Evaluation and Application of Algorithms For a Hybrid Environment System

[me.jpg]

Benjamin C. Lok

Department of Computer Science

University of North Carolina at Chapel Hill

Chapel Hill, NC 27599-3175

Research Advisor: Dr. Frederick P. Brooks, Jr.

ABSTRACT

Suppose one has a virtual model of a car engine and wants to use an immersive virtual environment (VE) to determine whether both a large man and a petite woman can readily replace the oil filter. This real world problem is difficult to solve efficiently with current modeling, tracking, and rendering techniques. Hybrid environments, systems that incorporate real and virtual objects within the VE, can greatly assist in studying this question. In this paper we describe new algorithms for generating virtual representations, *avatars*, of dynamic real objects at interactive rates and enabling virtual objects to interact with and respond to the real-object avatars. This allows dynamic real objects, such as the user, tools, and parts, to be visually and physically incorporated into the VE. The algorithms use image-based object reconstruction and a volume-querying mechanism to detect collisions and to determine plausible collision responses between virtual objects and the real-time avatars.

We then evaluate the algorithms from various standpoints:

- (Engineering) – We present an implementation of the reconstruction and collision detection algorithms in a prototype system
- (Theoretical) – We conduct performance and error analysis for the algorithms.
- (Usability) – Beyond theory though, are hybrid environments even practically useful for VE tasks? We conducted a user study that evaluated the hybrid environments' effect on VE task performance and sense-of-presence.
- (Applicability) – We looked to evaluate hybrid environments in the context of a real-world task.

The study showed that for spatial cognitive manual tasks, hybrid environments provide a significant improvement in task performance measures. Also, participant responses show promise of improving *sense-of-presence* over customary VE rendering and interaction approaches.

We detail our beginning collaboration with NASA Langley Research Center to apply the hybrid environment system to a satellite payload assembly verification task. In an informal case study, NASA LaRC payload designers and engineers conducted common assembly tasks on payload models. The results suggest that hybrid environments could provide significant advantages for assembly verification and layout evaluation tasks.

INTRODUCTION

Motivation

Conducting design evaluation and assembly feasibility evaluation tasks in immersive virtual environments (VEs) enables designers to evaluate multiple designs more efficiently than if mock-ups are built and more thoroughly than can be done from drawings. Design review has become one of the major productive applications of VEs [1]. Virtual models can be used to study: 1) can an artifact readily be assembled? and 2) can repairers readily service it? The ideal VE would be visually identical to the real task. In the assembly verification example, parts and tools would have mass, feel real, and handle appropriately. The participant would naturally interact with the virtual world, and in turn, the virtual objects would respond to the participant's action appropriately [2].

Obviously, current VEs are far from that ideal system. Indeed, not interacting with every object as if it were real has distinct advantages, as in dangerous or expensive tasks. In current VEs, almost all objects in the environment are virtual, but both assembly and servicing are hands-on tasks. The principal drawback of virtual models – that there is nothing there to feel, to give manual affordances, and to constrain motions – is a serious one for these applications. Simulating a wrench with a six degree-of-freedom wand, for example, is far from realistic, perhaps too far to be useful. Imagine trying to simulate a task as basic as unscrewing an oil filter from an engine in such a VE!

Getting shape, motion, and inputs from real objects requires specific development for modeling, tracking, and interaction. Every possible input, action, and model for all objects, virtual and real, needs to be defined, developed, and implemented.

Also the visual representations of these objects within the VE, their *avatars*, are usually stylized and not visually faithful to the object itself. We extend our definition of avatar to include a virtual representation of *any* real object. Ideally, these real-object avatars are registered in look, form, and function with the real object.

We believe a *hybrid environment* system, one that could handle *dynamic real objects*, would be effective in providing natural interactivity and visually-faithful self-avatars. We define *dynamic objects* as real objects that can change shape and appearance. We define *incorporating real objects* as generating avatars – registered with their real object counterpart – that interact with purely virtual objects. Such a system would allow designers to see if there is enough space to reach a certain location or train people in assembly with real parts, tools, and handling the physical variability among participants.

