric constraint is engaged at any given time. X varies from 0 to 1, where 0 means "not at all engaged" and 1 means "fully engaged". The force blending factor is dependent on the distance of separation and angle of rotation between the geometric constraint and the object. For example, in a cylinder-on-cylinder constraint, x would depend upon both the angle between the axes and the linear separation distance between axes. Let F_C be the force that the voxmap pointshell method computes for all point-voxel collisions that occur in the region of the geometric constraint. The voxmap pointshell method distinguishes such collisions from other collisions by consulting BREP information stored in points and voxels. Let F_A be the force that the voxmap pointshell method computes to represent the geometric constraint using a linear spring model. The net force and torque can be calculated and applied as follows (Eqns. 13 and 14):

$$\vec{F}_{Net} = \vec{F}_C * (1-x) + \vec{F}_A * x \qquad \textbf{(13)}$$

$$\vec{\tau}_{Net} = \vec{\tau}_C * (1-x) + \vec{\tau}_A * x \qquad \textbf{(14)}$$

When the geometric constraint is not at all engaged (x=0), the net force is F_C, and when the geometric constraint becomes fully engaged (x=1), the net force is F_A. It is easy to create a scenario where both the voxmap pointshell and constraint forces must be engaged to prevent the user from intersecting with parts. Fig. 25 shows a bolt that is being inserted into a cylindrical hole. The flat bolt surface also touches a cylindrical “stop” preventing the bolt to sit flush against the hole.

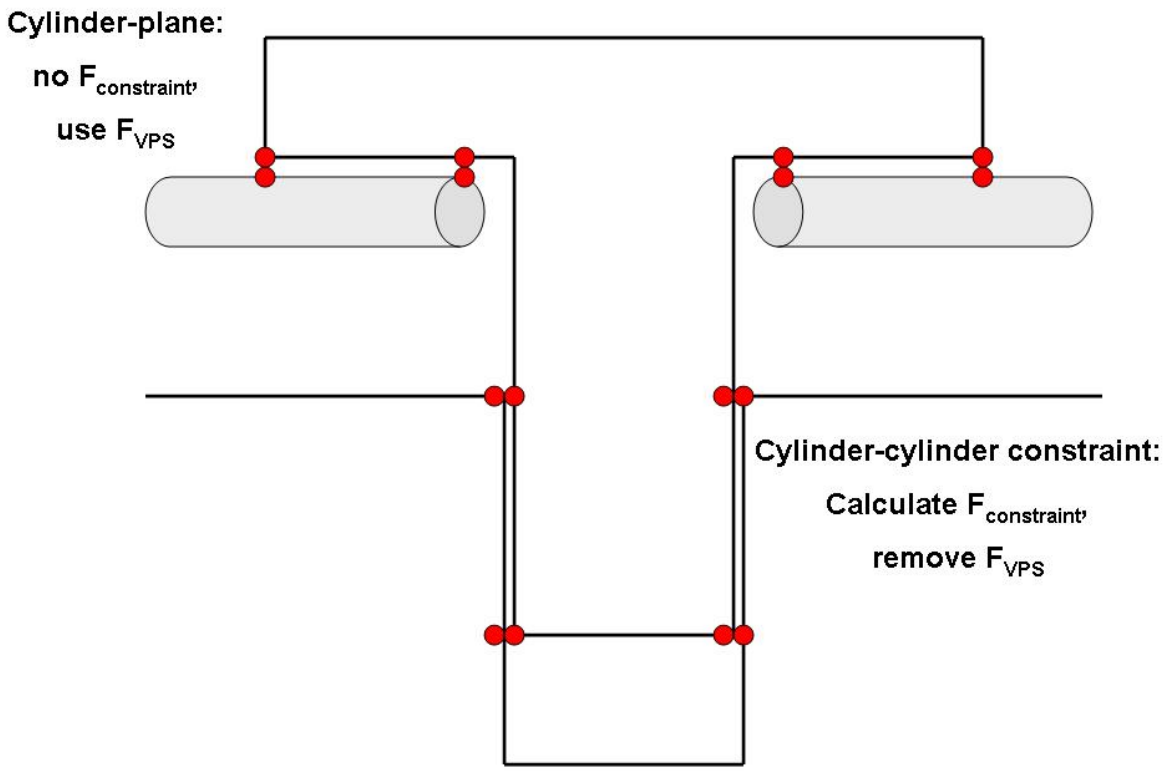

Figure 25: Sample Part Insertion Where Force Blending is Critical

In the case of the bolt being inserted into the hole, a cylinder-on-cylinder constraint can be applied. An alignment torque must be calculated so that the axes are aligned. An alignment force will pull the axes into congruency. At the same time, collision based forces must still be calculated in order to prevent parts from intersecting when no geometric constraints can be found during those voxel collisions (Fig. 25).

Fig. 25 shows a peg-hole mate modeled as a two-dimensional mate. The diameter of the hole and the diameter of the peg diameter are known. The page approaches along a path with some initial lateral and angular offset. Because of this initial offset, the peg must

both rotate and align in order to mate with the hole. θ, the angular offset, can be defined as the angle between the axis of alignment and the manipulated object. For example, during a concentric constraint, θ forms the angle between the axis of the hole and the axis of the peg. θ can be easily calculated using the alignment matrix and the overall body transformation matrix. When θ is very large, F_C should be the primary force acting on the object. When θ is very small, F_A should be only be acting on the part and keep the part aligned with its particular geometric constraint. The distance offset, d, can be defined as the difference in position between the axis of the hole and the axis of the peg. This difference can be calculated using the alignment matrix and the initial body transformation position.

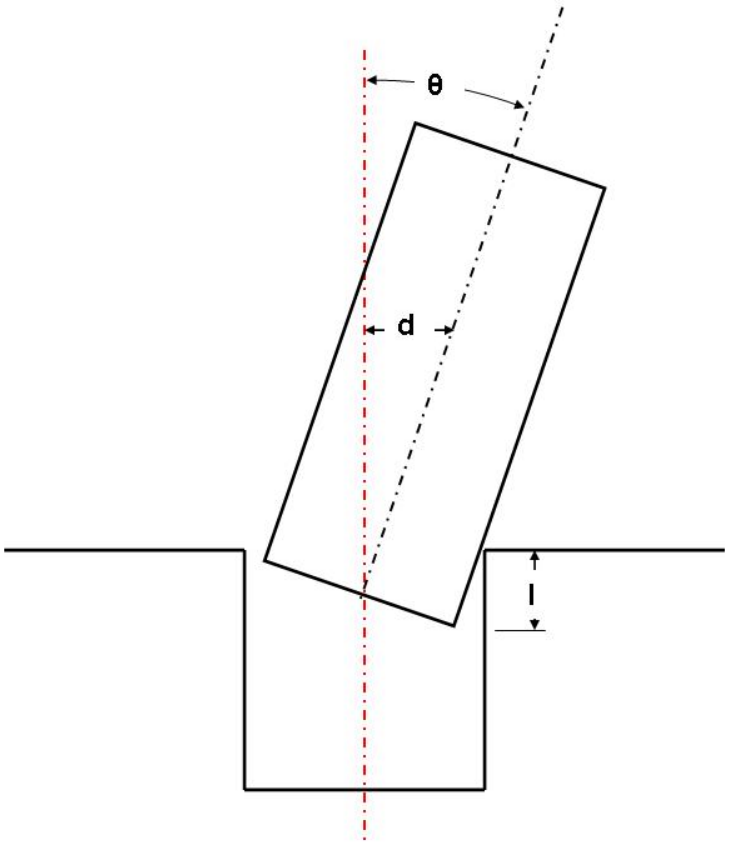

Figure 26: Peg-Hole Mate

In addition, an insertion depth, l, is defined as the insertion depth. The voxel pointshell method can return a list of locations of all point-voxel collisions. Only those collisions that are known to participate in the geometric constraint (based on private data) are used to calculate the current pin depth. For that subset, maximum and minimum values for x, y, and z are determined independently. The coordinate having the largest difference is adopted as a conservative estimate of current insertion depth. Such definitions could subsequently be fine-tuned for haptic quality.

Currently, the force blending factor, x, is parameterized from 0 to 1. Force blending relies on how fully the geometric constraint is engaged. 0 means that the geometric constraint is not at all engaged and 1 means a fully engaged geometric constraint. In addition, a cylinder-on-cylinder constraint has angular and distance displacements. Therefore, the force blending factor must be dependent on this angular and distance displacement as shown in Fig. 26.

$$x = 0.5 * e^{(-\theta/\theta_0)} + 0.5 * e^{(-d/d_0)} \quad \textbf{(15)}$$

where θ_0 and d_0 are defined as the following:

$$\theta_0 = voxelsize / l \quad \textbf{(16)}$$

$$d_0 = voxelsize \quad \textbf{(17)}$$

Currently, both angular and distance components have the same weight in determining the force blending factor. It is not clear at this point, if different weights would provide better results for low clearance scenarios. In order to calculate d, provided you know a pivot point which is preserved under rotation by both matrices. For the hybrid

method, the "pivot" is chosen to be the centroid of the model. This data is available in the BREP structure.

Furthermore, θ_0 and d_0 are chosen so that the resulting force blending factor, x, produces a smooth response surface. Fig. 27 shows the resulting blending factor. When θ and d are very small and approach 0, the resulting blending factor is 1. However, when θ and d are large, the resulting blending factor approaches 0. Adjustments to θ_0 and d_0 can be made. However, at this point, the resulting blending factor results in a smooth curvature and therefore would prevent any large changes in the blending forces to occur.

