9-2003

Using VPS (VoxMap Pointshell) as the Basis for Interaction in a Virtual Assembly Environment

Chang E. Kim
Iowa State University

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

Using VPS (VoxMap Pointshell) as the Basis for Interaction in a Virtual Assembly Environment

Abstract
Realistic part interaction is an important component of an effective virtual assembly application. Both collision detection and part interaction modeling are needed to simulate part-topart and hand-to-part interactions. This paper presents a comparison of several common collision detection algorithms and examines the VoxMap Pointshell (VPS) method as it is used in an application to evaluate proposed assembly methods. Results from several performance tests on VPS are presented. VPS was found to provide realistic collisions and physicallybased modeling interaction with excellent performance. This paper concludes by presenting how VPS has been implemented to handle multiple dynamic part collisions and two-handed assembly using the 5DT dataglove in a projection screen virtual environment.

Keywords
VRAC, Virtual assembly

Disciplines
Computer-Aided Engineering and Design

Proceedings of DETC'03
ASME 2003 Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
Chicago, Illinois, USA, September 2-6, 2003

DETC2003/CIE-48297

USING VPS (VOXMAP POINTSHELL) AS THE BASIS FOR INTERACTION IN A VIRTUAL ASSEMBLY ENVIRONMENT

Chang E. Kim
Department of Mechanical Engineering
Virtual Reality Application Center
Iowa State University
Ames, Iowa 50011
changkim@vrac.iastate.edu

Judy M. Vance
Department of Mechanical Engineering
Virtual Reality Application Center
Iowa State University
Ames, Iowa 50011
jmvance@vrac.iastate.edu

ABSTRACT

Realistic part interaction is an important component of an effective virtual assembly application. Both collision detection and part interaction modeling are needed to simulate part-to-part and hand-to-part interactions. This paper presents a comparison of several common collision detection algorithms and examines the VoxMap Pointshell (VPS) method as it is used in an application to evaluate proposed assembly methods. Results from several performance tests on VPS are presented. VPS was found to provide realistic collisions and physically-based modeling interaction with excellent performance. This paper concludes by presenting how VPS has been implemented to handle multiple dynamic part collisions and two-handed assembly using the 5DT dataglove in a projection screen virtual environment.

INTRODUCTION

In the last few years, virtual reality (VR) has emerged as an engineering design tool due to its ability to provide three-dimensional, interactive environments, which allow humans to interact with digital representations of products using natural human motions. Jayaram (Jayaram, S., Vance, M. J. et al. 2001) defines the key elements of VR as "a) immersion in a 3D environment through stereoscopic viewing, b) a sense of presence in the environment through tracking of the user and often representing the user in the environment, c) presentation of information of the sense other than vision, and d) realistic behavior of all objects in the virtual environment."

Virtual assembly, as referred to in this paper, is the ability to assemble CAD models of parts using a three-dimensional immersive, user interface and natural human motion. Most often, engineers view three-dimensional representations of CAD objects using a two-dimensional computer screen where the parts can be rotated for viewing using the desktop mouse. Parts can be shown assembled and interference checking performed. Immersive virtual reality provides the ability for a user to view and interact with full-scale CAD models by reaching out and grabbing the models in a stereo-viewing environment. In reference to assembly planning, users can enter the immersive virtual reality environment and interact with real size representations of parts while prototyping assembly operations.

To facilitate the development of a virtual assembly program, part interaction methods must be investigated. These part interaction methods must detect part-to-part collisions and also hand-to-part interactions. Realistic part behaviors which occur due to gravity and physical dynamics must also be simulated. The specific objectives of this research are to:

1. Investigate various collision detection and part behavior algorithms that will be suitable for a virtual assembly simulation.
2. Implement and design a program to facilitate immersive virtual assembly methods prototyping.

BACKGROUND

Virtual reality provides a tool where users can interact with digital objects using natural human motions. In an immersive virtual environment the user interacts with objects just like in the real environment. If the user wants to pick up an object from a table, he/she moves to the table, reaches out, intersects a virtual hand model with the object and performs some action that attaches the object to his/her hand. For virtual assembly, this medium can be used early in the design process to prototype assembly operations. Factory workers can be brought into the design process before the product design is finalized and asked to assemble products. Based on the finding of the virtual product assembly process, changes in product design can be recommended which can result in significant cost savings to the company.

Fraunhofer-Institute for Industrial Engineering (IAO) has developed an assembly planning system that makes it possible to interactively assemble and disassemble components and modules in a virtual surrounding. (Bullinger, H. J., Richer, M. et al. 2000) It uses VirtualANTHROPOS – a virtual model of a human being – in order to carry out assembly operations. Since VirtualANTHROPOS is based on the anthropometrical module ANTHROPOS, VirtualANTHROPOS is able to display accurate human kinematics in order to calculate assembly time and cost. This application uses collision detection to indicate part interaction, but does not implement part behaviors.