Approach

First, we developed a hybrid environment system that uses camera-based reconstruction algorithms to generate real-time virtual representations of real objects. Next, we developed algorithms to use the virtual representations in virtual lighting and in physically-based mechanics simulations. In a sense, there is a merging of two spaces, the physical space (real objects) and virtual space (corresponding virtual objects). The participant sees, handles, and feels real objects while interacting with virtual objects.

Then, we looked at applicability by conducting a user study on whether hybrid environments provide any benefit for typical VE tasks. The user study results show a

statistically significant improvement in task performance measures for interacting with real objects within a VE compared to interacting with virtual objects.

Finally, we looked at the usability of the system in a case study on applying the hybrid environment be applied to a real world task. The participants' experiences anecdotally showed the effectiveness of handling real objects while interacting with virtual objects.

PREVIOUS WORK

Incorporating Real Objects into VEs

Modeling. Prebuilt models are usually not available for specific real objects. Making measurements and then using a modeling package is laborious for complex static objects, and near impossible for capturing all the degrees of freedom of complex dynamic objects.

Automated capture systems assist in generating models from real objects. The 3-D Tele-Immersion work uses dense-stereo algorithms to create virtual representations of participants for tele-communication applications [3]. Matusik, *et al*., presented an image-based visual hull algorithm, "Image Based Visual Hulls" (IBVH), that uses image-based rendering to calculate the visual hull at interactive rates. Their work also provides methods to compute visibility, coloring, and polygonal meshes of the visual hull [4][5]. Our algorithm to recovering real object shape is similar.

Registration. The most common approach to registering a virtual representation and the real object is to employ tracking systems. Devices, using magnetic fields, acoustic ranging, optical readings, retro-reflectors or gyros, are attached to the object and the sensor's reports are used to transform the virtual models. Hoffman attached a tracker to a real plate to register a virtual model of a plate that was rendered in a VE [6], and this allowed the participant to handle a real plate where the virtual plate appeared. Commercial products include the Immersion Corporation's Cyberglove for hand tracking, and Measurand's ShapeTape, a flexible curvature-sensing device that reports its form.

Image-based algorithms, such Kanade's Virtualized Reality [7], IBVH, and this work, capture object motion by computing object representations anew from camera images.

Interactions. Collision detection between virtual objects is an area of *vast* research. Highly efficient and accurate packages, such as Swift++, detect collisions between polygonal objects, splines, and surfaces [8]. Hoff uses graphics-hardware accelerated functions to solve for collisions and generate penetration information [9]. Other work on collision detection between real and virtual objects first created geometric models of the rigid-body real objects and then used standard collision detection approaches [10].

Avatars and Interactions in VEs

The user is represented within the VE with a self-avatar, either from a library of self-avatar representations, a generic self-avatar, or no self-avatar. A survey of VE research shows the most common approach is a generic self-avatar – literally, one size fits all [1].

Avatars are typically represented with stylized virtual human models, such as those provided in commercial packages. Although these models contain a substantial amount of detail, they usually do not visually match a specific participant's appearance. Previous research hypothesizes that this misrepresentation of self is so detrimental to VE effectiveness, it will reduce how much a participant believed in the virtual world, his

sense-of-presence [11]. Usoh concludes, "substantial potential presence gains can be had from tracking all limbs and customizing avatar appearance [12]."

Inputs to the VE are traditionally accomplished by translating hardware actions, such as button pushes or glove gestures, to actions such as grasping [13]. Commercial interaction devices include a tracked articulated glove with gesture recognition or buttons (Immersion's Cyberglove), tracked mouse (Ascension Technology's 6D Mouse), or tracked joystick with multiple buttons (Fakespace's NeoWand).

Studies have also been done on interaction devices, techniques, such as 3-D GUI widgets and physical interaction [14], and specifically engineering real objects for VE input, such as augmenting a doll's head with sliding rods [15].

REAL OBJECT RECONSTRUCTION

The reconstruction algorithm was originally presented at the ACM Symposium on Interactive 3D Graphics 2001 [16].

