5-2010

Assessment of Pointshell Shrinking and Feature Size on Virtual Manual Assembly

Daniela Faas
Iowa State University, daniela.faas@gmail.com

Judy M. Vance
Iowa State University, jmvance@iastate.edu

Assessment of Pointshell Shrinking and Feature Size on Virtual Manual Assembly

Abstract
This paper investigates the effect of pointshell shrinking and feature size on manual assembly operations in a virtual environment with haptic force feedback. Specific emphasis is on exploring methods to improve voxel-based modeling to support 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 feature size, particularly part diameter and clearance. In a pin and hole assembly, as the pin diameter increases, for a given percent clearance, 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.

Keywords
Manufacturing, Shrinkage (Materials)

Disciplines
Mechanical Engineering

Proceedings of the ASME 2010 World Conference on Innovative Virtual Reality
WINVR2010
May 12-14, 2010, Ames, Iowa, USA

WINVR2010-3765

ASSESSMENT OF POINTSHELL SHRINKING AND FEATURE SIZE ON VIRTUAL MANUAL ASSEMBLY

Daniela Faas, Judy M. Vance[1]
Department of Mechanical Engineering
Virtual Reality Applications Center
Iowa State University
Ames Iowa 50011
[1]jmvance@iastate.edu

ABSTRACT

This paper investigates the effect of pointshell shrinking and feature size on manual assembly operations in a virtual environment with haptic force feedback. Specific emphasis is on exploring methods to improve voxel-based modeling to support 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 feature size, particularly part diameter and clearance. In a pin and hole assembly, as the pin diameter increases, for a given percent clearance, 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

Keywords: Virtual Assembly, Virtual Reality, Human Computer Interaction, Haptic Feedback, Mechanical Design.

INTRODUCTION

A major responsibility of the manufacturing engineer is the determination of assembly sequences and methods. Given a CAD assembly, the manufacturing engineer determines which parts are suitable for sub-assembly and the particular sequence of tasks for the entire assembly. The engineer also balances the work load of workers on the assembly line to avoid assembly bottlenecks.

Existing software tools to support these decisions are based on traditional computer interfaces such as the monitor and mouse. While these tools support some aspects of decision making, notably, they do not account for how humans interact with parts. For example, CAD software can be used to identify interference fits between mating parts, but that same software does not support human decision making when interacting with parts during assembly. Currently, physical prototypes are created to test ease of assembly and assess potential complications that may occur during production. Consequently, assembly errors resulting from human interaction are not apparent until late in the design process where changes to designs can be costly. Using VR to interact with and manipulate parts taps into human decision-making processes during the assembly and design stages and has greater potential to identify errors at that time.

VR evaluation with haptic feedback can reduce the need for prototyping by providing realistic part-to-part interactions that traditional desktop-based systems cannot provide. Haptic feedback provides additional sensory information to the user to indicate when parts collide through rendered force feedback.

In this paper, 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 primarily due to the approximate nature of the voxel model. Currently, selecting an appropriate voxel size is an iterative process that directly affects the approximate geometric model of the part. This paper examines the pointshell shrinking method and its ability to improve the assembly process and presents an approach to selecting an appropriate voxel size.

BACKGROUND

Today's VR is becoming a truly interactive 3D design environment. Assembly planning simulations have used VR to

allow the user to interact with 3D CAD models in a virtual environment. These applications support factory planning, maintenance, assembly planning, visualization, and ergonomic assessment, among other tasks. The following is an overview of research that has investigated issues with assembly operations in a virtual environment.

Kuehne and Oliver [1] 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. VSHOP was developed by Pere *et al.* [2] in 1996. Collision detection to avoid part interpenetration was based on bounding boxes. VSHOP was a PC-based system that integrated a force feedback device (Rutgers Master II).

Bullinger *et al.* [3] developed a method that created a script file containing the sequence of assembly actions. The method uses a Head-Mounted Display (HMD) for stereo viewing as well as data gloves for gesture recognition. VirtualANTHROPOS [4], which simulates the human body in a virtual environment, was used to view a virtual human during assembly tasks.