Jayaram et al. (Jayaram, S., Jayaram, U. et al. 1999) (Jayaram, S., Jayaram, U. et al. 2000) (Taylor, F., Jayaram, S. et al. 2000) (Jayaram, U., Tirumali, H. et al. 2000) developed a virtual assembly application called VADE (Virtual Assembly Design Environment) at Washington State University. This application can take Pro/E CAD files as input. Two-handed assembly can be performed using Cybergloves that can detect finger bend angles for a realistic representation of the hand. Both a menu system and a voice recognition system can be used to manage the virtual environment. VADE has the ability to detect collisions and also model part behaviors. Since VADE uses constraint-based part behavior modeling, reaction forces are not generated when objects collide with each other. VADE can display a virtual environment either through a head mounted display or a single-pipe projection system but does not currently have the capability to display in a multi-pipe environment.

Terrence Fernando et al. (Fernando, T., Marcelino, L. et al. 2000) developed a virtual assembly application called IPSEAM (Interactive Product Simulation Environment for Assessing Assembly and Maintainability) at the University of Salford that includes a limited ability to model part behavior. This application has been developed using the constraints based geometric modeling approach. Modeling, however, is limited to simulating part behavior of lower pair joints only such as constraints between surfaces, leaving out constraints involving vertices and edges.

One virtual assembly application that has been tested using industrial examples is the Virtual Environment for General Assembly (VEGAS) (Johnson, T. C. 2000) developed by Vance and Johnson at Iowa State University. It uses the geo file format for its graphic model input and Voxmap PointShell (VPS) for collision detection. VEGAS can be used in both single and multi-pipe display environments. The work presented here expands on the functionality of VEGAS to include part behavior modeling as well as to explore the use of VPS as a collision detection and part modeling software for virtual assembly.

In order to develop a virtual assembly application that will provide adequate feedback to the user in his/her evaluation of the assembly process, several factors must be present in the application. Stereo-viewing, and position tracking of both the user's head and hands are required to provide the three-dimensional interface to the CAD data. Collision detection is needed between parts and between the user's body and the parts in the environment in order to indicate to the user that there are collisions occurring during the assembly process.

Many collision detection algorithms have been developed and tested with three-dimensional CAD data. I-collide(Cohen, J. D., Lin, M. C. et al. 1995), SWIFT (Ehmann, S. A. and Lin, M. C. 2000), RAPID (Gottschalk, S., Lin, M. C. et al. 1996), V-collide (Hudson, T., Lin, M. C. et al. 1997), PQP (Larsen, E., Gottschalk, S. et al. 1999) and SWIFT++ (Ehmann, S. A. and Lin, M. C. 2001) have been designed by individuals at the University of North Carolina GAMMA (Geometric Algorithms for Modeling, Motion and Animation) research group. V-clip (Mirtich, B. 1998) was created by Brian Mirtich in 1998 at the Mitsubishi Electric Research Laboratories. William McNeely at Boeing developed VPS (McNeely, W. A., Puterbaugh, K. D. et al. 1999) in 1999. The characteristics and comparisons of these algorithms will be addressed in following sections.

In addition to collision detection, simulating physical part behaviors in the virtual environment is a key component of a realistic virtual assembly application. Physical constraints must be present in the environment that simulate part behavior such as collars sliding on shafts and parts sliding on surfaces in order to simulate real assembly operations. There are two methods to simulate physical properties in VR: geometric constraint modeling and physically-based modeling. To apply geometric constraints, certain geometric properties of the objects are identified which would result in assembly constraints. For each hole, for example, a sliding axis is identified. For each surface that could be used as a contact surface, a contact surface constraint is identified. Each part must go through a pre-processing step where all possible constraints must be identified. Physically-based modeling, on the other hand, is a method that incorporates equations governing the motion of objects in the simulation. Gravity effects and contact forces are modeled in a general sense and applied when contact is detected. In the assembly application presented here we have implemented physically-based modeling because of the desire to minimize the pre-processing of CAD input files.

The following sections of this paper will present a description of the virtual reality system that was used, a comparison of collision detection software with respect to the unique requirements of virtual assembly, an evaluation of the physically-based modeling capabilities of VPS, and a description of the virtual assembly application.

SYSTEM

Although the software developed as a result of this research can be used with single pipe display systems such as head-mounted displays, single projection walls, and projection benches, the preferred virtual reality device used at Iowa State University is the multi-pipe stereo projection environment. The Virtual Reality Applications Center has two such systems, the C4 and the C6. The C6 is a 10 ft. x 10 ft. x 10 ft. room equipped with 6 rear projection surfaces, which serve as the walls, ceiling and floor. The users wear stereo shutter glasses which are synchronized with the computer display to alternate the left and right eye views at a rate of 96 Hz in order to produce stereo images. A magnetic tracking system tracks the user's head, hand, and arm position. A 24-processor SGI Onyx2 Reality Monster supplies the computational power and six InfiniteReality2 graphic pipes, each with 256MB of texture memory manage the graphics output. The processors are 400MHz MIPS R12000's and the computer contains 12Gb of RAM. The C4 is a re-configurable projection system that has three projection walls and a floor projection surface which is also driven by an SGI computer.

For user input, two wireless 5-W Data Gloves from Fifth Dimension Technologies are used. The gloves feature advanced fiber-optic flexure sensors which generate 15 levels of finger-bend data. This enables the users to grab, move and release parts in the virtual environment. A software driver for the gloves to interface with the virtual reality software, VRJuggler, used in this application was developed.