Introduction. We present a real-time algorithm for computing the visual hull of real objects that exploits the tremendous recent advances in graphics hardware. The visual hull technique examines only the silhouettes of the real objects, viewed from different locations. The projection of a silhouette image divides space into a volume that contains the real objects, and a remaining volume that does not. The intersection of the projections of silhouette images approximates the object shape [17]. Along with the IBVH work, this algorithm is one of the first for real-time object reconstruction.

Capturing Real Object Shape

The reconstruction algorithm, takes multiple, live, fixed-position video camera images, identifies newly introduced real objects in the scene (*image segmentation*) and then computes a novel view of the real objects' shape (volume-querying).

Image Segmentation. We assume that the scene will be made up of static background objects and foreground objects that we wish to reconstruct. The goal of this stage is to identify the foreground objects in the camera images of the scene.

We use the image segmentation technique of *image subtraction with thresholds* for extracting the objects of interest. When a live frame is captured, each pixel is compared against a reference image pixel of an "empty" scene (only the background). The pixels whose differences are greater than a threshold (to reduce the effect camera images noise) are labeled *object pixels* correspond to newly introduced objects. This is done for each camera at every frame and produces a set of object pixels ($S(O_i)$). The visual hull is the intersection of the projected right cones of the 2-D object pixels.

Volume-Querying. Given the object-pixels from image segmentation, we want to view the visual hull of the real objects. To do this, we use a method we call *volume-querying*, a variation on standard techniques for volume definition given boundary representations. Volume-querying asks, g*iven a 3-D point (P), is it within the visual hull (VH) of a real object in the scene? P* is within the visual hull iff for each camera *i* (with projection matrix M_C), *P* projects onto an object pixel.

$$P \ni \sim VH_{object} \text{ iff } \forall\, i \text{ such that } M_{C,\,i} * P \ni S(O_i). \quad (1)$$

To render the visual hull from a novel viewpoint, the view frustum volume is volume-queried. In effect, this asks which points in the view frustum are within the visual hull.

Accelerating Volume-Querying with Graphics Hardware. The graphics-hardware-accelerated functions of projected textures, alpha testing, and stencil testing in conjunction with the depth, stencil, and frame buffers are used for volume-querying.

After image segmentation, each camera's image, with the corresponding object-pixel data stored in the alpha channel (pixel alpha=1 for object pixels, else 0), is loaded into a texture. The camera image color values are not used in generating object shape.

For *P* to be within the visual hull, *P* must project onto an object pixel in each camera. Thus when rendering *P* with projected textures, *P* must be textured with an object pixel from each camera. *P* is rendered *n* times, and when rendering for the *i*th time, camera *i*'s texture is used, and the texture matrix is set to the $M_{C,i}$. To apply a texel only if it is an object pixel, an alpha test to render texels with alpha = 1 is enabled.

A pixel's stencil buffer value is used to count the number of cameras that projected an object pixel onto *P*, and is initialized to 0. If *P* is textured by an object pixel from camera *i*, the pixel's stencil buffer is incremented. Once all *n* textures are projected, the stencil buffer will contain values [0, *n*]. A pixel is within the visual hull iff its stencil value is equal to *n*. The stencil buffer is then cleared of all pixels whose stencil value < *n*. The depth buffer value holds the distance of *P* from the novel viewpoint.

As the view frustum volume is continuous, we sample the volume with a set of planes perpendicular to the view direction, and completely filling the viewport. The planes are volume-queried from front to back. Each plane is rendered *n*+1 times, once with each camera's object-pixel map projected and once to clear pixels with a stencil value < *n*. These correspond to points on the plane that are within the visual hull. The frame and stencil buffers are not cleared between planes, and the depth buffer is the volume-sampled visual hull first visible surface from the novel viewpoint.

The number and spacing of the planes are user-defined. Given the resolution and location of the input cameras, we sample the volume with 1.5 cm spacing between planes for a meter in front of the user. By only volume-querying points within the view frustum, we only test elements that could contribute to the final image.