VEDA (Virtual Environment for Design for Assembly) by Gupta *et al.* [5, 6] relies on collision detection to model real physical behavior. These physics-based algorithms simulated part trajectories once collision occurred. VEDA used two SensAble PHANTOM haptic devices to interact with the models. However, the assembly process was limited to two-dimensional (2D) models. HIDRA (Haptic Integrated Dis/Re-assembly Analysis), which was developed by Coutee *et al.* [7], also supported haptic feedback. The user was able to grab models between two fingertips, but had only limited 3D manipulation ability.

The Virtual Environment for General Assembly (VEGAS), in addition to haptic feedback, supported data glove interaction and implemented physically-based modeling for parts using VPS™ (Voxmap PointShell™). 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. Dual-handed assembly operations as well as desktop configurations were also supported [8-10].

Out of this research, NHE (Network Haptic Environment) emerged to enable assembly tasks to be evaluated by individuals in geographically distinct locations [11]. 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.

Constraint-based applications render the interaction between parts in a simulated environment using constraints such as parallel or coplanar. One of the first applications that integrated geometric constraint-based modeling was VADE (Virtual Assembly Design Environment), developed in 1995 by Jayaram *et al.* [12-16]. Pro/Toolkit was used to import assembly data including geometric constraints and assembly hierarchy to simulate the assembly task.

The automotive industry has been receptive to integrating VR tools in their production planning. For example, Gomes *et al.* [17] developed a virtual assembly system in conjunction with BMW. The researchers used an HMD for the graphics display.

Seth *et al.* [18] developed a method for close tolerance assembly using a feature-based approach to geometric constraint recognition by taking advantage of dynamically contacting geometric features to aid in the assembly of low clearance parts. This approach to representing realistic model behavior is based on physical and geometric constraints. Although this methodology enables very accurate collision detection and allows feature-based automatic constraint recognition, it lacked haptic feedback.

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 [19].

MOTIVATION

Currently, software exists to support animation and planning of assembly and maintenance sequences using CAD models. These tools can identify interference of parts and awkward assembly sequences. However, they cannot identify problems that occur when assembly workers, maintenance workers, or users actually interact with the product. An immersive virtual environment provides an interface for interaction with CAD models before physical prototypes are built.

The key to achieving this capability is simulating realistic collision and force feedback to the user in the virtual environment. Complex CAD geometry and the high refresh rates required to adequately simulate haptic feedback present significant challenges.

In 2004, Kim and Vance examined several collision detection methods and found the voxmap-pointshell method provided both collision detection and force feedback calculations to support virtual assembly of complex CAD models [20, 21]. This method is a voxel-based method that models the static objects in the scene as collections of voxels and the moving objects in the scene as collections of points (comprising a pointshell). Collisions and force feedback must be calculated at a constant 1000 Hz refresh rate to provide realistic interaction with highly complex CAD parts [22].

Based on the results of the Kim paper, researchers at Iowa State University have been exploring the full potential of voxmap-pointshell methods. The main limitation to the use of

voxel-based methods is the inability to model low clearance assemblies. Because voxels approximate the actual model surface, collisions are indicated before the actual surfaces collide. The accuracy of the collision information is dependent on the voxel sizes of the objects. In the limit, infinitesimally small voxels would most accurately represent the model surface; however, as the size of the voxels decrease, the number of voxels increases, resulting in increased memory requirements and increased haptic calculation time. Determining the appropriate voxel size is generally done by trial and error, and based on experience, as the modeler explores the tradeoffs between smaller (more numerous) voxels and storage and performance requirements.

This paper describes a method to support low clearance haptically-enabled assembly of voxelized models. Two specific aspects are explored: modifying the approximate model by pointshell shrinking and developing heuristics to guide the modeler in selection of a voxel size.