COLLISION DETECTION ALGORITHMS

There is considerable discussion in the geometric modeling community concerning the use of polygon-based vs. volume-based collision detection algorithms (Coutee, A. S. and Bras, B. 2002) (McNeely, W. A., Puterbaugh, K. D. et al. 1999). Our decision to use VPS, which is a volume-based algorithm, is specifically tied to our need to perform virtual assembly. The rationale for the selection of VPS over other more common polygon-based algorithms is presented in this section.

The virtual assembly application must take CAD file input and allow users to naturally pick up and assemble digital objects in the immersive virtual environment. The factors to be considered in selecting a collision detection algorithm for this application include:

1. Ability to handle complicated part topology
2. Performance speed
3. Preprocessing requirements for CAD input models
4. Accuracy of collision detection
5. Ability to detect not only collisions, but to perform other types of part-to-part interaction queries

The collision detection algorithms investigated in this research include:

1. I-collide (Cohen, J. D., Lin, M. C. et al. 1995)
2. V-clip (Mirtich, B. 1998)
3. SWIFT (Ehmann, S. A. and Lin, M. C. 2000)
4. RAPID (Gottschalk, S., Lin, M. C. et al. 1996)
5. V-collide (Hudson, T., Lin, M. C. et al. 1997)
6. PQP (Larsen, E., Gottschalk, S. et al. 1999)
7. SWIFT++ (Ehmann, S. A. and Lin, M. C. 2001)
8. VPS (McNeely, W. A., Puterbaugh, K. D. et al. 1999).

I-collide is an exact collision detection library developed in 1995 for large environments composed of convex polyhedra for multi-body collision detection. RAPID works with non-convex models but detects pair-wise collision only. V-collide, which is based on RAPID, includes the ability to detect multiple body collisions. PQP, which is also based on RAPID, is a pair-wise collision detection algorithm that supports non-convex modes. It also can perform distance computation and tolerance verification queries. SWIFT provides various queries such as intersection detection, tolerance verification, exact and approximate distance computation, and contact determination of convex models. SWIFT++ has been developed based on SWIFT and supports general three-dimensional polyhedral objects. All of these methods are polygon-based intersection algorithms. VPS, on the other hand, represents geometry using voxels which are small cube elements (McNeely, W. A., Puterbaugh, K. D. et al. 1999). VPS can detect collisions, perform tolerance verification, approximate distances, determine contact normals and center of mass. VPS also has the ability to calculate physically-based modeling of part behavior.

The VPS method defines two objects in the environment: a dynamic object that the user can move in the environment and a static object that consists of all other objects that do not move in the environment. The geometric model of all parts are voxelized prior to start up of the virtual assembly simulation. When an object becomes a dynamic object, the center of each voxel is maintained and the object is represented by a collection of these points, called a point shell. When the point shell penetrates into the static voxel object, a collision is detected. In addition, the penetration is used to determine the reaction forces. These forces can be used to model object behavior.

The rest of this section will explore each of the five consideration factors used to distinguish collision detection algorithms for virtual assembly.

Ability to handle complicated part topology

Collision detection algorithms can be divided into two categories according to the required topology of the input files: convex-based packages and polygon soup-based packages. I-Collide, V-clip and SWIFT are the convex-based packages and RAPID, V-collide, PQP, SWIFT++ and VPS are the polygon soup-based packages. Convex-based packages only work if the input geometry consists of convex polyhedra. These algorithms can give more information than the polygon soup-based packages, such as distances between objects, penetration depth and an approximate distance, with good speed. Polygon soup-based packages can deal with arbitrary topology, but they do not have the ability to determine much beyond detecting the collision. Though convex-based packages are fast algorithms and can provide additional query information, they are not suited for virtual assembly applications where parts consist of arbitrary topology. Therefore, I-Collide, V-clip and SWIFT are not suitable for virtual assembly applications.

Performance speed

Because of the need for real-time collision detection in virtual reality, performance speed is a critical consideration. In general, polygon-based algorithms that deal only with convex topology (I-Collide, V-Clip and SWIFT) are faster than those that deal with more general topology (RAPID, V-Collide, PQP and SWIFT++). However since our virtual assembly application must process general topology, we are limited to selecting only from the polygon-soup algorithms if we choose a polygon-based method. Of these methods, SWIFT++ is the fastest.

SWIFT++ is fast, even though it is a polygon soup-based algorithm, because it uses SWIFT, which is a convex-based algorithm, as its core (Ehmann, S. A. and Lin, M. C. 2001). The SWIFT++ algorithm takes nonconvex geometry and subdivides the geometry into a series of convex objects using its "decomposer" preprocessor. The convex-based algorithm can then be applied to all of the sub-objects in the scene.

VPS is approximately four times faster than V-collide because volume elements called voxels are used to represent the static object and a set of points called a point shell are used to represent the dynamic object (Boeing Company 1999). During each time step, the motion transformation of the dynamic object is applied to every point shell. Collision detection in VPS is volumetric intersection between voxels and pointshells which differs from the conventional surface intersection approach of polygon-based algorithms. The

volumetric intersection scheme makes VPS faster than any polygon-based algorithm when complicated geometry is present.