Capturing Real Object Appearance

Volume-querying only captures the real object shape. To capture the real object's appearance from the participant's point of view, a lipstick camera with a mirror attachment was mounted onto the HMD. Because of the geometry of the fixture, this camera had a virtual view that was essentially the same as the participant's. The image from this camera textures the visual hull. This particular camera choice finesses a set of difficult problems of computing the correct pixel color for the visual hull, which involves accounting for visibility and lighting. Figure 1 is a screenshot from the system.

This approach is not well suited for rendering other than from the participant's point of view. Coloring approaches are discussed in the original paper, but the results are far from satisfactory. The IBVH algorithm by Matusik computes the model and visibility and is a better for reconstruction from viewpoints other than the participant's [4][5].

Combining with Virtual Object Rendering

During the plane-sweeping step, the planes are rendered and volume-queried in the same coordinate system as used to render the VE. Therefore rendering the virtual objects into the same frame buffer and depth buffer correctly resolves occlusions between real objects and virtual objects. The real-object avatars are visually composited with the VE as shown in Figure 2. The real-object avatars can also be used lighting and shadowing between real and virtual objects, and were covered in the original paper.

[FIGURE 1 AND FIGURE 2]

Analysis

Performance. The algorithm's overall work is the sum the image segmentation and volume-querying work. This analysis does not take into account the time and bandwidth of capturing new images, interprocessor communication, and VE rendering.

For each frame, the image segmentation work is composed of subtracting each camera image pixel from a background pixel, and comparing the result against a threshold value. Given n cameras with u x v resolution, $u*v*n$ subtract and compares are required.

The volume-querying work has both a graphics transformation and a fill rate load. For n cameras, rendering l planes with u x v resolution and divided into an i x j camera-distortion correction grid, the geometry transformation work is $(2(n*i*j)+2)*l$ triangles per frame. Volume-querying each plane computes $u * v$ point volume-queries in parallel. Since every pixel is rendered $n+1$ times per plane, the *fill rate* = $(n+1)*l*u*v$ per frame.

Accuracy. The final image of the visual hull is a combination of image segmentation, volume-querying, and visual hull sampling. How closely the final rendered image of the real-object avatar matches the actual real object has two separate components: how closely the shape matches, and how closely the appearance matches.

The primary source of error in shape between a real object and its real-object avatar is due to the visual hull approximation of the real object's shape. Fundamental to using the visual hull approaches, errors in real object shape approximation enforces a lower bounds of overall error, regardless of other sources of error [18].

Image segmentation errors (mislabeling a pixel as an object or background pixel) results from foreground objects similar in color to background objects, areas of high spatial frequency in the background, and changes in lighting. Segmentation errors incorrectly segment the volume (inside or outside the visual hull), but do not contribute to errors in the visual hull location.

The next error source is how closely the virtual volume that an object pixel sweeps out matches the physical space volume. This depends on the accuracy of the calculated intrinsic and extrinsic parameters. With a 1 m^3 reconstruction volume, camera rotation and resolution are the major factors that affect volume-querying accuracy. For example, 1° of rotational error results in 5.75 cm error in the reconstruction volume.

The camera resolution determines the minimum size of a foreground object to be reconstructed. The largest distance from a camera to a point in the reconstruction volume is 3.3 m. Using one field of the NTSC-resolution cameras (720x243) with 24° FOV lenses, a pixel sweeps out a pyramidal volume with at most a base of 0.58 cm by 0.25 cm.

The effect of camera calibration error on visual hull *location* is a bit more difficult to quantify, as this type of error would cause object pixels to sweep out a volume not

registered with the physical space. It will also shift the projection of an object pixel, but this does not necessarily change the location of the visual hull.

The head tracker's noise, sub-millimeter in position and 0.1° in rotation, was not a significant component in reconstruction error. The primary factor that affects the rendering the visual hull is the spacing between the planes.

Our Experience: We attempt to reduce segmentation errors by draping dark cloth on surfaces to reduce high spatial frequency areas, keeping lighting constant and diffuse, and using foreground objects that were different in color from the background. Our Sony DFW-500 cameras had about a 2% color variation for the static cloth draped scene. The cameras are placed as close to the working volume as possible. There is an estimated 1 pixel of error for the rotation parameters and sub-millimeter error for the position parameters. We estimate 0.5 cm error for the center of the reconstruction volume is the lower bound for the certainty of the results for volume-querying a point. The plane spacing was 1.5 cm in the reconstruction volume.