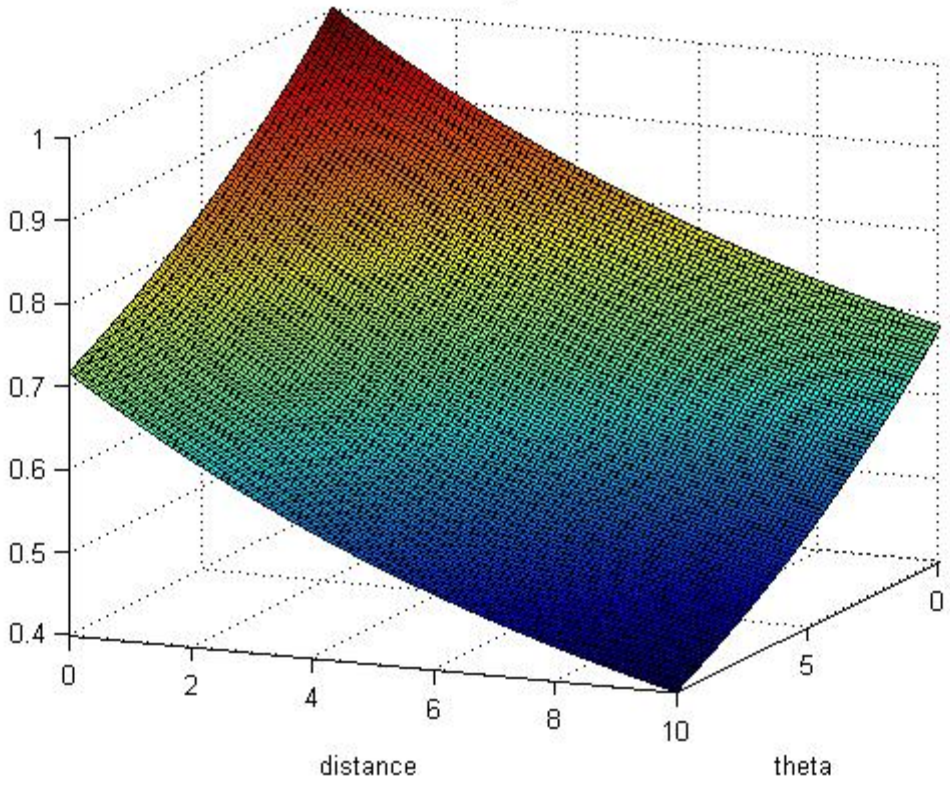

Figure 27: Response Surface of Force Blending Factor

Fig. 27 shows the response surface for the blending factor when θ_0 and d_0 are plotted. A smooth response surface will produce less abrupt changes in the force blending factor and therefore less abrupt changes in the resulting alignment forces.

The alignment force is a simple spring force, just like the user's virtual coupler. As such it will contain logically contain a stiffness coefficient, call it k_A. Before force blending, therefore, the alignment force is defined as:

$$\vec{F}_0 = k_A * \vec{d} \qquad \textbf{(18)}$$

where F_0 is the alignment force before force blending, d is the alignment distance and k_A is a spring constant chosen arbitrarily. A stiffer spring constant would produce a higher F_A, which could potentially prevent the user from getting out of geometric constraints. The stiffness constant k_A is therefore chosen to be similar to the spring constant used in the spring force calculation. During force blending, F_A is multiplied by x. Therefore, under force blending the constraint force becomes:

$$\vec{F}_1 = K_{alignment} * \vec{d} * (0.5 * e^{(-\theta/\theta_0)} + 0.5 * e^{(-d/d_0)}) \qquad \textbf{(19)}$$

where F_1 is the alignment force after force blending. The effective stiffness can then be calculated under force blending, by taking the partial derivative of the above expression with respect to d. That result then needs to be adjusted by adjusting the alignment stiffness, k_A, so as to always be less than the stiffness of the user's virtual coupler. This ensures that the user always remains "stronger" than the geometric constraint. A similar discussion applies to constraint torque.

The user (acting through the spring-damper virtual coupling) must be "stronger" than the alignment force. This prevents the user from being involuntarily pulled into the geometric constraint and remaining stuck there. Otherwise, the alignment force would be simulating a powerful magnet. However, the geometric constraint must be powerful enough to be effective. In addition, the haptic operation must be stable, meaning that the net stiffness must not exceed the device's rated maximum virtual stiffness.

For example, a peg is inserted into a hole. The hybrid method can identify that both colliding surfaces are cylindrical and a concentric constraint can be applied to the parts. Once a cylinder-on-cylinder constraint is fully engaged, the part requires only alignment forces and torques to stay aligned, but no collision forces. If collision forces and torques were calculated based on the voxmap pointshell method, it would prevent the bolt from being inserted (as was the problem with previous approaches that did not combine geometric constraint based assembly with the voxmap pointshell method). The plane-on-cylinder collision between the bolt flat surface and a cylindrical stop does not result in a geometric constraint. The collision forces and torques must be calculated for this face-to-face collision, otherwise, otherwise there would be no force or torque preventing the bolt to pass through the cylindrical stop.

CHAPTER 6. IMPLEMENTATION

6.1 Hardware

Providing multiple types of VR systems would allow smaller companies to take part in VR-based design. Large systems such as multi-sided CAVEs are expensive and require a team of experienced administrators. A low cost approach to virtual assembly simulation would give an engineer the opportunity to use an immersive system every day, thus truly integrating VR in the design process. Collaborative work can also be encouraged through VR systems.

This research presents a methodology that can be used on a wide variety of display systems, such as single pipe, single projection walls or multiple pipe projection environments such as the C6. The C6 is a three-dimensional, full-immersion, synthetic environment, newly renovated in 2007. This facility consists of a 10 ft. x 10 ft. room where all four walls, the floor and the ceiling are projection screens that are capable of displaying back-projected stereoscopic images. The six projection surfaces each display 4096x4096 pixels, making it the highest resolution CAVE system in the world, with over 100 million pixels of total resolution.

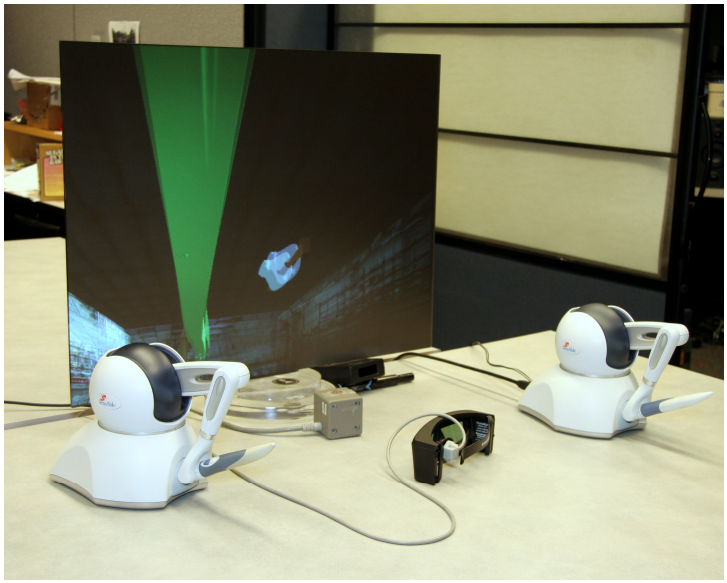

Figure 28: Desktop VR Setup with 2 Phantom Omnis

The application runs on a Windows workstation, but can be ported to a Linux system (Fig. 28). Currently, only 32 bit libraries are available for this hybrid method, but 64 bit libraries should soon be available from third party software companies. The Windows machine (64 bit operating systems) consists of 4 3.2 GHz Intel Xeon processors with 6GB of RAM. An Nvidia Quadra FX 5800 graphics card with 4GB graphics memory is used. A magnetic tracking system (Polhemus Patriot) tracks the user's head to provide the system with the user's position. Crystal Eye shutter glasses from StereoGraphics Corporation provide stereo viewing. The user wears the shutter glasses, which are synchronized with the computer display to alternate the right and the left eye views at a rate of 120 Hz to produce stereo images of the virtual environment.

6.2. Haptic Devices

The application supports PHANTOM haptic devices from Sensable Technologies. The OpenHaptics Toolkit library is used to enable haptic devices. The OpenHaptics Toolkit works with a variety of haptic devices from Sensable. The PHANTOM Omni was chosen for its compact footprint and cost-efficiency. The Omni communicates with the computer using an IEEE-1394 FireWire port.

6.3 Software

The test bed will be developed using C++ as the programming language. The VR Juggler open source software toolkit [3, 75] will be used for controlling the virtual environment. VR Juggler provides an application interface that supports a wide variety of display devices. By selecting different configuration files, the application can be run on a desktop monitor, one wall projection screen, multiple wall projection screens or a head mounted display [76, 77]. Sensable Technologies OpenHaptics Toolkit will provide device control for the PHANTOM Omni. OpenSceneGraph (OSG) will be used for rendering graphics and visualization. The voxmap pointshell method will be used for collision detection and haptic rendering, while D-Cubed from Siemens will be used to handle the BREP data and the geometric constraint recognition (Fig. 29).

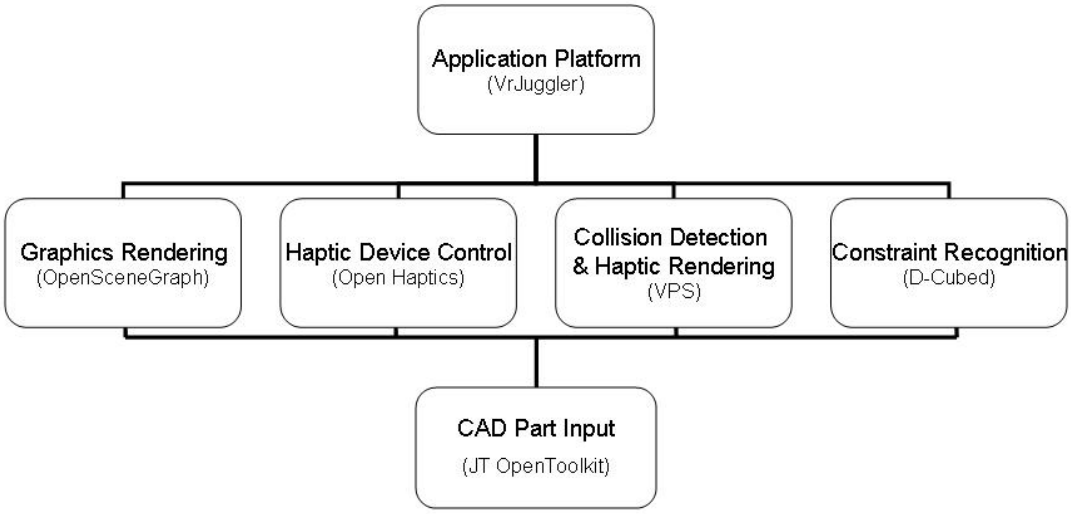

Figure 29: Application Libraries

JT files contain both triangulated and boundary representation models (amongst other data). JtOpenToolKit is used to access this information. For the hybrid method, the JT file is traversed and triangle strip data is accessed and converted to triangles for the voxelization process. In addition, the triangle data is also fed into a module that visualizes and saves them as OpenSceneGraph files. The boundary data is also accessed during the JT file traversal and a corresponding Parasolid file is saved. Due to the large amount of data that a boundary representation file contains, it was not possible to keep this data structure in memory. The BREP data is read into the method once the voxelization process is complete and enough memory has been freed up.