Preprocessing requirements for CAD input models

One of the goals of the research presented here is to develop a flexible and easy-to-run virtual assembly application that can accommodate different models and assembly conditions, therefore ease of making the collision input file is an important factor. The input file format of RAPID, V-collide and PQP consist of simple ASCII files that can be generated from the general CAD file format such as STL or ASE.

Creating input files for SWIFT++ is more difficult. SWIFT++ requires an extra file conversion step. Simple tessellated ASCII files that are generated from general CAD files are input into the "decomposer" software. Decomposer is a standalone executable library that takes basic model geometry and subdivides it into a series of convex objects for SWIFT++. The graphic model must be perfect for the decomposer process to work without an error. Most CAD packages create graphic models containing some geometrical errors. To fix these errors, an application program named IVECS (Interactive Virtual Environment for the Correction of STL files) (Morvan, S. M. and Fadel, G. M. 1996) has been developed by Dr. Georges M. Fadel at Clemson University. IVECS displays the errors found in the STL file surface and allows the user to correct them manually. Once the STL file is consistent, decomposer can be used to prepare the file for processing by SWIFT++. IVECS is a powerful tool, but the process of fixing the errors in complicated .STL files is very time consuming and tedious.

VPS also accepts standard ASCII files in the STL or ASE format. The conversion program called stl2vps in VPS converts STL files into binary VPS format files. This conversion creates the voxel representations needed for the collision detection. VPS has the ability to create the voxel model either within the VR application at run-time or through the use of the stl2vps conversion program.

Accuracy of collision detection

Since polygon-to-polygon intersections form the basis of polygon-based collision detection algorithms, the accuracy of polygon-based collision detection is the same as the accuracy of the tessellated model. Since VPS approximates the surface geometry using volume elements, or voxels, to calculate collisions, the accuracy using VPS is inversely proportional to the voxel size. Similar to the polygon-based methods, however, a trade-off exists between accuracy and performance. Smaller voxels or smaller polygons require more computation time.

Ability to detect not only collisions, but to perform other types of part-to-part interaction queries

If a physical constraint interaction model is needed during the assembly process operation, the collision detection algorithm needs to query tolerance verification, exact and approximate distances, nearest features, center of mass, and contact normal vectors in addition to intersection status. Types of queries for four different collision detection algorithms are shown in Table 1.

Coutee and Bras (2000) compare collision detection methods for disassembly applications according to the following five features: closest point, collision features, depth of penetration, programmatic geometry construction, and n-body detection. In this paper we expand the "collision features" comparison in depth to include examination of intersection, tolerance verification, distance, contact normal, and center of mass capabilities of each collision detection algorithm. SWIFT++ is the most versatile algorithm and can provide the most queries.

VPS does not provide exact distance, nearest features, or nearest point calculations, but provides part-to-part interactions using a physically based modeling approach. Within VPS are functions that calculate the interaction forces between colliding objects. These forces are used to model the part-to-part interactions.

Table 1: Types of queries in four collision detection algorithms (0 = present, x = not present)

	V-collide	PQP	SWIFT++	VPS
Intersection	o	o	o	o
Tolerance verification	x	x	o	o
Exact distance	x	o	o	x
Approximate distance	x	x	o	o
Nearest features	x	x	o	x
Nearest points	x	x	o	x
Contact normal	x	x	o	o
Center of mass	x	x	o	o

Physical constraint interaction generation algorithm

Along with collision detection, physical constraints in the virtual environment are implemented to make users feel immersed. VPS has a built-in physical constraint interaction capability called PBM (Physical Based Modeling). It generates a collision response and calculates the subsequent motion. Details about implementation issues concerning the physical interaction capabilities of VPS are further detailed later in this paper. Without physical constraint modeling, the user must pre-define geometry constraints between objects before the virtual reality application starts. The use of physically based modeling, therefore, is a more general approach to modeling interaction constraints. In addition, VPS has swept volume generation capability and a haptic device controller.

Summary of collision detection algorithm selection decision

VPS does not have any restriction on the input model shape and has been shown to be compatible with our graphic interface, SGI Performer. It is fast and provides sufficient query results. It is easy to make input files, and has a built-in interaction generation library including swept volume generation, and a haptic device controller. Therefore VPS has been used in this project to support collision detection and object manipulation.

VIRTUAL ASSEMBLY APPLICATION

(Fig. 1) shows a user interacting in the C6 using the virtual assembly application. Models can be imported from .JT, .3DS, .WRL, and any generic CAD file format.

Figure 1: Virtual Assembly Application

Additional position trackers are placed on the user's wrist, forearm and upper arm (Fig. 2). Virtual hand and arm models have been made to represent positions and locations of the user's hand, forearm and upper arm (Fig. 3). Collision detection without force calculation is implemented between the hand, arm model and each object. When the user collides the hand model with any of the objects in the virtual environment, a bounding box will appear around the object to indicate which object the user intends to grab. Gestures are used to indicate if the object should be grabbed. Once an object has been selected, the collision detection between the hand-object model is deactivated and the object becomes attached to the hand. When the object is released, collisions then can be detected between the hand and the other objects. Another collision detection is implemented between the arm models and the other objects.