VOXMAP-POINTSHELL METHOD

The voxmap-pointshell method is a volume-based collision detection algorithm developed by McNeely *et al.* [22-24] for haptic rendering of complex models. The functions of voxelization, voxel-sampling, and force generation were implemented in the Voxmap PointShell™ (VPS™) software available from Boeing. The voxmap-pointshell method can sustain a consistent 1000 Hz haptic refresh rate to render forces smoothly.

In the voxmap-pointshell method, models in the scene are voxelized in a pre-processing step to a voxel size specified by the user. To begin voxelization, the model is divided into regions of free space, object surface, and object interior. 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 (Fig. 1). 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 (Fig. 1).

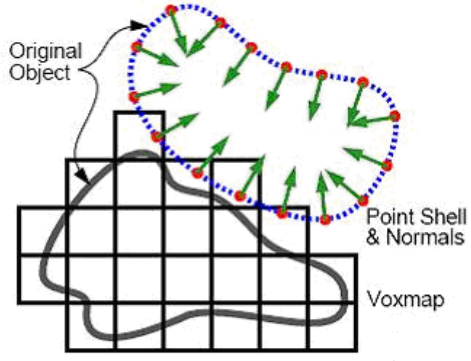

Fig. 1 Static and Dynamic CAD model representations [22]

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 algorithm checks for penetration of the sample points into the set of voxels in the scene. When penetration occurs, a local force model is applied. Depth of interpenetration, d, is calculated as the distance the point penetrates into the voxel relative to a plane that passes through the center of the voxel perpendicular to the pointshell normal direction (Fig. 2). Based on this depth of penetration, d, a force is calculated using a simple spring-force model ($F = kd$), where the force field stiffness is k.

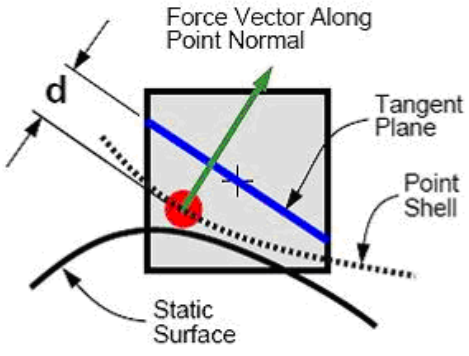

Fig. 2 Tangent plane force model [22]

In order to avoid instability, a coupling force is applied between the virtual manipulator, or haptic handle as shown in Fig. 3, and the dynamic object.

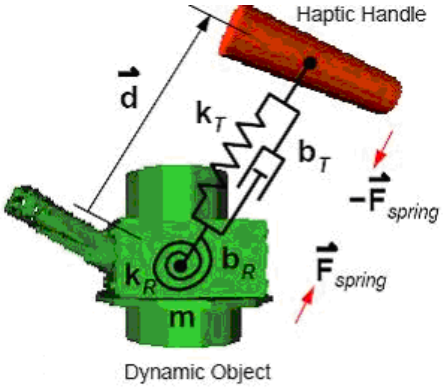

Fig. 3 Virtual coupling model [22]

This coupling force is composed of a spring force (Eq. 1) and a torque (Eq. 2). The coupling force includes a viscosity component to introduce some damping in the system. The spring force, F_{Spring}, and torque, τ_{Spring}, are computed as:

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

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

The collision force and torque between colliding objects are proportional to the amount of inter-object penetration, *d*, and a force field constant (Eq. 3 and 4).

$$\vec{F}_{Collision} = k_{ForceField}\,\vec{d} \quad \textbf{(Eq. 3)}$$

$$\vec{\tau}_{Collision} = k_{ForceField}\,\vec{\theta} \quad \textbf{(Eq. 4)}$$

The total force is based on the coupling and collision forces.

$$\vec{F}_{Total} = \vec{F}_{Spring} + \vec{F}_{Collision} \quad \textbf{(Eq. 5)}$$

$$\vec{\tau}_{Total} = \vec{\tau}_{Spring} + \vec{\tau}_{Collision} \quad \textbf{(Eq. 6)}$$