In order to tie the boundary data to the voxel, this method performs collision checks with the boundary voxel based on the voxel size and location. This process occurs after the BREP data has been added to the scene. A Parasolid cube is created for each object loaded in the scene. The dimensions of this Parasolid cube are that of the object's voxel size. During

a traversal of the voxelized object, each voxel can be accessed. This access allows the hybrid method to grab positional info of the voxel and use this info to move the Parasolid cube. The Parasolid cube is then collided with the corresponding boundary representation of the objects. During this collision detection, the hybrid method determines which faces and edges the Parasolid cube was colliding against and stores this information in a Look-Up-Table.

D-Cubed is a wrapper that is used to implement the geometric constraint recognition algorithm. The wrapper allows further access to the boundary data and can be used to determine whether a given face belongs to a cylindrical or planar face. The geometric constraint algorithm then can determine the appropriate geometric constraints.

A Look-Up-Table is an array that holds a set of pre-computed results for a given operation. This array provides access to the results in a way that is faster than computing the result of the given operation each time. This Look-Up-Table can be expressed as a function of an integer value. This integer value is then saved into the voxel's private data. The private data is accessible during the voxmap pointshell collision detection. The Look-Up-Table for the colliding voxels, containing face and edge data, can then be used to determine whether geometric constraints can be applied using the geometric constraint recognition algorithm.

The voxmap pointshell method is used to calculate collision and constraint forces. OpenHaptics is used to communicate with the haptic device and render the forces appropriately to the user. While Omni devices only transfer 3DOF Haptics, the hybrid method calculates torques as well. Use of a 6DOF haptic device is possible with this method.

6.4 Application Structure

6.4.1 Initialization

Previously, CAD software such as SolidEdge and Pro/Engineer were used to save the geometry of a part as a BREP model (*.x_t) as well as a graphics file (*.osg). In addition, a tessellated model (*.stl or *.itri) was saved, which was then converted to the voxelized haptic model. The use of *.jt files makes this conversion step unnecessary and allows the user to specify only one file instead of two or more input files. JT OpenToolKit is used to traverse the model and grab tessellated data from the JT file directly. In addition, the BREP model is also accessed and stored. During the JT traversal, the tessellated data is loaded, and element and normal information is read and stored. The element data is then converted to a VPS voxmap (*.vps). The tessellated data is also used to visualize the given geometry with OpenSceneGraph. The initialization of the application is completed when all three models are bound to each other.

Binding of the BREP data occurs during voxel traversal when all boundary objects are loaded in the scene. This process may require some time since each voxel in the scene used to bind the BREP data. For scenes with very large numbers of voxels, this process may take several minutes.

The application is now ready for interaction with the user. The actions of grabbing, moving, or colliding parts are monitored by the application and the corresponding haptic feedback is calculated by the voxmap pointshell method. When the user is ready to

assemble the parts and requires low clearance parts to fit together, the Automatic Constraint Recognition (ACR) module is activated through the menu. The user can then use geometric constraint recognition to finish the assembly. During the ACR activation, D-Cubed software determines if geometric constraints can be applied to parts and calculates a new transformation matrix. The voxmap pointshell method collision loop is never stopped.

6.4.2 Simulation Loops

This methodology is written as a multi-threaded application. There are several simulation loops in this methodology that require a high level of interaction. The graphics loop updates the graphics and monitors inputs from the mouse, keyboard or a wand. The haptic loop is responsible for computing collision forces and rendering them back to the haptic device. It also reads the device position data and switch state. The physics loop checks if parts are colliding. The hand collision loop detects if the hand pointer is touching a part and requires movement of the part. The geometric constraint loop checks what BREPs are involved in the collision and performs a geometric constraint recognition algorithm to determine if any geometric constraints can be applied to the colliding parts.

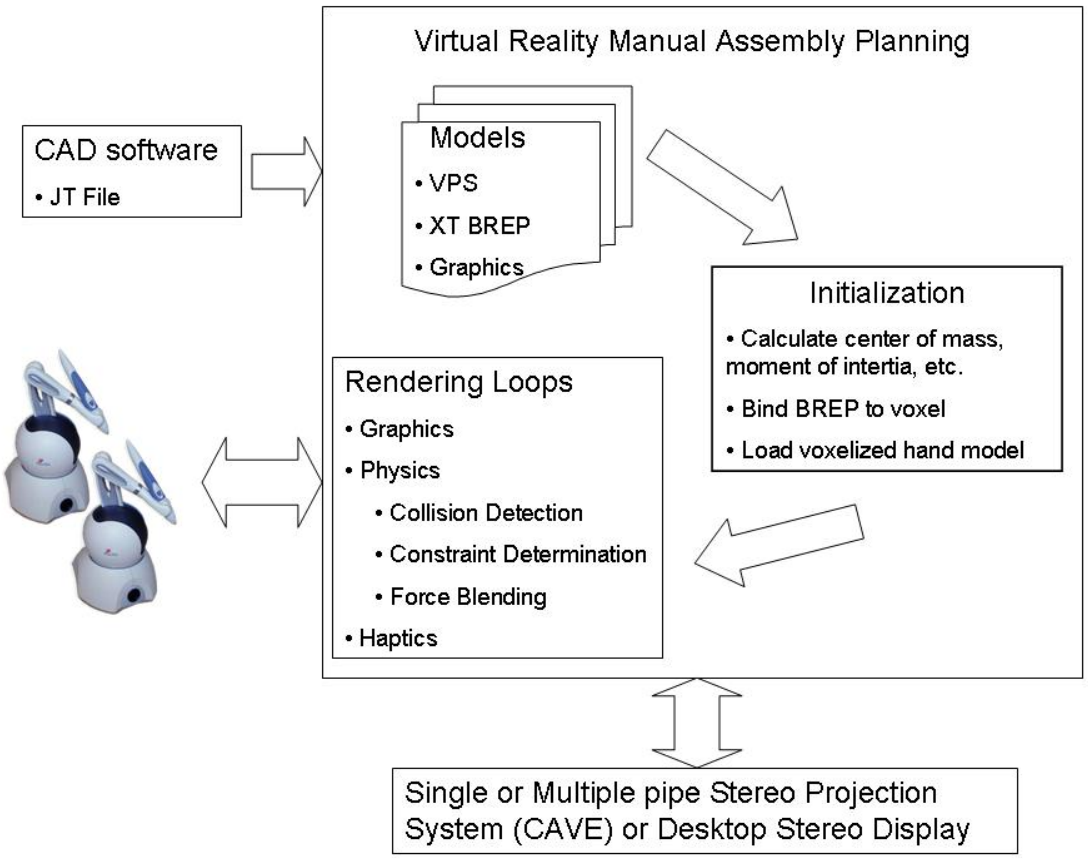

Figure 30: Simulation Loops

Fig. 30 shows the different simulation loops for the hybrid method and the required model inputs.

6.4.3 Application Flowchart

The application reads a text-based configuration file that includes the models being loaded and state flags. The user can specify the initial position of the model and the scale and size for each voxel. The voxmap pointshell method then voxelizes the tessellated model files in preparation for calculating collisions. The graphics models are also stored in a scene

graph structure. Once the initialization is complete, the methodology uses the simulation loops to monitor the part interactions (Fig. 31).

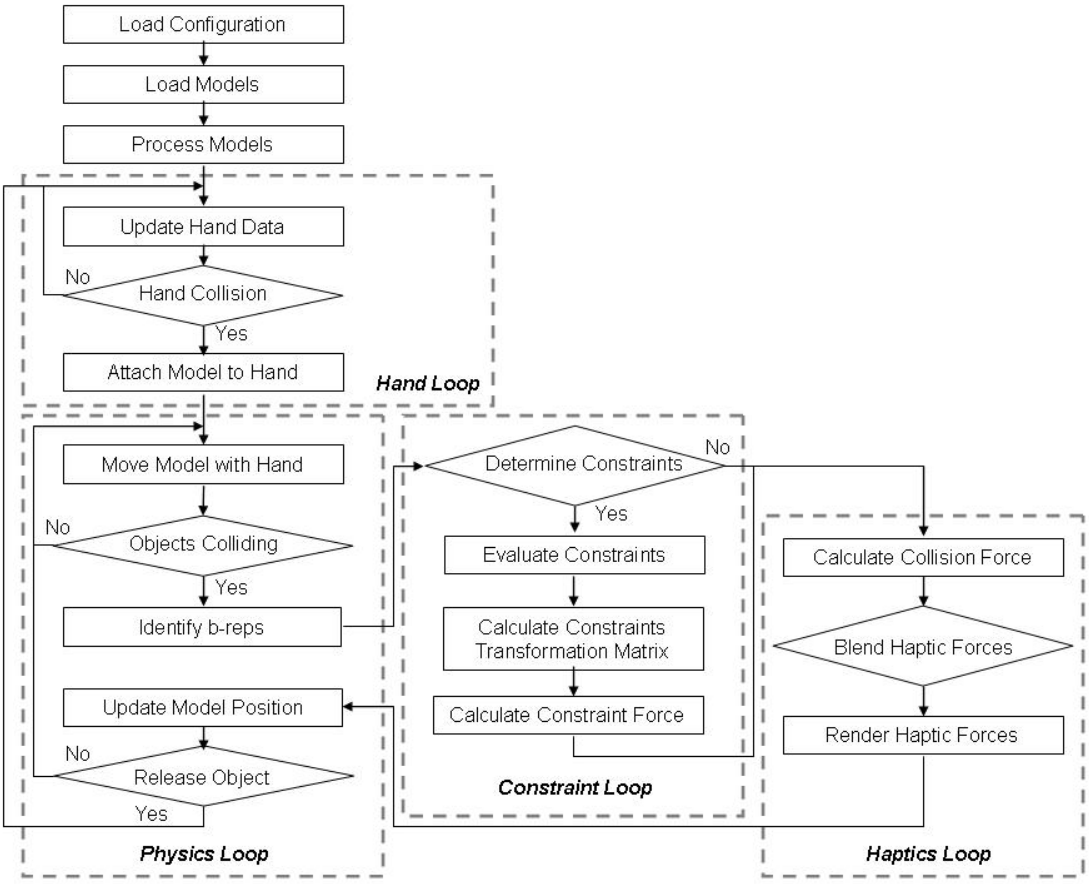

Figure 31: Application Flowchart

The hand loop is responsible for the interaction between the user and the part. If the user moves the hand pointer over a part, the hand loop will detect that the user has selected a part and attaches the part to the hand. The user can now move the model with the hand. After this action by the user, the physics loop now controls the interactions. If a collision occurs, pointers to BREP faces and edges are used to determine plausible geometric constraints. If geometric constraints can be determined, geometric constraint

based forces are calculated and the model position is updated accordingly. The haptic loop is responsible for calculating and rendering forces to the haptic device. When the user is satisfied with the new part position, the parts are released and the user is returned to the initial hand loop.