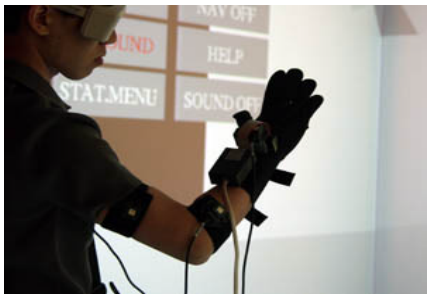


Figure 2: 5DT Data Glove 5-W and trackers

The collision processes are shown in (Fig. 4).

Interaction with the virtual environment is through a three-dimensional menu that can be positioned anywhere with fixed height in the virtual space (Fig. 5). The menu initially appears on the left wall. The options on the menu are Reset, Navigation, Background Change, Help, Dynamic menu on/off, Sound on/off and Arm Models Init. The user can reset all the objects to their original positions and orientations. Arm Model Init buttons calibrate the locations and lengths of each lower arm and upper arm model. The Navigation button will activate or deactivate the navigation mode. The menu can be moved according to the position of the user's head tracker by activating the dynamic/static menu button on the menu. The user can also change the background image and background sound. The Help button shows a textured three-dimensional model of the wand with operation directions. In addition to the background sound, localized collision sound is implemented. The predefined collision sound is generated at the position where the collision is detected. The overall structure of the program is shown in (Fig 6) to (Fig 10).

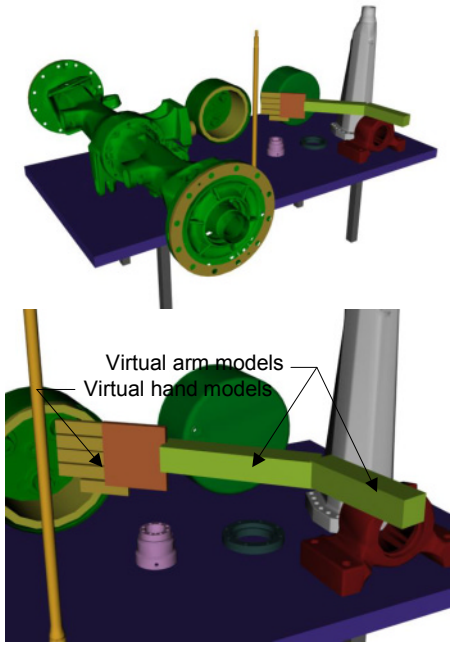
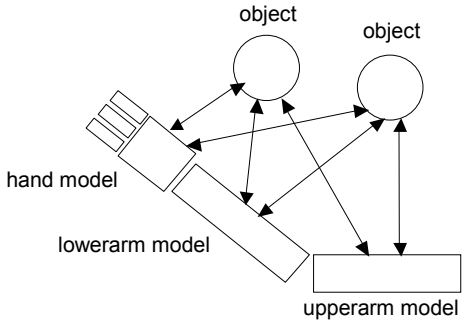
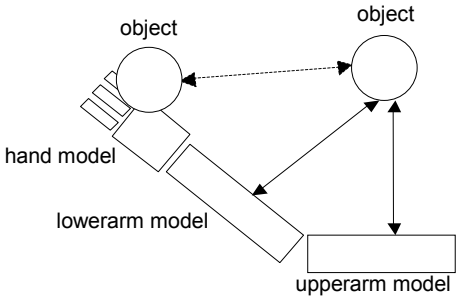


Figure 3: Interaction in virtual environment



(a) Before grabbing an object



(b) After grabbing an object
(Dashed line represents collision with force calculation and solid lines represent collision without force calculation)