The voxmap-pointshell method renders forces from the virtual coupling (spring-damper system) between the dynamic object and the virtual hand, in addition to modeling part interaction. The method uses 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:

$$M\ddot{\vec{x}} = \vec{F}(t) \quad \textbf{(Eq. 7)}$$

$$I\dot{\vec{\omega}} + \vec{\omega} \times I \bullet \vec{\omega} = \vec{N}(t) \quad \textbf{(Eq. 8)}$$

where *x* is the linear displacement, *F(t)* is the applied force along time *t*, *M* is the object's mass, ⍰ is the angular velocity, *I* is the moment of Inertia, and *N(t)* is the angular force. The voxmap-pointshell method solves these equations 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.

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 detection of collisions at higher accuracy, but smaller voxels increase the memory requirements of the application and increase the computational load due to an increase in voxel-pointshell collisions.

Voxel size plays a crucial role in determining part collision and interpenetration. This becomes problematic during low clearance assemblies when a pin is inserted into a hole, for example. Multiple collisions occur with tight clearances. The limitation of voxel-based collision detection is its voxel-scale accuracy. This voxmap-pointshell method works well for most assembly scenarios, but due to the geometry approximation, the voxmap-pointshell method does not support assembly between two models with clearances smaller than twice the voxel size. Reducing the voxel size to produce a better geometry approximation is sometimes not feasible due to the increased memory requirements for smaller voxels. The approach explored in this research is to interactively shrink the pointshell model to support low clearance assembly.

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 shrkinking 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.

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. 4). 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.

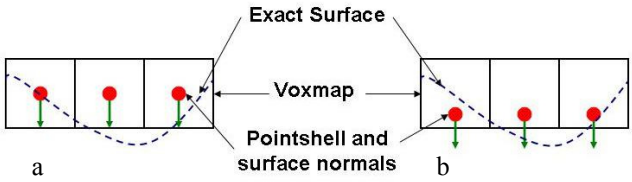

Fig. 4 a) Original location of pointshell with respect to voxels on moving object.
b) New location of pointshell after pointshell shrinking

Unfortunately, there is no limit on the amount of shrinkage that can be applied. It is possible to shrink the pointshell more than the voxel distance. This may cause local spatial inversion leading to erroneous forces and should be avoided. However, there is virtually no run-time cost for shrinking the pointshells, which is important to maintaining the haptic refresh rate.

METHODOLOGY

This section describes the development of a test environment used to explore pointshell shrinking and feature-based voxel size.

1. Experimental setup

The experimental setup consists of software and hardware.

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 [25-27]. VR Juggler provides an application interface that supports a wide variety of display devices.

OpenGL Performer will be used for rendering graphics and visualization. VPS™ will be used for collision detection and haptic rendering.

1.2. Hardware. The application runs on a Windows workstation with dual 3.6 GHz Intel Xeon processors and 3GB of RAM. A PCI Express Nvidia Quadro 4400 graphics card with 512 MB graphics memory is used. A magnetic tracking system (Polhemus Patriot) tracks the user's head and CrystalEye® shutter glasses provide stereo viewing (Fig. 5). Figure 5 shows a user manipulating the haptic device to control the placement and orientation of the models on the screen.