6.5 Application Features

Direct import of CAD geometry into the application is a key feature of this methodology. Reducing the steps required to transfer CAD data into VR-compatible models will allow engineers to use VR technology more efficiently. Additional pre-processing of the CAD data requires more time in the product design process and may also lead to distortion of the original CAD model. Most engineering companies use standards in their CAD software and compatibility between the CAD software and this methodology is desired. As described before, three geometry models are used for this application. The collision model is a tessellated model that most CAD software supports (*.jt). The tessellated model is parsed and the voxmap pointshell method uses the triangle and normal information to generate a voxel representation for the collision detection module. A graphics file has to be exported, but OpenSceneGraph accepts several generic graphics models.

An advantage to using VR Juggler is that the runtime configuration environment can be adapted easily to run on a typical desktop machine or a CAVE. In addition, the assembly environment can be changed using a configuration file. The number of models and location for each model can be changed for each model in the virtual environment. The application

also allows the user to specify the voxel size for each model. Smaller parts that require low clearance assembly can be specified to have smaller voxels, whereas larger models used for fixturing may have larger ones. Due to the nature of the hybrid approach to virtual assembly, some of the application modules can be turned on or off. For example, if the user desires to run the application without geometric constraint recognition, a flag in the Visual Studio preprocessor can be turned off and the application will run only voxel-based collision detection without geometric constraint recognition. The same application executable file can therefore be used to run different modules, and can be used to interact with varying models using the configuration file.

Dual haptic interfaces have been explored previously with the voxmap pointshell method. Interacting with two hands in the virtual environment improves the realism of the application. The user will be able to perform assembly tasks with both hands, receiving force feedback using the same natural motions and dexterity as in the real world. Two parts can be manipulated simultaneously. This may reduce the assembly steps a user is required to undertake while replicating real world interactions. However, dual haptic interfaces drop the haptic refresh rate considerably and may reduce the performance of the methodology even further.

Runtime voxel size variation has been discussed in previous chapters. In addition to shrinking/expanding the voxels during the geometric constraint recognition sequence, the user will also be able to change the voxel size “manually”. During the initialization process, a voxel size is assigned to a part. Without the runtime module, the user would not be able to change the voxel size while using the application. This module will enable the user to

change the voxel size when the initial voxel size is not optimized for the assembly task and can override the initial size. The user will be able to select a part and the voxel size can then be increased or decreased using menu options. The model is then parsed with the new voxel size using the tessellated model. All simulation loops are stopped while the model is voxelized. Changing the voxel size does not require the application to be stopped and restarted while the user in the process of assembling parts. Future work could include selective voxelization of certain areas of interest within a part to provide even smoother transition for the hybrid switch.

Subassemblies are an integral part of an assembly planning sequence. It is very common that subassemblies consist of several parts or several subassemblies. When two or more parts are selected and grouped into a subassembly, the VPS, D-Cubed and graphic scene graphs need to be rearranged so that the subassembly represents a single entity in the virtual environment. Once the user is satisfied with the final position of parts, the user will then be able to select the group of objects, which in turn are treated as a single object by the application. The voxmap pointshell module must be restarted to calculate the mass, center of mass and other properties of the new subassembly. Once the internal calculations are complete, the user will be able to move and interact with the new subassembly like every other part in the environment.

Although the implementation of these features is not crucial to this framework, they will, however, give the user the ability to plan a more realistic assembly sequence. The use of dual-handed haptic has been shown to create a more realistic approach to assembly

simulations. The exploration of a dual-handed haptic in a CAVE would be a direct extension of this research.

CHAPTER 7. POINTSHELL SHRINKING ASSESSMENT

This chapter investigates the effect of pointshell shrinking and feature size on manual assembly operations in a virtual environment with haptic force feedback. Specifically, this research examines the use of pointshell shrinking and the effect of feature size in voxel-based modeling for haptic rendering as a way to improve manual assembly of low clearance parts. CAD parts were created, voxelized and tested for assembly. The results showed that pointshell shrinking allows the engineer to assemble parts with a lower clearance than without pointshell shrinking. Further results showed that assemble-ability is dependent on part diameter and clearance. As the diameter increases, assembling low clearance features becomes difficult. An empirical equation is developed to guide the designer in selecting an appropriate voxel size based on feature size. These results advance the effort to improve manual assembly operations via haptic feedback in the virtual environment.

In this chapter, the voxmap pointshell method is examined to determine if shrinking the pointshell aids in assembling low clearance parts. Previous research using voxel-based methods has been unsuccessful in achieving low clearance assembly. Furthermore, selecting an appropriate voxel size and clearance has been an iterative process. This paper examines the pointshell shrinking method and its ability to improve the assembly process.

7.1. Experimental setup

7.1.1 Software

The test bed was developed using C++ as the programming language. The VR Juggler open source software toolkit was used for controlling the virtual environment. VR Juggler provides an application interface that supports a wide variety of display devices. OpenGL Performer (PFB) will be used for rendering graphics and visualization. VPS™ will be used for collision detection and haptic rendering.

7.1.2. Hardware

The application runs on a Windows workstation. The Window machine consists of dual 3.6 GHz Intel Xeon processors with 3GB of RAM. A PCI Express Nvidia Quadro 4400 graphics card with 512 MB graphics memory is used. A magnetic tracking system (Polhemus Patriot [78]) tracks the user's head to provide the system with the user's head position. CrystalEye® shutter glasses provide stereo viewing (Fig. 32).

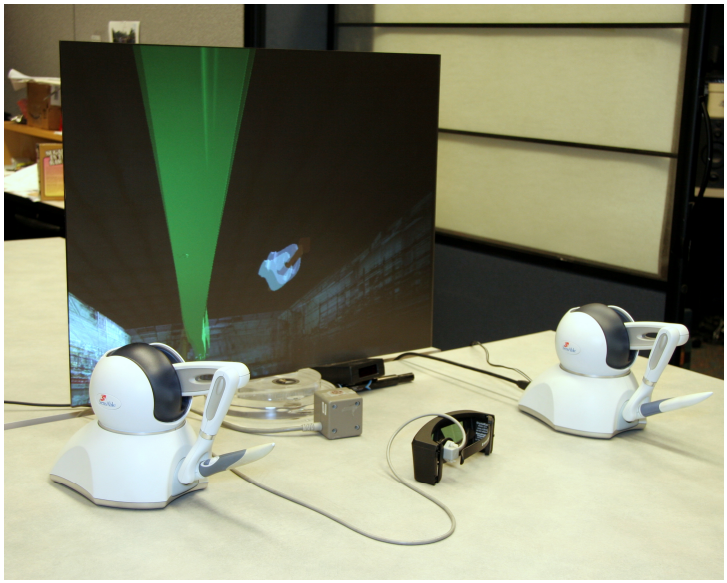

Figure 32: Hardware Setup

Two PHANTOM Omni haptic devices from Sensable Technologies are using in this setup (Fig. 34). The Omni was chosen for its compact footprint and cost-efficiency. The Omni communicates with the computer using an IEEE-1394 FireWire port.

7.2. Procedure

There are two studies presented in this paper. One examines the effect of pointshell shrinking on assemble-ability and the other determines whether assemble-ability was dependent on feature size. For example, a typical assembly scenario is a peg inserted into a whole. When the peg radius increases, more opportunities arise for the peg to become stuck in the hole, because the voxel-topology-based surface normals are not strictly perpendicular to the peg's surface, and so they "catch" on the hole's surface voxels (under the point-voxel force model) and create small resistive forces that cause the peg to stick. If

the peg radius is sufficiently small, then the user's applied force (under the virtual coupling model) can still overpower the accumulated resistive forces, but if the peg radius is too large, then resistive forces sum to a value that the user can no longer overpower.

7.2.1. Pointshell shrinking affect on assemble-ability study

Two CAD parts, one with a peg feature and one with a hole are modeled in SolidEdge. The nominal diameter of the hole is 18.75 mm. The models being used are shown in Fig. 33.

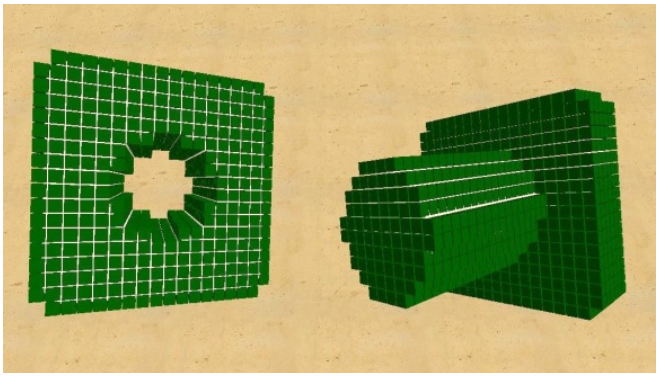

Figure 33: Peg and Hole

A standard ASCII *.stl file is prepared as the basis for voxelization. During the voxelization step, the triangular polygon and surface normal vector information is read from the .stl file and converted into a voxmap. Fig. 33 shows the pin part and the block with hole feature after voxelization.

Table 1 summarizes the experimental design. The voxel size of the pin part and the voxel size of the hole part are varied. The nominal dimensional clearance, defined as the difference in diameters, of the models is also varied. This is accomplished by keeping the

diameter of the hole part constant and changing the diameter of the pin during the study. The amount of pointshell shrinking is also varied.