Figure 4: Collision detections before/after grabbing an object

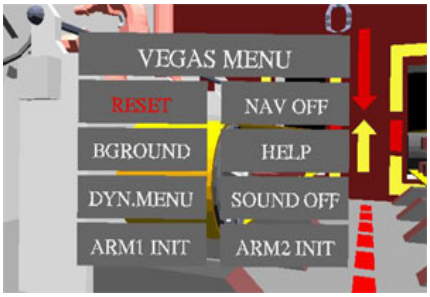


Figure 5: Virtual Menu

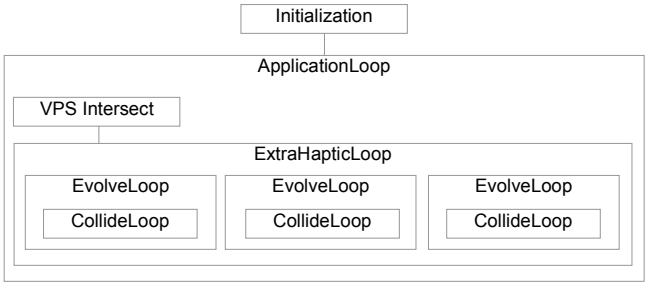


Figure 6: Overall Structure

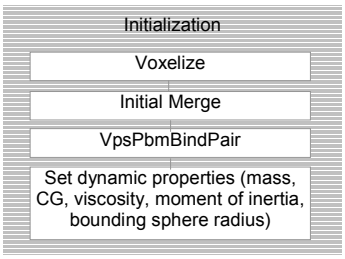


Figure 7: Initialization

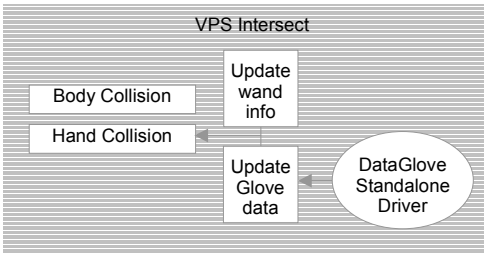


Figure 8: VPS Intersect

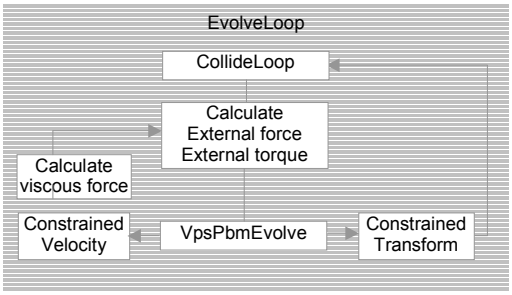


Figure 9: EvolveLoop

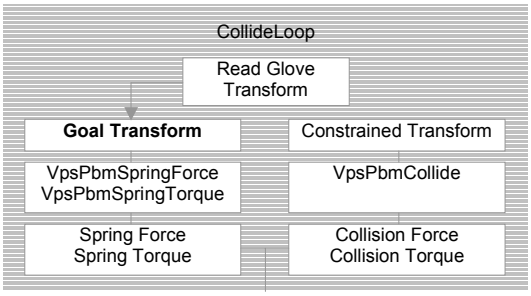


Figure 10: CollideLoop

VPS PHYSICALLY-BASED INTERACTION MODELING IMPLEMENTED IN VIRTUAL ASSEMBLY APPLICATION

VPS contains a part interaction library called PBM (Physically Based Modeling). This library models part interaction using rigid body dynamics principles. The basic equation of motion used to describe the model reaction is the Newton-Euler equation,

$$M\frac{d^2x}{dt^2} + C\frac{dx}{dt} + Kx = F(t) \qquad (1)$$

where x is the displacement, F(t) is an external force along time t, M is an object's mass, C is a damping coefficient and K is a spring constant. VPS Physically Based Modeling (PBM) solves the Newton-Euler dynamic equation numerically by using finite difference approximations to describe rigid body dynamics. The appropriate time marching step, linear/angular spring constants, and linear/angular damping coefficients need to be defined for stability. The computational cost of the physically-based modeling method is relatively more expensive than that of the constraint-based geometric modeling, and numerical instability can be a problem with physically-based modeling. However, physically-based modeling techniques enable the realistic dynamic manipulation of a complex rigid object. The theory of physically-based modeling has been extensively studied in (Baraff, D. and Witkin, A. 1997) and (Mirtich, B. 1998).

The next sections discuss the implementation of VPS physically-based modeling within the virtual assembly environment. A performance study is also presented which examines the effect of increasing the number of dynamic parts in the virtual environment and also increasing the number of voxels for one dynamic part.

Part-to-part interaction limitations

When the user grabs and moves a part in the application, the part's speed should be fast enough to follow the natural human motion. If the application cannot keep up with the user, the user may see the object lagging behind the hand movement. In VPS, two factors affect a part's speed: voxel size and VPS physically based modeling (PBM) update rate.

The maximum distance an object can move and maintain part-to-part interaction is defined in VPS as maxTravel to

prevent a part from penetrating into the others, and it is expressed as:

$$\text{maxTravel} = 0.5 \times \text{VoxelSize} \qquad (2)$$

The maximum speed that an object can be moved is defined as maxSpeed and can be expressed as:

$$\text{maxSpeed} = \text{maxTravel} \times \text{AUR} \qquad (3)$$

where AUR is the application update rate defined as a minimum of the stereo projector graphic update rate (GUR) which is the stereo projector refresh rate in our case, the tracker update rate (TUR) and 1/PBM calculation time (1/PBMCT). It is expressed as:

$$\text{AUR} = \text{Min}(\text{GUR}, \text{TUR}, 1/\text{PBMCT}) \qquad (4)$$

In our assembly application the VoxelSize is 9 mm. The stereo projector refresh rate in the C6 is 96 Hz, which means that stereo images are updated at 48 Hz. The tracker update rate is 68.3 Hz. PBMCT depends on the number of colliding parts and the part's size. It can be reasonably assumed that GUR is smaller than 1/PBMCT. Therefore, the maximum speed will be about 216 mm/s.

In order to increase the maximum speed an object can travel, the program can perform multiple PBM calculations per update frame. If N number of PBM calculations happen every application frame, equation (3) and (4) can be changed as:

$$\text{maxSpeed} = \text{maxTravel} \times \text{AUR} \times N \qquad (5)$$

$$\text{AUR} = \text{Min}(\text{GUR}, \text{TUR}, 1/(\text{PBMCT} \times N)) \qquad (6)$$

The maxSpeed can now be controlled by varying N. However, incrementing N does not always increase the maxSpeed because AUR can also change maxSpeed. When N is set relatively high or the PBM calculation takes a long time, AUR becomes small which decreases maxSpeed. If the number of dynamic parts is increased, more calculation time is needed for PBM. A PBMCT test was completed to examine the limitation on the number of dynamic parts that could be in the environment with the desired maxSpeed.