The reconstructed shape is texture-mapped with the image from HMD mounted camera. The camera image was hand-tuned with interactive GUI sliders to keep the textured image registered to the real objects. We did not calibrate this front camera. We do not have an estimate for the error in visual hull appearance.

Other sources of error include: the lack of camera synchronization, system latency, and the variability in the position of the participant's head in the HMD. The most significant of these is the end-to-end system latency, estimated to be 0.3 seconds. The magnitude of the latency was such that participants recognized the lag and its effects on their ability to interact with virtual and real objects.

Implementation

Hardware. The reconstruction algorithm has been implemented in a system that reconstructs objects within a 5' x 4' x 3' volume above a tabletop. The system used three wall-mounted NTSC cameras (720x486 resolution) and one camera mounted on a Virtual Research V8 HMD (640x480 resolution). One camera was directly overhead, one camera to the left side of the table, and one at a diagonal about three feet above the table. Lab space and maintainability constrained the cameras' positions.

To compensate for the two fields, reconstruction was always done on the same field – field zero was arbitrarily chosen. While this increased the reconstruction error, latency was reduced and dynamic objects exhibited less shearing. The participant was tracked with the UNC HiBall, a scalable wide-area optical tracker mounted on the HMD.

The four cameras are connected to Digital In – Video Out (DIVO) boards on an SGI Reality Monster system. Whereas PC graphics cards could handle the transformation and pixel fill load of the algorithm, the SGI's video input capability, multiple processors, and high memory-to-texture bandwidth made it a better solution during initial development.

In implementation, the camera images contain non-linear distortions that the linear projected-texture hardware cannot process. Each plane is subdivided into a regular grid, and undistorted texture coordinates at the grid points are computed in software using a standard camera model. We have observed that dividing the plane into a 5 x 5 grid for undistorting the camera image improves visual hull shape accuracy.

In the past three years, other multiple camera algorithms have been implemented on a dedicated network of commodity PCs. With the increase of PC memory, bus, and device

I/O bandwidth, a PC based system is now a viable solution and would also benefit from a short development cycle, speed upgrades, and additional features for new hardware.

We used five SGI graphics pipes: a *parent pipe* to render the VE and assemble the reconstruction results, a *video pipe* to capture video, two *reconstruction pipes* for volume-querying, and a *simulation pipe* to run simulation and collision detection. The reconstruction was done in a 320x240 window to reduce the fill rate requirements. The results were scaled to 640x480, which is the VE rendering resolution.

Performance. The implemented system runs on an SGI Reality Monster, and runs at 15-18 FPS for 1.5 cm spaced planes for 0.7 m deep (about 50 planes) in the novel view volume. The total work is $15.7 * 10^6$ subtracts and segmentation threshold tests per second, $0.23 * 10^6$ triangles per second are perspective-transformed, and the fill rate is $0.46 * 10^9$ per second. The latency is estimated at about 0.3 of a second.

The SGI can transform about $1.0 * 10^6$ triangles per second and has a fill rate of about $0.6 * 10^9$ pixels per second. For comparison, the current latest consumer graphics card, the nVidia GeForce4, can transform about $75.0 * 10^6$ triangles per second and has a fill rate of $1.2 * 10^9$ pixels per second. The fill rate requirements limits the number of planes with which we can sample the volume, which then limits reconstruction accuracy.

Accuracy Summary. For our setup, the overall total error in the visual hull shape is estimated at 0.5 cm and the rendering of the visual hull at 1.5 cm. One practical test we used was to move our hand with a finger (about 1 cm in diameter) extended around the reconstruction volume. We examined the reconstruction width of the finger to observationally evaluate error. The finger reconstruction was relatively constant throughout most of the working volume. This is inline with our estimates of 0.5 cm error for the visual hull shape, and 1.5 cm error for rendering the visual hull.