Table 1: Experimental Setup for Pointshell Shrinking

Factor	Range	# of levels
Peg voxel size (mm)	0.25 -> 2.0	10
Hole voxel size (mm)	0.25 -> 2.0	10
Clearance (mm)	2.5, 1.4, 1.0	3
Pointshell shrinking	0 ->1.0	11

JMP, statistical software, was used to create the design of experiment. There were 27 trials. Within the range for each factor, experimental values were chosen at equal intervals (Table 1). This experiment has a multi-factorial design to test the assemble-ability of a peg part and hole part based. The design varies both parts' voxel sizes, clearance and pointshell shrinkage.

The task is to try to assemble the parts together. An operator is initially presented with two parts as displayed in the virtual environment. Full assembly is determined as the state of assembly when the pin is fully inserted into the hole part. The assembly results for each assembly trial were recorded and analyzed. If the parts can be assembled, the operator records a "1" and if they cannot be assembled the operator records a "0". If the peg was able to be partially inserted, the result was recorded as "0". The operator was not limited by trial time and therefore, the trial time was not recorded. In general it was estimated that it took less than two minutes to determine if parts could be assembled. All tests were performed by the same operator.

7.2.2. Feature Size Affect on Assemble-ability Study

The same general methodology was used in the second study to determine if feature size affects assemble-ability. The goal is to develop an empirical relationship between voxel size, clearance and assemble-ability to give guidance to the engineer when picking suitable voxel sizes for mating parts. The same base CAD models were used with different dimensions. In this study, the peg and hole diameters were varied (Table 2). There were two distinct voxel sizes used in the study: 2.0 mm and 0.4 mm. The pointshell shrinkage was set to 0.5 for all trials. There were a total of 62 trials.

Table 2: Experimental Setup for Feature Size

Factor	Values
Peg diameter (mm)	15, 35, 55, 75, 95
Peg voxel size (mm)	2.0, 0.4
Hole diameter (mm)	varying
Hole voxel size (mm)	2.0, 0.4
Pointshell shrinking	0.5

The peg diameter was varied at five levels. The hole diameter was chosen randomly. The results for each assembly test are recorded and analyzed. Again, if the user was able to insert the peg into the hole, it was denoted with a “1”, while not being able to insert the peg was denoted with a “0”. All tests were performed by the same operator.

7.3. Results

Analysis of variance statistics (ANOVA) were performed to analyze the results. A probability, p, was computed where p is the probability that difference is due to chance

factors. χ^2 , or chi-squared, distribution of independent standard normal random variables. An F value was determined for the voxel size prediction formula. A significance level of $p \leq 0.05$ was selected for all experiments.

In this experiment, the pointshell shrinkage was varied from 0 to 100% to determine what effect pointshell shrinking has on the assemble-ability of the peg and hole. The results were analyzed using an ordinal logistic model. The factors were specified as numerical continuous and the result of each trial was an ordinal data type. The p-value for observed significance of a one-tailed t-test was recorded (Table 3).

Table 3: ANOVA Table for Pointshell Shrinkage

Source	χ^2	p
Peg voxel size (mm)	0.598	0.4395
Hole voxel size (mm)	0.869	0.3513
Pointshell shrinking	3.209	0.0732
Clearance	4.646	0.0311

Clearance is statistically significant in determining if parts can be assembled (χ^2 4.646, $p<0.0311$). Interestingly, the pointshell offset percentage is weakly significant as well (χ^2 3.209, $p<0.0732$).

Pointshell shrinking at 100% always produced the best results: allowing assembly at the lowest clearance. However, a 100% value for pointshell shrinking has the potential to produce spatial inversion. This is because each individual point of the pointshell is originally located at the center of the voxel. A 100% pointshell shrinking value would result in moving the point a distance of one voxel dimension along the surface normal. This movement places the point outside of its own voxel boundary. Due to the risk of spatial inversion,

McNeely recommends that realistically, pointshell shrinkage should not exceed 50% of voxel size. Spatial inversion leads to erroneous collision forces and torques. While it is possible to achieve tighter clearances with the aid of pointshell shrinking, it is a methodology that should not be overused to obtain better fits.

The experimental data from the feature size affect was used to formulate a prediction formula for the voxel size to be used for the voxelization of the CAD parts. In this case, the voxel size for both the peg and the hole were the same value. The results of the ANOVA are shown in Table 4. There were a total of 33 trials. A standard least squares fit model was chosen to construct model effects.

Table 4: Analysis of Variance

Source	Degree of Freedom	Sum of Squares	Mean Square	F Ratio
Model	2	0.112	0.056	21.250
Error	15	0.040	0.003	
Total	17	0.154		

The effects are found to be not significant (F 21.250), the differences between the means were equally effective (Table 4). Table 5 shows the sum of squares, F and p values for the peg and hole diameter effects.

Table 5: Table for Effect Tests

Source	Degree of Freedom	Sum of Squares	F	p
Peg diameter (mm)	1	0.080	29.989	<0.001
Hole diameter (mm)	1	0.088	33.139	<0.001

The p values for peg (F 29.989, $p<0.001$) and hole diameter (F 33.139, $p<0.001$) are highly significant. This indicates that assemble-ability depends on peg and hole diameter

and voxel size. In order to create a fit for this formula, only results from assembled parts were used. Both the diameter of the hole and the peg have a significant influence on the assembly experiment. Analysis of variance (F 21.25 $p<0.001$) shows that the fit of the voxel size prediction is significant. The summary of fit reveals that the model reveals that the R^2 value is 73.91%, which means that the errors between the actual and the predicted response are significant and this formula cannot predict with a high enough certainty a voxel size that will actually allow assembly. The voxel size prediction formula follows in Eqn. 20.

$$voxelsize = 0.027 * (D_{Hole} - D_{Peg}) - 0.001 \qquad \textbf{(Eq. 20)}$$

The difference between the hole and the peg diameter is defined as clearance. Rearranging, the equations becomes:

$$voxelsize = 0.027 * C - 0.001 \qquad \textbf{(Eq. 21)}$$

where C is the clearance. Solving for clearance, the clearance must be larger than 0.037 for part assembly. However, the R^2 value suggests that the fit for this equation is not very good and the clearance might be larger or smaller once a higher R^2 value is achieved through more testing.

With a more accurate prediction expression, it should be much easier for the engineer to calculate an appropriate voxel size and reduce the amount of trial and error that was previously required to ensure that parts would be able to fit together. This

prediction equation could be improved upon by refining the voxel sizes and running more experiments. This would create a much better fit for determining the optimal voxel size.

7.4. Conclusion

In conclusion, this chapter improves prediction of a voxel size that will support assembly. This prediction takes a given hole diameter and a given peg diameter into account. Engineers choose peg and hole diameters based on the fits they desire. This paper allows the engineer to determine a voxel size that would meet assemble-ability requirements.

Low clearance manual assembly is very important to understanding how parts fit together and the ways that a human will handle parts. Pointshell shrinking aids in assembly of low clearance parts. However, any pointshell shrinking beyond 50% of voxel size runs the risk of spatial inversion. This is a well-known limitation of the pointshell-shrinking technique.

Smaller voxels would allow tighter clearances to be assembled. However, the number of voxels is limited by the size of computer memory and the speed of the CPU. The operating system requires some memory and so does the voxelization process itself. A typical 32 bit Windows operating system (OS) can address up to 3GB of memory (minus any additional memory required by other programs). Using a 64 bit operating system would provide access to greater amounts of memory and allow us to increase the number of voxels.

Future work includes exploration of voxelization techniques that improves the representation the CAD geometry. In addition, better approximation of surface normals would result in higher accuracy for the collision models. Currently, surface normals associated with each point of the pointshell are calculated from voxel topology, and therefore they are only approximately perpendicular to the underlying surface. Methods of obtaining surface normals based on boundary representations of the CAD models will be explored in the future to improve accuracy. Manual assembly simulations with haptic feedback can reduce the need for expensive and repetitive prototyping if successful methodologies with low clearance collision detections emerge.

CHAPTER 8. RESULTS

The goal of this research is to combine haptic feedback with geometric constraint-guided assembly. The research will explore and develop methods to combine voxmap pointshell method-based haptic feedback with BREP geometric constraints to allow realistic part interaction. Direct CAD input, low cost and precise collision detection are emphasized to produce a virtual assembly simulation that is capable of assembling low clearance parts. Performance enhancing procedures may be required to optimize the haptic refresh rate. Real-time applications such as VR require that all components be optimized to maintain high refresh rates. System robustness is another key component to produce an assembly methodology that is able to handle multiple models without reducing its refresh rates.

This hybrid method uses a seamless integration of geometric constraints and physics-based behavior. The requirements for this method were:

- No geometric constraint metadata from CAD systems
- On-the-fly geometric constraint recognitions using BREP data
- Fast and robust operation
- No or minimal user intervention

The hybrid method only considers geometric constraints between circular faces and edges and planar faces. All other geometric constraints are ignored by this algorithm. If geometric constraints are possible, the algorithm queries the face and edge data from the BREP geometry, which is tied directly to the colliding voxels. If two planar faces are colliding

and the angle between their surface normals is less than a certain value, a coplanar constraint (or coincident alignment) is defined. If the colliding voxels are part of a circular face or edge and the difference in diameter between the two geometries and the angle between their axes is below a certain threshold value, a concentric constraint is defined. Once the geometric constraints are defined, a validity check is performed. This check determines if a solution is possible and calculates a new transformation matrix for the colliding geometries.

8.1 Evaluation of Hybrid Method

To evaluate the effect of the hybrid approach on system performance, this section presents experimental results of assembly tasks involving several different scenarios. First, the system performance was tested without the use of the hybrid method algorithm. The system was then tested with the use hybrid method algorithm. The assembly steps using a single handed haptic interface will be as follows:

Step 1: Grab the block model – position and orient it.

Step 2: Release the block model.

Step 3: Grab the peg and try to orient and insert it into the stationary block model.

Step 4: Perform Step 2 – 4 if necessary.

Using dual handed haptic interaction the user can manipulate both parts simultaneously, orienting them with respect to each other to complete assembly. However, for the sake of improved performance with the automatic geometric constraint recognition algorithm, the

dual handed configuration was not used. Two evaluations are performed on the hybrid method. First, the hybrid method is tested and compared to the voxmap pointshell method in terms of being able to assemble parts. Secondly, haptic performance and refresh rates are examined under the hybrid and the voxmap pointshell method without force blending or BREP geometric constraint recognition. It is critical that a refresh rate of 500 to 1000 Hz is maintained; otherwise convincing haptic feedback is compromised.