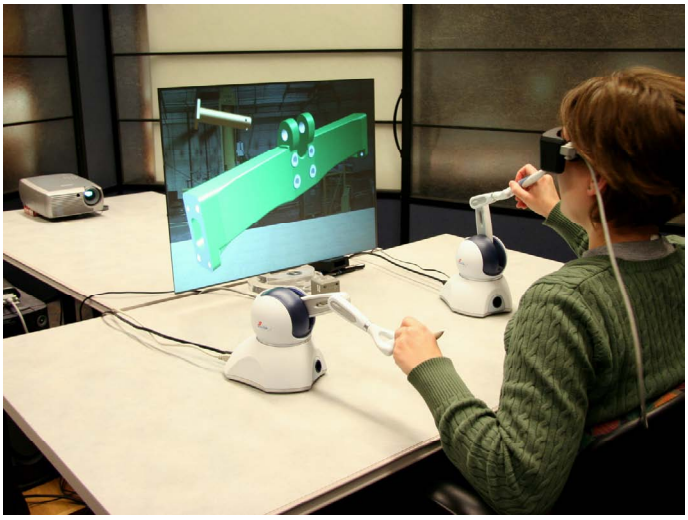

Fig. 5 Hardware setup

Two PHANTOM Omni haptic devices from SensAble Technologies are used in this setup. The Omni was chosen for its compact footprint and low cost. The Omni communicates with the computer using an IEEE-1394 FireWire port.

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. A standard test case of assembling a peg into a hole was chosen. When the peg radius increases, more opportunities arise for the peg to become stuck in the hole. This is 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 and create small resistive forces that cause the peg to stick. If the peg radius is sufficiently small, then the user's applied force can 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.

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 Solid Edge. The nominal diameter of the hole is 18.75 mm. The models being used are based on models used in Seth *et al.* [10].

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. Figure 6 shows the pin part and the block with the hole feature after voxelization.

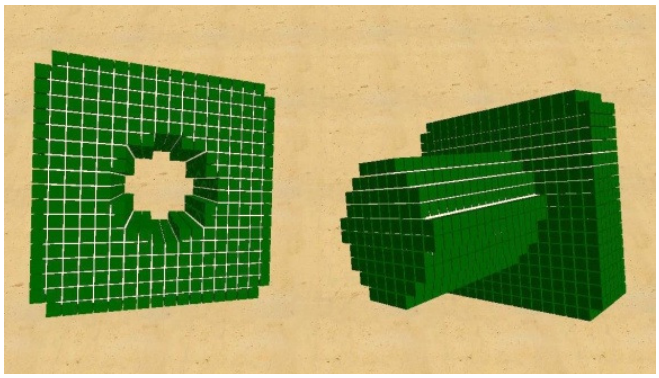

Fig. 6 Peg and hole

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. Pointshell shrinking is in the range of 0-100 %.

Table 1 Experimental setup for pointshell shrinking

Factor	Range	# of levels
Peg voxel size (mm)	0.25 to 2.0	10
Hole voxel size (mm)	0.25 to 2.0	10
Clearance (mm)	2.5, 1.4, 1.0	3
Pointshell shrinking	0 to 100%	11

JMP, statistical software available from SAS, 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 the hole part. The design varies both part's 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 inserted halfway into the hole, 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 2 minutes to determine if a single set of parts could be assembled. All tests were performed by the same operator (not the authors).

For these studies, the influence of assembly time and different operators is not considered. This would be an interesting extension of this research; however, the enhancement of a proven method with pointshell shrinking is of interest here.

2.2 Feature size affect on assemble-ability study. The same general methodology was used in this 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: 0.4 mm and 2.0 mm. The pointshell shrinkage was set to 50% 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	50%

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.

RESULTS

Analysis of variance statistics (ANOVA) were performed to analyze the results. A probability, p, was computed where p is the probability that the difference is due to chance factors. A chi-squared analysis, χ^2, represents the 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 shrinking is weakly significant as well (χ^2 3.209, $p<0.0732$).

The experimental data from the feature size effect was used to generate a prediction formula for appropriate voxel size. 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.

Table 4 ANOVA table for diameter dependency

Source	F	p
Peg diameter (mm)	29.989	<0.001
Hole diameter (mm)	33.139	<0.001

The p values for the peg (F 29.989, $p<0.001$) and the 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. The diameters 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 voxel size prediction formula follows in Eq. 9.

$$voxelsize = -0.001 - 0.027 * D_{Peg} + 0.027 * D_{Hole} \quad \textbf{(Eq. 9)}$$

where voxelsize is in mm, D_{Peg} is the peg diameter in mm and D_{Hole} is the hole diameter in mm.

The R^2 value of the curve fit is 73.91%, which presents some problems. This is a low R^2 value and therefore the equation is not a very close prediction of voxel size to support assembly. However, this equation can be used as a guideline to improve the trial and error method of selecting voxel size. With this 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.