Advantages. The hardware-accelerated reconstruction algorithm benefits from the improvements in graphics hardware. It also permits using graphics hardware for detecting intersections between virtual models and the real-objects avatars (see later section).

The participant is free to extemporaneously bring in other real objects and naturally interact with the virtual system. For example, we implemented a VE with a virtual faucet and particle system. We observed participants cup their hands to catch the water, hold objects under the stream to watch particles flow down the sides, and comically try to drink the synthetic water. Unencumbered by additional trackers and intuitively interacting with the VE, participants exhibit uninhibited exploration.

Disadvantages. Sampling the volume with planes gives this problem $O(n^3)$ complexity. Large volumes would force a tradeoff between accuracy and performance.

Visibility and coloring, assigning the correct color to a pixel considering obscuration, is not handled well. This is not a problem since we are interested in a 1st person view and use an HMD-mounted camera for a high-resolution texture map. For novel viewpoint reconstruction, these are important issues to resolve.

COLLISION DETECTION

The *collision detection* and *collision response* algorithms, along with the lighting and shadowing rendering algorithms, enable the real objects to be dynamic inputs to

simulations and provide a natural interface with the VE. That is, participants would interact with virtual objects the same way as if the environment were real.

For example, we will later show in a participant parting a virtual curtain to look out a virtual window. The interaction between the real hand and virtual cloth involves first upon detecting the collision between hand and cloth, and then upon the cloth simulation's appropriately responding to the collision. Collision detection occurs first and computes information used by the application to compute the appropriate response.

The laws of physics resolve collisions between real objects. Standard collision detection packages handle collisions between virtual objects. We present an image-space algorithm to detect and allow the virtual objects to plausibly respond to collisions with real objects. We do not handle virtual objects affecting real objects due to collision.

Real Object Visual Hull – Virtual Model Collision Detection

Overview. Since the reconstruction algorithm does not generate a geometric model of the visual hull, we needed new algorithms to detect collisions between the real-object avatars and virtual objects. Similar to how the object reconstruction algorithm volume-queries the novel view frustum, the collision detection algorithm tests for collisions by volume-querying with the virtual objects primitives.

The *real-virtual* collision detection algorithm takes as inputs a set of *n* live camera images and virtual objects defined triangles. It outputs a set of points (CP_i) on the virtual object surface that are within a real-object avatar. It also estimates the following: point of first contact on the virtual object (CP_{obj}) and the visual hull (CP_{hull}), recovery vector (V_{rec}) and distance (D_{rec}), and surface normal at the point of visual hull contact (N_{hull}).

Assumptions. A set of simplifying assumptions makes interactive-time real-virtual collision detection a tractable problem.

1. *Only virtual objects can move or deform as a consequence of collision.*
2. *The real-object avatar and virtual object are considered stationary when resolving collisions.* With no real object motion information available, the algorithm cannot determine *how* or *when* the real and virtual objects came into collision. It simply suggests a way to move the virtual object out of collision.
3. *There is at most one collision between a virtual and real-object avatar at a time.* For multiple intersections, we heuristically choose one as the point of contact.
4. *The real objects that contribute to the visual hull are treated as a single object*. The system cannot distinguish between the real objects that form a visual hull.
5. *Collisions are detected relatively shortly after a virtual object enters the visual hull, and not as the virtual object is exiting the visual hull.* This assumes the simulation time step (frame rate) is fast compared to the dynamics of the objects.

Detecting Collisions. Collision points, CP_i, are points on the surface of the virtual object that are within the visual hull. As the virtual surfaces are continuous, the set of collision points is a sampling of the virtual object surface.

The real-virtual object collision detection algorithm uses volume-querying. In novel viewpoint reconstruction, we volume-queried points in the view frustum volume to determine which were inside the visual hull. Collision detection volume-queries with the triangles defining the virtual object's surface to determine if any parts of the surface is inside the visual hull. If any part of a triangle lies within the visual hull, the object is intersecting a real-object avatar, and a collision has occurred.