8.2 Low-Clearance Assembly Evaluation of Hybrid Method

The following section describes the evaluation of the hybrid method with the voxmap pointshell method. The voxmap pointshell method will not use any force blending or pointshell shrinking to improve its low-clearance assembly. A pin and hole assembly is considered for the low clearance testing. In this case, a peg and a hole are considered. Both CAD parts are modeled in SolidEdge and exported as *.JT files. The diameter was chosen Table 5 describes the CAD model geometries.

Table 6: CAD Model Statistics

Model	Diameter (m)	Voxel Size (m)	Number of Voxels
Block	0.70	0.01	81,448
High Clearance Peg	0.65	0.01	32,868
Low Clearance Peg	0.69	0.01	35,304

8.2.1 Case I: Voxmap PointShell Method Only

Previous implementations of the voxmap pointshell method are limited because they require a large enough clearance between parts due to model approximation with voxels.

The first case featured a clearance between the peg and hole of 0.05m. The primary factor in determining whether parts can be assembled is the voxel size. In both test cases, a voxel size of 0.01m was chosen. In the first test case, the clearance was 0.05m. Since both models used a voxel size of 0.01m, insertion of the peg into the block was not difficult. The second test case used a clearance of 0.01 with both models using a voxel size of 0.01. Assembly was not possible because colliding voxels prevented the peg from insertion into the block. As discussed in the previous chapter, assembly depends on the voxel size and the chosen clearance between parts. The voxmap pointshell method

8.2.2 Case II: Hybrid Method

The second test case describes using the hybrid method to assemble the peg and the hole. During this test case, the automatic geometric constraint recognition was used to determine which voxels are in a possible geometric constraint. The force blending algorithm then reduced any forces and torques from the voxmap pointshell collisions and used an alignment force and torque to aid in the assembly process by pulling the peg into axis alignment. The high clearance scenario presents no challenge to the hybrid method and both parts can be assembled successfully. The low clearance scenario could not be completed with just the voxmap pointshell method.

To test the hybrid method, the constraint evaluation was turned off. All voxels in collision were still checked if constraints are possible, but no geometric constraint evaluation was done. Using the hybrid method without geometric constraint evaluation, the

collision forces of voxels in geometric constraints are reduced enough to allow both parts to be assembled.

Secondly, the geometric constraint recognition was turned on within the hybrid method. The peg was not able to be inserted into the hole due to an unsafe drop in the haptic refresh rate. Previous approaches using automatic geometric constraint recognition have noted that the refresh rates drop significantly during the evaluation of geometric constraints. For the hybrid method, each voxel in collision is tested if a geometric constraint can be applied and the geometric constraint is then evaluation. The geometric constraint algorithm is only active when collisions occur unlike other applications where geometric constraints are always checked for. The actual evaluation of the geometric constraints is only done every 10^{th} frame to prevent the frame rate from dropping during the automatic geometric constraint recognition algorithm. The haptic frame rate will be discussed in the next section.

Figure 34: Peg and Hole Assembly with Hybrid Method– no Constraint Evaluation

Fig. 34 shows the peg inserted into the block using the hybrid method without geometric constraint recognition. Because the voxmap pointshell method is the underlying method for the collision detection, there are a high number of penetrations between interior voxels occurring. These penetrations unfortunately produce collision forces, which are used to prevent any parts from ever penetrating another part. Collisions between surface voxels are reduced with the hybrid method, collisions forces from interior voxel penetrations can be reduced through the hybrid method. The hybrid approach allows parts with lower clearances to be assembled, but suffers from low haptic refresh rates during geometric constraint evaluation.

8.5 Refresh Rate Evaluation of Hybrid Method

A key component to haptic assembly methods is the need for very high refresh rates. While 500 to 1000 Hz is acceptable for haptic refresh rates, any significant drop in refresh rates prevents realistic force feedback to the user. If the haptic loop does not perform at the required rate, the consequences are severe. Perceptual lapses such as jumps in the graphic loop are undesired. Low haptic update rates can induce haptic instability such as possible jolting and buzzing of the haptic interface.

The assembly scenario is as follows. The peg is moved in the virtual environment by the user and collided with the stationary block. The peg collides with the planar face of the block and collisions occur. During the hybrid method with constraint evaluation, no

constraints are detected. Next, the peg is repositioned. The peg is then inserted into the hole (see Fig. 35).

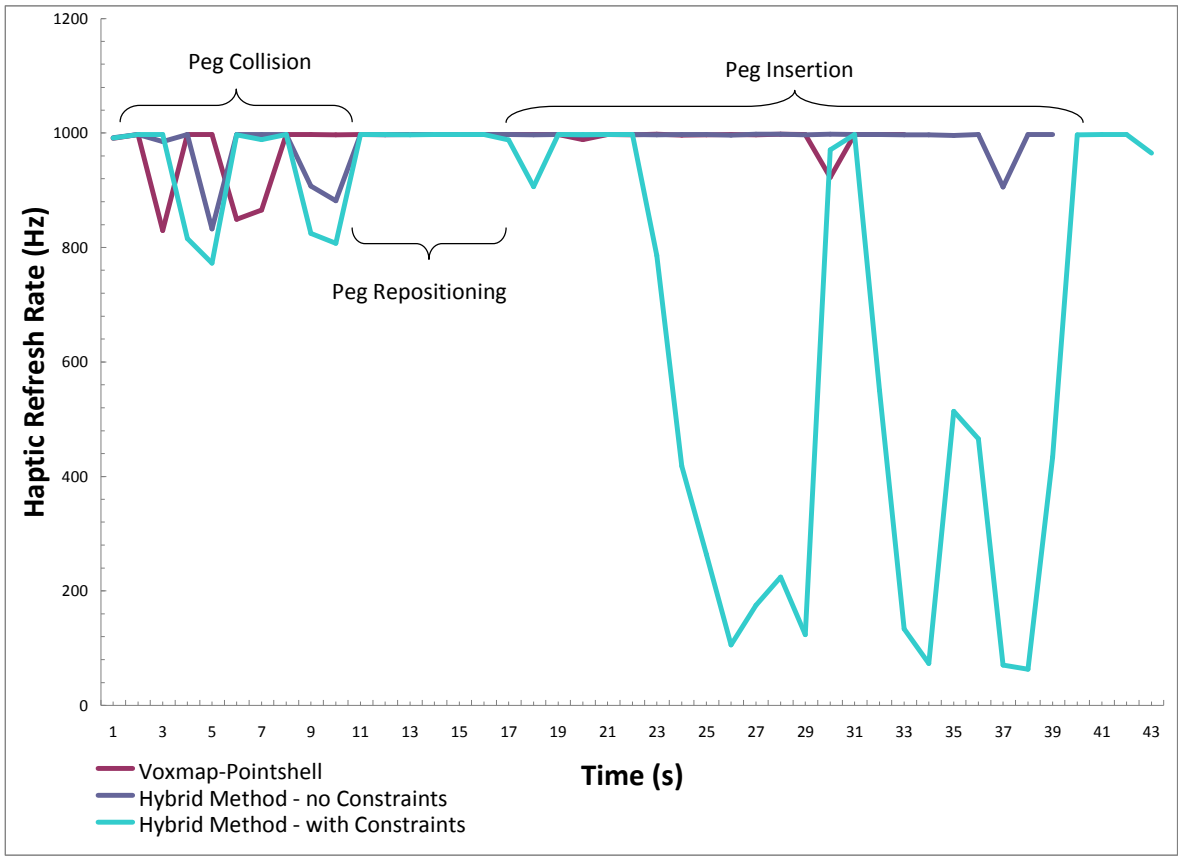

Figure 35: Haptic Refresh Rate Comparison

Fig. 35 compares the three different test cases. For all three cases, only the low clearance scenario is used. In the first case, only the voxmap pointshell method is used. The assembly could not be completed. However, the haptic refresh rate was always above 800 Hz. This is not a problem as long as the refresh rate does not drop below 500 Hz to prevent haptic instabilities (too large time steps can create too large differentials in object velocity, collisions, etc). The hybrid method without geometric constraint

recognition was tested. Again, the haptic refresh rates did not drop below 800 Hz. Lastly, the hybrid method with geometric constraint recognition was tested. The haptic refresh rate of this test dropped to 100 Hz when the automatic constraint recognition evaluates the geometric constraints. This is very problematic as 100 Hz is not acceptable for haptic scenarios.

8.4 Summary

The above test results show that the hybrid method combining highly accurate BREP geometric constraint recognition and very fast voxel-based collision detection led to a distinct improvement in this method's performance for low clearance scenarios over similar research efforts. The hybrid method presents a novel approach by combining inexact voxel-based collision detection with highly accurate BREP data to allow low clearance of CAD parts. Geometric constraints are used to determine which voxel's collision forces are reduced. This allows collision forces to be calculated and applied to voxels to prevent parts from penetrating each other.

The user feels no magnetic force or torque pulling the object into the final assembly position. Instead, the hybrid method uses the constraint forces and torques to calculate a new position of the object using collision and constraint forces and torques. The user still feels the same forces as the voxmap pointshell method by itself. The hybrid method aims to provide seamless integrating of collisions and constraint recognition. The constraint recognition

However, the evaluation of geometric constraints produced a drop in the haptic refresh rate that prevents low clearance assembly. Reducing the geometric constraint evaluation will improve the haptic refresh rate, but will ultimately lead to inaccurate geometric constraint forces and torques. Overlaying the BREP onto the voxel does allow exact geometry to be retrieved during collisions and used to

The hybrid method joins both voxel-based and BREP models concurrently to support collision detection with constraint recognition and haptic rendering. Currently, the geometric constraint recognition is a bottleneck that is difficult to overcome.

CHAPTER 9. CONCLUSION

Haptic force feedback requires frame rates of 1000 Hz to render forces smoothly in a virtual environment. Most test bed applications for assembly simulation will provide accurate collision detection, but may not have any haptic feedback. On the other hand, applications which have haptic feedback require high frame rates to allow smooth haptic rendering, but are unable to assemble low clearance parts because of inexact model approximations. This research will culminate in a methodology that will allow engineers to perform assembly simulations with low clearance parts while simultaneously providing haptic feedback. During low clearance assembly situations, geometric constraint based guidance will be provided to the user to aid in the assembly process. Haptic rendering adds an additional sense to the VR environment. As Chen notes, haptic devices can improve realism, allow faster product development cycles, and promote better designs through the use of a six DOF force feedback device [79].