CONCLUSION AND FUTURE WORK

In conclusion, this paper has presented results that aid designers in the assembly of low clearance parts with haptic feedback in a virtual environment. Low clearance manual assembly is very important to understanding how parts fit together and the ways that a human will handle parts. Pointshell shrinking has been shown to be an effective aid in assembly of low clearance parts. Methods have been presented here to guide the designer in selection of voxel sizes to support low clearance assembly.

There are several areas where improvements could be explored. 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 improve 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.

ACKNOWLEDGMENTS

We are grateful for the technical assistance of William McNeely. We also acknowledge the assistance of Justin Albright in preparation of models for the study and his assistance during the study. The authors would also like to thank the Iowa State University's Virtual Reality Applications Center for the use of computational resources and hardware. This research was funded by National Science Foundation grant CMMI - 0928774.

REFERENCES

[1] 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.

[2] 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.

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

[4] Lippmann, R.; Rößler, A.: Virtual Human Models in Product Development. In: Göbel. M.; Lang, U.; Landauer, J.; Wapler, M. (Hrsg.): *Proceedings of Virtual Environments*. Springer, Stuttgart, 1998, pp. 53/1-4.

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

[6] Gupta, R., Whitney, D., and Zeltzer, D., 1997, "Prototyping and Design for Assembly Analysis Using Multimodal Virtual Environments," Computer Aided Design (Special issue on VR in CAD), 29(8), pp. pp. 585-597.

[7] Coutee, A. S., McDermott, S. D., and Bras, B., 2001, "A Haptic Assembly and Disassembly Simulation Environment and Associated Computational Load Optimization Techniques," ASME Journal of Computing and Information Science in Engineering, 1(2), pp. 113-122.

[8] 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.

[9] Seth, A., Su, H. J., and Vance, J. M., 2006, "Sharp: A System for Haptic Assembly & Realistic Prototyping," Proceedings of *ASME International Design Engineering Technical Conferences*, 2006, Philadelphia, Pennsylvania.

[10] Seth, A., Su, H.-J., and Vance, J. M., 2008, "Development of a Dual-Handed Haptic Assembly System: SHARP," ASME Journal of Computing and Information Science in Engineering, 8(4), p. 004502, 8 pages.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.
[16] 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..
[17] Gomes, A., and Zachmann, G., 1999, "Virtual Reality as a Tool for Verification of Assembly and Maintenance Processes," Computers and Graphics, 23, pp. 389-403.
[18] Seth, A., Vance, J. M., and Oliver, J. H., 2007, "Combining Geometric Constraints with Physics Modeling for Virtual Assembly Using Sharp," DETC2007-34681, *Proceedings of ASME Design Engineering Technical Conferences*, Las Vegas, 2007.
[19] Seth, A., Vance, J. M., and Oliver, J. H., 2010, "Virtual Reality for Assembly Methods Prototyping: A Review," Virtual Reality Journal, in press.
[20] 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.
[21] 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.
[22] McNeely, W. A., Puterbaugh, K. D. And, Troy, J. J., 1999, "Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling," *SIGGRAPH '99 Conference Proceedings, Annual Conference Series*, Los Angles, CA., pp. 401-408.
[23] 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.*
[24] McNeely, W., Puterbaugh, K. D., and Troy, J. J., 2006, "Voxel-Based 6-Dof Haptic Rendering Improvements," Haptics-e, 3(7).
[25] Just, C., A. Bierbaum, A. Baker, and, C. Cruz-Neira, 1998, "VR Juggler: A Framework for Virtual Reality Development", *Proceedings of the 2nd Immersive Projection Technology Workshop)*. 1998. Ames, IA.
[26] Bierbaum, A., and Cruz-Neira, C., 2000, "Runtime Reconfiguration in VR Juggler," *Proceedings of the 4th Immersive Projection Technology Workshop*, June 2000, Ames, IA.
[27] 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.