A natural speed of hand movement is set to .9 m/s. The voxel size for assembly parts is set to 9 mm. According to the equation (5) above, AUR x N should be over 200 Hz to meet the 0.9 m/s maxSpeed. That 200 Hz is called TargetFrame. Figure 11 shows the test model setting. When the number of dynamic parts is increased, more PBMCT is needed. When the number of dynamic parts exceeds 16, the PBMCT increases exponentially and when 1/(PBMCT) becomes smaller than GUR, the application update rate (AUR) will decrease.

In order to examine the effect the number of voxels has on the PBMCT, a trial test of from 1 to 20 dynamic parts has been performed with various numbers of voxels. When the application is started, all dynamic parts fall down, bounce a little on the table box and collide with other dynamic parts. Figure 12 is the graph of PBMCT for multiple dynamic parts. It shows that the total number of voxels does not affect the application performance as much as the number of dynamic objects. Table 2 presents the number of voxels test data results.

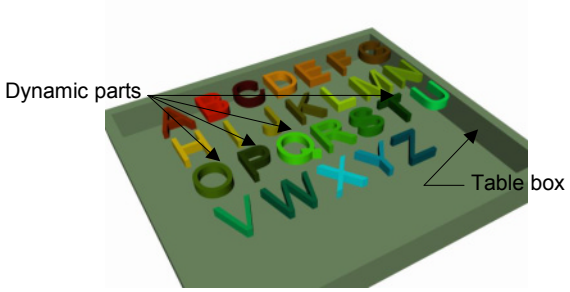


Figure 11: Performance test model setting

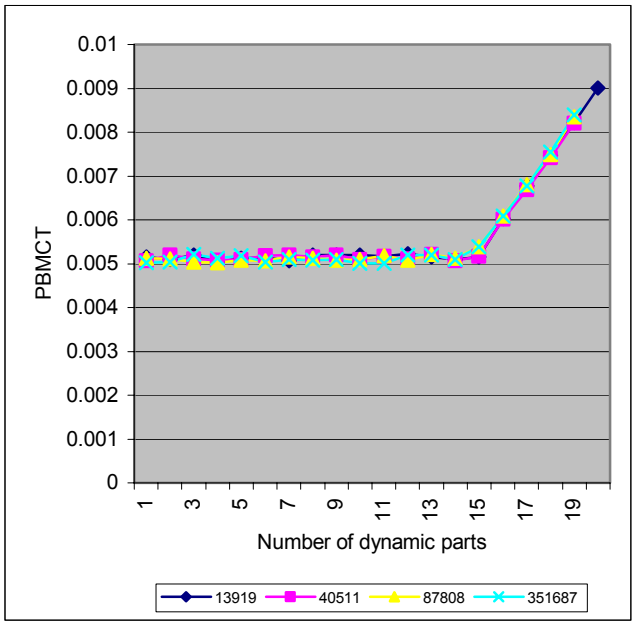


Figure 12: Change in PBMCT with increasing number of dynamic parts and various numbers of total voxels from 113919 to 351687

Table 2. Effect of number of voxels on HapticFrame time period

voxel size	0.05	0.03	0.02	0.01
No. voxels	13918	40511	87808	351687
No. dyn parts	HapticFrame time period			
1	0.00515	0.00507	0.00513	0.00502
2	0.0051	0.0052	0.00511	0.00503
3	0.0052	0.00511	0.00503	0.00521
4	0.00504	0.00508	0.00502	0.00513
5	0.00513	0.00509	0.00508	0.00518
6	0.00514	0.00518	0.00507	0.00504
7	0.00507	0.0052	0.00515	0.00509
8	0.00519	0.00516	0.00512	0.00508
9	0.0052	0.00519	0.00508	0.0051
10	0.0052	0.00509	0.00509	0.00501
11	0.00517	0.00517	0.00518	0.00501
12	0.00522	0.00511	0.00508	0.00519
13	0.00516	0.00521	0.00521	0.00519

14	0.0051	0.00506	0.00513	0.0051
15	0.00515	0.00517	0.00538	0.00539
16	0.00603	0.00602	0.00609	0.00609
17	0.00678	0.00668	0.00681	0.00678
18	0.00747	0.00742	0.00749	0.00755
19	0.00822	0.00821	0.00835	0.00839
20	0.00901			

Pair-wise collision detection

VPS PBM has two main queries: VpsPbmCollide and VpsPbmEvolve. VpsPbmCollide calculates reaction forces between two objects and VpsPbmEvolve generates a new position for the dynamic object based on reaction information generated from VpsPbmCollide. For two-handed or multiple users assembly, the application currently supports multiple dynamic parts. Since VPS is a pair-wise collision detection algorithm, VpsPbmCollide is called for each dynamic object's intersection with every other object in the environment (Fig. 13). VpsPbmEvolve is also called for each dynamic object (Fig. 13).