Currently, this hybrid method only considers one geometric constraint. Allowing multiple constraints to be applied to the geometric entities is not difficult and should not require more computation time. Assemblies typically require more than one geometric constraint to be considered fully constrained. Currently, the constraint evaluation drops the haptic refresh rate, which is highly problematic and reduces usability of this approach. Investigations into implementing a different geometric constraint solver or other methods to calculate constraint positions are required.

This hybrid approach was developed under a 32-bit operation system. Using a 64-bit operating system would allow the user to voxelized the models with more voxels. Large, complex scenes may also require 64-bit operation. While the application has been written to support 64-bit operation, not all libraries that are pulled into the project have a 64-bit compilation.

This research only considers rigid bodies. Dynamically linked or articulated mechanisms have not been considered in this hybrid approach. Developing a method to be able to insert two pegs joined by a link that needs to be inserted into two holes may require a more parallelized geometric constraint checking to allow for multiple geometric constraints to occur. In addition, a method needs to be developed to allow non-rigid bodies to be compatible with the voxmap pointshell method. Non-rigid bodies have not been considered under the voxmap pointshell method and would require extending the physically-based modeling to include links, dampers and other non-rigid connections at the individual part level.

VR applications allow direct manipulation of parts and can enhance collaboration between teams. This research will allow manipulation of 3D CAD models within the VR environment, furthering human computer interaction by linking haptic feedback models and BREP models. Verification of this research will be done through cooperation with Deere & Company. Engineers will be able to test the application and an extension to user studies can be easily made to verify the effectiveness of the hybrid approach. This hybrid methodology allows engineers to have haptic feedback while performing tight assembly

operation. Automatic geometric constraint recognition will aid in the assembly process and provide guidance when parts are close to each other.

Although VR assembly scenarios will never become as realistic as the shop floor, they will certainly aid in the discovery of better designs for manufacturing and assembly. Providing better methodologies to handle complicated scenarios will benefit not only the engineering design community, but also medical training, where doctors train on virtual patients to perform intricate surgeries. This methodology provides a new way to interact with models and improve product design cycle efficiency. Assembly processes imply the need for 6DOF haptic rendering. This research's goal was to support low clearance assembly with haptic force rendering and to minimize manual pre-processing required for the data structure.

BIBLIOGRAPHY

[1] Sherman, W. R., and Craig, A. B., 2003, *Understanding Virtual Reality*, Elsevier, San Fransisco, CA.

[2] Jayaram, S., Vance, J.M., Gadh, R., Jayaram, U. And Srinivasan, H, 2001, "Assessment of VR Technology and Its Applications to Engineering Problems," Journal of Computing and Information Science in Engineering, 1(1), pp. 72-83.

[3] Vrjuggler, 2007, Open Source Virtual Reality Tools., http://www.vrjuggler.org

[4] Antonya, C., and Talaba, D., 2006, "Design Evaluation and Modification of Mechanical Systems in Virtual Environments," Virtual Reality, (11) pp. 275-285.

[5] Chipperfield, K., Vance, J. M., and Fischer, A., 2006, "Fast Meshless Reanalysis Using Combined Approximations, Preconditioned Conjugate Gradient, and Taylor Series," AIAA Journal, 44(6), pp. 1325-1331.

[6] Ye, N., Banerjee, P., Banerjee, A., and Dech, F., 1999, "A Comparative Study of Virtual Assembly Planning in Traditional and Virtual Environments," IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Review, 29(4), pp. 546 - 555.

[7] Heger, R., and Betsos, G., 1996, "Virtual Assembly Planning System for Complex Assembly Operations- a Practical Concurrent Engineering Application," eds., pp.

[8] Ramaswamy, S., and Yan, Y., 1999, "Interactive Modeling and Simulation of Virtual Manufacturing Assemblies: An Agent-Based Approach," Journal of Intelligent Manufacturing, 10(pp. 503-518.

[9] Gomes, A., and Zachmann, G., 1999, "Virtual Reality as a Tool for Verification of Assembly and Maintenance Processes," Computers and Graphics, (23) pp. 189-403.

[10] Kuehne, R., and Oliver, J., 1995, "A Virtual Environment for Interactive Assembly Planning and Evaluation," *Proceedings of ASME Design Automation Conference*, DE-Vol. 83(2), pp. 863-867.

[11] Pere, E., Langrana, N., Gomez, D., and Burdea, G., 1996, "Virtual Mechanical Assembly on a PC-Based System," DFM-1306, *Proceedings of the ASME Design Engineering Technical Conference and Computers in Engineering Conference*, Irvine, California, J. M. McCarthy (eds.), ASME, New York.

[12] Bullinger, H. J., Richer, M., and Seidel, K.-A., 2000, "Virtual Assembly Planning," Human Factors and Ergonomics in Manufacturing, 10(3), pp. pp. 331-341.

[13] Lippman, R.; Rößler, A.: "Planungsverfahren Zur Ergonomischen Gestaltung Der Arbeit," REFA Nachrichten, pp. 12-19.

[14] Gupta, R., and Zeltzer, D., "Prototyping and Design for Assembly Analysis Using Multimodal Virtual Environments," *Proceedings of ASME Design Automation Conference, 1995.*

[15] Gupta, R., Whitney, D., and Zeltzer, D., 1997, "Prototyping and Design for Assembly Analysis Using Multimodal Virtual Environments," Computer Aided Design, 29(8), pp. 585-597.

[16] Coutee, A. S., Mcdermott, S. D., and Bras, B., 2001, "A Haptic Assembly and Disassembly Simulation Environment and Associated Computational Load Optimization

Techniques," ASME Transactions - Journal of Computing & Information Science in Engineering, 1(2), pp. 113-122.

[17] Coutee, A. S., and, Bras, B., 2002, "Collision Detection for Virtual Objects in a Haptic Assembly and Disassembly Simulation Environment," *presented at ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (DETC2002/CIE-34385), Montreal, Canada.

[18] Kim, C. E., and Vance, J.M., 2004, "Collision Detection and Part Interaction Modeling to Facilitate Immersive Virtual Assembly Methods," ASME Journal of Computing and Information Sciences in Engineering, 4(1), pp. 83-90.

[19] Johnson, T. C., and, Vance, J. M., 2001, "The Use of the Voxmap Pointshell Method of Collision Detection in Virtual Assembly Methods Planning," *presented at ASME Design Engineering Technical Conferences & Computers in Information Engineering* (DETC2001/DAC-21137), Pittsburgh, PA.

[20] Kim, C. E., and Vance, J.M., 2004, "Development of a Networked Haptic Environment in VR to Facilitate Collaborative Design Using Voxmap Pointshell (VPS) Software," DETC2004/CIE-57648, *Proceedings of ASME Design Automation Conference,* Salt Lake City, UT.

[21] Garbaya, S., and Zaldivar-Colado, U., 2007, "The Affect of Contact Force Sensations on User Performance in Virtual Assembly Tasks," Virtual Reality, 11(4), pp. 287-299.

[22] Jayaram, S., Connacher, H. I., and Lyons K.W., 1997, "Virtual Assembly Using Virtual Reality Techniques," Computer Aided Design, 29(8), pp. 575-584.

[23] Jayaram, S., Jayaram, U., Wang, Y., Tirumali, H., Lyons, K. And, Hart, P., 1999, "Vade: A Virtual Assembly Design Environment," Computer Graphics and Applications, 19(6), pp. 44-50.

[24] Jayaram, S., Jayaram, U., Wang, Y., and Lyons, K., 2000, "Corba-Based Collaboration in a Virtual Assembly Design Environment," DETC 2000/CIE-14585, *Proceedings of ASME Design Engineering Technical Conferences 2000*, Baltimore, MD.

[25] Jayaram, U., Tirumali, H. And, Jayaram, S., 2000, "A Tool/Part/Human Interaction Model for Assembly in Virtual Environments," (DETC2000/CIE-14584) *Proceedings of ASME Design Engineering Technical Conferences 2000*, Baltimore, MD.

[26] Taylor, F., Jayaram, S. And, Jayaram, U., 2000, "Functionality to Facilitate Assembly of Heavy Machines in a Virtual Environment," DETC 2000/CIE-14590, *Proceedings of ASME Design Engineering Technical Conferences 2000*, Baltimore, MD.

[27] Jayaram, S., Joshi, H., Jayaram, U., Kim, Y., Kate, H., and Varoz, L., 2006, "Embedding Haptic-Enabled Tools in Cad for Training Applications," presented at ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2006-99656), Philadelphia, PA.

[28] Shaikh, I., Jayaram, U., Jayaram, S., and Lyons, K., 2004, "Participatory Ergonomics Using Vr Integrated with Analysis Tools," eds., Washington, DC, pp.

[29] Fernando, T., Wimalaratne, P., and Tan, K., 1999, "Constraint-Based Virtual Environment for Supporting Assembly and Maintainability Tasks," *presented at ASME Computers in Engineering International Conference*, Las Vegas, NV.

[30] Marcelino, L., Murray, N., and, Fernando, T., 2003, "A Constraint Manager to Support Virtual Maintainability," Computers & Graphics, 27(1), pp. 19 - 26.

[31] Wan, H., Gao, S., Peng, Q., Dai, G and Zhang, F., 2004, "Mivas: A Multi-Modal Immersive Virtual Assembly System," *presented at ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Salt Lake City, UT.

[32] Gottschalk, S., and Lin, M. C., 1996, "Obbtree: A Hierarchical Structure for Rapid Interference Detection," eds., pp. 171 - 180.

[33] Chen, X., Xu, N., and Li, Y., 2005, "A Virtual Environment for Collaborative Assembly," *Proceedings of the Second International Conference on Embedded Software and Systems*, Xian, China.

[34] Seth, A., Su, H.-J., Vance, J. M., "Development of a dual-handed haptic assembly system: SHARP", *ASME Journal of Computing and Information Sciences in Engineering*, 8(4), 2008, pp. 044502, 8 pages.

[35] Iacob, R., 2008, "Contac Identification for Assembly Disassembly Simulation with a Haptic Device," Visual Computer, pp. 1-8.

[36] Wang, Y., Jayaram, S., Jayaram, U., and Lyons, K., 2001, "Physically Based Modeling in Virtual Assembly," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburg, PA.

[37] Tching, L., Dumont, G., and Perret, J., 2009, "Interactive Simulation of Cad Models Assemblies Using Virtual Constraint Guidance," IJIDEM, pp.