Everything else (the textures, stencil testing, etc.) is the same. A collisions occurs if any pixel has a stencil buffer = *n*, thus indicated some part of a triangle, and in turn a virtual object, is within the visual hull. If the triangle is projected 'on edge' during volume-querying, the sampling of the triangle surface during scan-conversion will be sparse, and collision points could be missed. No one viewpoint will be optimal for all triangles. Thus, each triangle is volume-queried in its own viewport, such that the triangle's projection maximally fills the viewport.

After all the triangles are volume-queried, the frame buffer is read back. The collision pixels (stencil buffer value = *n*) are unprojecting from screen space coordinates (u, v, depth) to world space coordinates (x, y, z) to produce the 3-D coordinates of a collision point. These 3-D points form a set of collision points, CP_i, for that virtual object. As an optimization, collision detection is first done with virtual object bounding boxes, and if there are collisions, on a per-triangle test is done.

How a simulation utilizes this information is application- and even object-dependent. This division of labor is similar to current collision detection algorithms [8]. We provide tools to move the virtual object out of collision with the real object.

Recovery from Interpenetration. We present one approach to use the collision information to generate a plausible response. The first step is to move the virtual object out of collision. We estimate the point of first contact on the virtual object, CP_{obj}, with the collision point farthest from the virtual object's reference point, RP_{obj}. The default RP_{obj} is the center of the virtual object.

Our estimate to move the virtual object out of collision by the shortest distance is along a recovery vector, V_{rec} defined as the vector from CP_{obj} to RP_{obj}. This vector works well for most objects; though the simulation can specify V_{rec} for virtual objects with constrained motion, such as a hinged door, for better object-specific results. V_{rec} crosses the visual hull boundary at the *hull collision point*, CP_{hull}. CP_{hull} is an estimate of the point of contact on the visual hull, and to where CP_{obj} will be backed out.

To find CP_{hull}, V_{rec} is searched from RP_{obj} towards CP_{obj} for the first point within the visual hull. This is done by volume-querying an isosceles triangle *ABC*, $A = CP_{obj}$ and the base, *BC*, is bisected by V_{rec}. Angle *BAC* (at CP_{obj}) is set small (10°) so that *AB* and *AC* intersects the visual hull near CP_{hull}, and the height is set relatively large (5 cm) so the triangle base is likely to be outside the visual hull. *ABC* is volume-queried in the entire window's viewport, and from a viewpoint along the triangle normal and such that V_{rec} lies along a scan line. CP_{hull} is found stepping along the V_{rec} scan line for the first pixel within the visual hull. Unprojecting the pixel from screen to world space yields CP_{hull}.

The *recovery distance*, D_{rec}, is the distance between CP_{obj} and CP_{hull}, and is the distance along V_{rec} required to move CP_{obj} outside the visual hull. It is not necessarily the *minimum separation distance*, as is found in some collision detection packages [8][9].

To compute the visual hull collision point surface normal, N_{hull}, we locate 4 points on the visual hull surface near CP_{hull} . Stepping along *BA* and *CA* and finding where they intersect the visual hull boundary determines points *I* and *J*. A second triangle, *DAE*, is constructed that is similar to *ABC* and lies in a plane roughly perpendicular to *ABC*. Points *K* and *L* are located by stepping along *DA* and *EA*. N_{hull} is the cross product of *IJ* and *KL*.

[FIGURE 3]

Analysis

Performance. Given n cameras, virtual objects with m triangles, and a u x v viewport in a x x y window: the geometry transformation cost is ($n * m$), fill rate cost is ($n*m*u*v$)/2, and ($x*y$) pixel readbacks and compares per frame. Our window and curtain hybrid environment had 720 triangles that made up the curtains. We used 10 x 10 viewports in a 400 x 400 window for collision detection, which ran at 6 FPS. The collision detection work was 13,000 triangles transformed per second, and 648,000 pixels per second fill rate, and 160,000 pixel readbacks and compares. The collision response work was 2 triangles and 480,000 pixels fill rate per virtual object in collision. As this was a first implementation, there are many optimizations that should improve the performance.

Accuracy. The collision detection accuracy depends on image segmentation, camera models (previously discussed), and the volume-querying viewport. The size of the viewport is proportional to the volume-querying spatial sampling accuracy, and inversely proportional to the speed and number of triangles that can be queried in a single pass. The accuracy of collision detection for a u x v resolution viewport (if $u = v$, viewport layout is easier) and a triangle with x x y bounding box (in world space) is x/u by y/u.