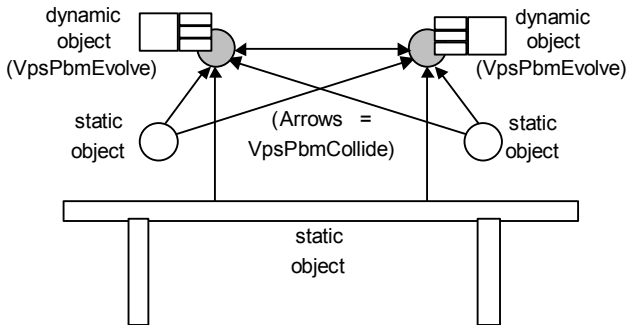


Figure 13: VpsPbmCollide and VpsPbmEvolve with two dynamic parts

Accuracy in VPS

An understanding of the magnitude of the accuracy of the collision detection algorithm is important when designing a virtual assembly application. By definition, the voxmap is "a single spatial occupancy map" with a certain predefined size and the point shell is "the center point of the voxmap" (McNeely, W. A., Puterbaugh, K. D. et al. 1999). The environment of static objects is represented by voxmaps and the dynamic object's motion is described as point shells. When a point shell interpenetrates a tangent plane that passes through the voxel's center point, a depth of penetration is calculated. Therefore the maximum error in VPS PBM is:

$$\text{MaxError} = \sqrt{3} \cdot \text{voxel size} \qquad (6)$$

If the voxel size is 9 mm, the MaxError is 15 mm.

CONCLUSIONS

In this paper, we described our efforts to develop an application for the virtual assembly environment for multiple dynamic parts and we summarized what we have learned about using VPS techniques for the dynamic interactions. Standard ASCII polygonal data files are used for the input models. Various collision detection algorithms have been investigated. Integrating VPS collision, VPS PBM techniques, and Data Glove hardware into a virtual assembly application provides the user a three-dimensional interactive experience. Designers and engineers can gain invaluable insights into the entire design and assembly process using VPS (Voxmap PointShell) as the basis for the interaction in a virtual assembly environment.

ACKNOWLEDGEMENTS

We are very grateful for the technical assistance of William McNeely of the Boeing Company with our investigation of the performance of VPS. This work was funded by Deere & Company.

REFERENCE

Baraff, D. and Witkin, A., 1997, Physically Based Modeling: Principles and Practice (Online Siggraph '97 Course Note), http://www-2.cs.cmu.edu/~baraff/sigcourse/.

Boeing Company, 1999, VPS Performance Comparison, http://www.boeing.com/assocproducts/vps/performance.html.

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

Cohen, J. D., Lin, M. C., Manocha, D. and Pomangi, M. K., 1995, "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments." The 1995 ACM International 3D Graphics Conference, pp.189-196.

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

Ehmann, S. A. and Lin, M. C., 2000, "SWIFT: Accelerated Proximity Queries Between Convex Polyhedra By Multi-Level Voronoi Marching." Technical report, Computer Science Department, University of North Carolina at Chapel Hill.

Ehmann, S. A. and Lin, M. C., 2001, "Accurate and Fast Proximity Queries between Polyhedra Using Surface Decomposition." Eurographics. Computer Graphics Forum **20** (3).

Fernando, T., Marcelino, L., Wimalaratne, P. and Tan, K., 2000, "Interactive Assembly Modeling within a CAVE Environment." Eurographics-Portuguese Chapter, pp. 43-49.

Gottschalk, S., Lin, M. C. and Manocha, D., 1996, "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection." ACM SIGGRAPH'96, pp. 171-180.

Hudson, T., Lin, M. C., Cohen, J. D., Gottschalk, S. and Manocha, D., 1997, "V-COLLIDE: Accelerated Collision Detection for VRML." Proceedings of the Second Symposium on Virtual Reality Modeling Language, pp. 119-125.

Jayaram, S., Jayaram, U., Wang, Y. and Lyons, K., 2000, "CORBA-based Collaboration in a Virtual Assembly Design Environment." The ASME Design Engineering Technical Conferences 2000 (DETC 2000/CIE-14585).

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.

Jayaram, S., Vance, M. J., 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**, pp. 72.

Jayaram, U., Tirumali, H. and Jayaram, S., 2000, "A Tool/Part/Human Interaction Model for Assembly in Virtual Environments." The ASME Design Engineering Technical Conferences 2000 (DETC 2000/CIE-14584).

Johnson, T. C., 2000, A General Virtual Environment for Part Assembly Method Evaluation. Department of Mechanical Engineering. Ames, Iowa State University.

Kim, C., 2002, Collision Detection References, http://www.vrac.iastate.edu/DEERE/p3.3/pdf/collision_detection.pdf.

Larsen, E., Gottschalk, S., Lin, M. C. and Manocha, D., 1999, "Fast Proximity Queries with Swept Sphere Volumes." Technical Report TR99-018, Department of Computer Science, University of North Carolina. http://citeseer.nj.nec.com/larsen99fast.html.

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., pp. 401-408.

Mirtich, B., 1998, "Rigid Body Contact: Collision Detection to Force Computation." MERL Technical Report, TR-98-01, pp. 8-10.

Mirtich, B., 1998, "V-Clip: fast and robust polyhedral collision detection." ACM Transactions on Graphics **17(3)**, pp. 177-208.

Morvan, S. M. and Fadel, G. M., 1996, "IVECS: An Interactive Virtual Environment for the Correction of .STL files." ASME Design Engineering Technical Conferences 1996 (DETC 1996/DFM-1305).

Taylor, F., Jayaram, S. and Jayaram, U., 2000, "Functionality to Facilitate Assembly of Heavy Machines in a Virtual Environment." The ASME Design Engineering Technical Conferences (DETC 2000/CIE-14590).