[38] Seth, A., Vance, J. M., and Oliver, J. H., 2010, "Virtual Reality for Assembly Methods Prototyping: A Review," Virtual Reality Journal, in press.

[39] Mortensen, M. E., 1985, *Geometric Modeling*, Wiley, New York.

[40] Stroud, I., 2006, *Boundary Representation Modelling Techniques*, Springer,

[41] Liu, X., Dodds, G., Mccartney, J., and Hinds, B. K., 2004, "Virtual Designworks- Designing 3d Cad Models Via Haptic Interaction," Computer-Aided Design, (36) pp. 1129-1140.

[42] Lin, M. C., and Otaduy, M. A., 2008, *Haptic Rendering: Foundations, Algorithms, and Applications*, A.K. Peters, Wellesley, MA.

[43] Zilles, C. B., and Salisburg, J. K., 1995, "A Constraint-Based God-Object Method for Haptic Display," eds., 3, pp. 146-151.

[44] Nelson, D., and Cohen, E., 2000, "Optimization-Based Virtual Surface Contact Manipulation at Force Control Rates," *Presented at the 2000 ASME International Mechanical Engineering Congress and Exposition*, Orlando, FL, pp. 37-44.

[45] Gregory, A., Lin, M. C., Gottschalk, S., and Taylor, R., 1999, "A Framework for Fast and Accurate Collision Detection for Haptic Interaction," *Presented at the IEEE VR Conference*, Houston, TX, USA, pp. 38-45.

[46] Redon, S., Kheddar, A., and Coquillart, S., 2002, "Fast Continuous Collision Detection between Rigid Bodies," EUROGRAPHICS, 21(3).

[47] Gregory, A., Mascarenhas, A., Ehman, S., Lin, M., and Manocha, D., 2000, "Six Degree-of-Freedom Haptic Display of Polygonal Models," eds., Salt Lake City, Utah, United States, pp. 139-146.

[48] Gregory, A., Masarenhas, A., Ehmann, S., Lin, M., and Manocha, D., 2000, "Six-Degrees-of-Freedom Haptic Display of Polygonal Models," eds., pp. 139-146.

[49] Ho, C.-H., Basdogan, C., and Srinivasan, M. A., 1999, "Efficient Point-Based Rendering Techniques for Haptic Display of Virtual Objects," Presence, (8) pp. 477-491.

[50] Srinivasan, M. A., and Basdogan, C., 1997, "Haptics in Virtual Environment: Taxonomy, Research Status, and Challenges," Comput. & Graphics, 21(4), pp. 393-404.

[51] Cuisenaire, O., 1999, "Distance Transformations: Fast Algorithms and Applications to Medical Image Processing," Ph.D. thesis, Universite Catholique de Louvain, France.

[52] He, T., and Kaufman, A., 1997, "Collision Detection for Volumetric Objects," eds., Phoenix, Arizona, United States, pp. 27 - 37.

[53] Kaufman, A., Cohen, D., and Yagle, R., 1993, "Volume Graphics," IEEE Computer, 26(7), pp. 51-64.

[54] Koenig, H., and Strothotte, T., 2002, "Fast Collision Detection for Haptic Displays Using Polygonal Models," *Proceedings of the Conference on Simulation and Visualization*, Ghent, Belgium, pp. 289--300.

[55] Gregory, A., Lin, M. C., Gottschalk, S., and Taylor, R., 2000, "Fast and Accurate Collision Detection for Haptic Interaction Using a Three Degree-of-Freedom Force-Feedback Device," Computational Geometry: Theory and Applications, 15(1-3), pp. 69-89.

[56] Colgate, J. E., and Brown, J. M., 1994, "Factors Affecting the Z-Width of a Haptic Display," *Proceedings of the IEEE 1994 International Conference on Robotics & Automation*, San Diego, CA, pp. 3205-3210.

[57] Burdea, G. C., 2000, "Haptics Issues in Virtual Environments," *Proceedings of Computer Graphics International*, Geneva, Switzerland, pp. 295 - 302.

[58] Massie, T., 1993, "Design of a Three Degree of Freedom Force-Reflecting Haptic Interface," Ph.D. thesis, Massachusetts Institute of Technology, USA.

[59] Massie, T., and Salisbury, K., 1994, "The Phantom Haptic Interface: A Device for Probing Virtual Objects," *Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems,* Chicago, IL.

[60] Salisbury, J. K., and Srinivasan, M. A., 1997, "Phantom-Based Haptic Interaction with Virtual Objects," IEEE Computer Graphics and Applications, 17(5), pp. 6-10.

[61] Hinckley, K., Pausch, R., Proffitt, D., and Kassell, N. F., 1998, "Two-Handed Virtual Manipulation," ACM Transactions on Computer-Human Interaction, 5(3), pp. 260-302.

[62] Guiard, Y., 1987, "Asymetric Division of Labor in Human Skilled Bimanual Action," Journal of Motor Behaviour, 19(4), pp. 487-517.

[63] Buxton, W., and Myers, B., 1986, "A Study in Two-Handed Input," Proceedings of CHI, CHI April 1986), pp. 321-326.

[64] Seth, A., Su, H.-J., and Vance, J. M., 2008, "Development of a Dual-Handend Haptic Assembly System: Sharp," Journal of Computing and Information Science in Engineering, 8(4), p. 044502, 8 pages.

[65] “PHANToM Haptic Device Implemented in a Projection Screen Virtual Environment” Immersive Projection Technology and Virtual Environments 2003 (Eurographics/Fraunhofer IAO Workshop Proceedings), pp. 225-230.

[66] Wang, M., and Mcneely, W., 2003, "Quasi-Static Approximation for 6 Degrees-of-Freedom Haptic Rendering," IEEE Visualization, Washington, DC, pp. 34.

[67] Mcneely, W., Puterbaugh, K. D., and Troy, J. J., 2005, "Advances in Voxel-Based 6-DOF Haptic Rendering," SIGGRAPH '05: *ACM SIGGRAPH 2005 Courses: SESSION: Recent advances in haptic rendering & applications.*

[68] Kim, C. E., and Vance, J.M., 2003, "Using VPS (Voxmap Pointshell) as the Basis for Interaction in a Virtual Assembly Environment," DETC2003/CIE-48297, *ASME Design Engineering Technical Conferences,* Chicago, IL.

[69] Mcneely, W., Puterbaugh, K. D., and Troy, J. J., 2005, "Advances in Voxel-Based 6-DOF Haptic Rendering," SIGGRAPH '05: *ACM SIGGRAPH 2005 Courses: SESSION: Recent advances in haptic rendering & applications.*

[70] Mcneely, W., Puterbaugh, K. D., and Troy, J. J., 2006, "Voxel-Based 6-Dof Haptic Rendering Improvements," Haptics-e, 3(7), pp.

[71] Ugs, S., 2008,

[72] Hoffmann, C. M., 2001, "D-Cubed's Dimensional Constraint Manager," ASME Journal of Computing and Information Sciences in Engineering, 1(1), pp. 100-101.

[73] Seth, A., Su, H. J., and Vance, J. M., 2005, "A Desktop Networked Haptic VR Interface for Mechanical Assembly," IMECE2005-81873, *Proceedings of the ASME International Mechanical Engineering Congress and Exposition* 2005 Orlando, FL.

[74] Seth, A., Vance, J. M., Oliver, J. H., “Combining dynamic modeling with geometric constraint management to support low clearance virtual manual assembly tasks”, *ASME Journal of Mechanical Design*, in press 2010.

[75] Just, C., Bierbaum, A., Baker, A., and Cruz-Neira, C., 1998, "VR Juggler: A Framework for Virtual Reality Development," *Proceedings of the 2nd Immersive Projection Technology Workshop*. 1998. Ames, IA.

[76] Bierbaum, A., and Cruz-Neira, C., 2000, "Runtime Reconfiguration in VR Juggler," *Proceedings of the 4th Immersive Projection Technology Workshop,* June 2000, Ames, IA.

[77] Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., and Cruz-Neira, C., 2001, "VR Juggler: A Virtual Platform for Virtual Reality Application Development," *Proceedings of the Virtual Reality 2001 Conference*, Yokohama, Japan, pp. 89-97

[78] Polhemus, "Polhemus (Http://Www.Polhemus.Com/)," pp.

[79] Chen, E., 1999, "Six Degree-of-Freedom Haptic System for Desktop Virtual Prototyping Applications," *Proceedings of the First International Workshop on Virtual Reality and Prototyping*, Laval France.

ACKNLOWDEDGEMENTS

This dissertation is dedicated to my mom and my aunt, breast cancer survivors.

I would like to take this opportunity to express my thanks to Dr. Vance, for her guidance during my research. This research would not have been possible without her support, guidance and patience. My deepest gratitude is also due to Dr. Bill McNeely, who offered invaluable assistance about VPS and haptic rendering.

Dr. Jackie Shanks has been an incredible teaching mentor and I will miss our lunch conversations. I was very fortunate to have such a compassionate teacher and researcher as my teaching mentor. I would also like to thank my committee members for their efforts and contributions to this work: Dr. James Oliver, Dr. Eliot Winer, Dr. James Bernard and Dr. Stephen Gilbert.

In my daily work I have been blessed with a friendly and group of students. I have thouroughly enjoyed the intellectual stimulation only VRAC could provide. Ryan Pavlik deserves a special thanks as the resident computer guru. I would like to thank Catherine and Mike for furthering my addiction to coffeine and the numerous walks around campus.

Finally, I thank my parents and my family for supporting me throughout my studies. Without their help and emotional support, I would have never gotten this far.

BIOGRAPHICAL SKETCH

Daniela Faas received a Bachelor of Science in Mechanical Engineering and a Bachelor of Arts in International Relations from Bucknell University in 2005. She also received her Master of Science in Mechanical Engineering from Bucknell in 2006. For her Master's work, she focused on developing methods to reconstruct image data from embryonic chicken hearts and to create 3D models from the data. The left ventricle was of primary interest and a finite element model was developed to study the effect of pressure-underloading and pressure-overloading in the left ventricle. Daniela completed her Ph.D. in Mechanical Engineering and Human-Computer Interaction from Iowa State University in 2010. Her thesis developed a hybrid method to further manual assembly methods for low clearance part assembly with haptic force feedback. Her research interests include Virtual Reality applications, finite element analysis, design optimization, and biomedical engineering. She is currently a member of ASME, IEEE and SWE.