The size of virtual object triangles will vary, but typical tabletop objects had triangles less than 2 cm, which would have 0.2 cm x 0.2 cm collision point detection error. For example in the cloth system had a collision detection resolution of 0.75 cm x 0.3 cm.

For collision response, the accuracy of the CP_{hull} point impacts D_{rec} and N_{hull}. The error in finding CP_{hull} along the V_{rec} is the length of triangle *ABC*'s major axis divided by the horizontal length of collision response window (assuming a square window). With a 400 x 400 rendering window, this results in .0125 cm error for detecting CP_{hull}. The accuracy of N_{hull}, depends on the surface topology (effected by camera resolution), the distance from these points to CP_{hull}, and the distance from CP_{hull} to CP_{obj}. The magnitude of these errors is smaller than the error in the visual hull location and visual hull shape.

[FIGURE 4]

Implementation. Figure 4 is a sequence of frames of a user pushing aside a curtain with his hands. This shows the use of the algorithm with a deformable virtual object with constrained motions (specified V_{rec} to the collision response algorithm). Now, when trying to move the cloth nodes out of collision, the motion is primarily in the constrained vector direction. We have also prototyped particle systems, lighting, and shadowing simulations interacting with the real-object avatars.

USER STUDY

Motivation. Two components that the implemented hybrid environment provides are interacting with real objects and visually faithful self-avatars. But does that even benefit VE tasks? We conducted a study to identify the effects of interaction methodologies and avatar visual fidelity on task performance and *sense-of-presence* while conducting a cognitive manual task. Compared to virtual objects and generic self-avatars,

- Does interacting with real objects improve task performance?
- Does seeing a visually faithful self-avatar improve sense-of-presence?

Experiment

Design Decisions. In devising the task, we sought to abstract tasks common to VE design applications. Through surveying production VEs [1], we noted that a substantial number of VE goals involve participants doing spatial cognitive manual tasks. For example, in layout applications, users evaluate different configurations and designs.

The task we designed is similar to, and based on, the block-design portion of the Wechsler Adult Intelligence Scale (WAIS). Developed in 1939, the WAIS is a test widely used to measure intellectual quotient, IQ [19]. The block-design portion measures reasoning, problem solving, and spatial visualization.

Description. Participants manipulated a number of identical 3"wooden blocks to make the top face of the blocks match a target pattern. The faces represented the possible quadrant-divided white-blue patterns. There were two sizes of target patterns, *small* four block patterns in a 2x2 arrangement, and *large* nine block patterns in a 3x3 arrangement.

Design. The user study was a between-subjects design. Each participant performed the task in a real space environment (RSE), and then in one of three VE conditions. The independent variables were the interaction modality (real or virtual blocks) and the self--avatar fidelity (generic or visually faithful). The three VE conditions were:

- Virtual objects with generic self-avatar (purely virtual environment)
- Real objects with generic self-avatar (hybrid environment)
- Real objects with visually faithful self-avatar (vis-faithful hybrid environment)

Experiment Conditions. *Real Space Environment (RSE)* - participants manipulated nine wooden blocks inside a rectangular 36" x 25" x 18" enclosure. *Purely Virtual Environment (PVE)* – participants wore Fakespace Pinchgloves, each tracked with Polhemus Fastrak trackers, and a Virtual Research V8 HMD. Using pinching gestures, the participant manipulated virtual blocks with his self-avatar. Hybrid Environment (HE) – participants wore yellow dishwashing gloves and the HMD. Within the VE, participants handled physical blocks, identical the RSE blocks, and saw a self-avatar with accurate shape and generic appearance (dishwashing gloves). *Visually-Faithful Hybrid Environment (VFHE)* – similar to the HE except the participants did not wear gloves. The self-avatar was visually faithful as the shape reconstruction was textured with images from a HMD mounted camera. The participant saw an image of himself, warts and all.

Virtual Environment. The VE was identical in all three of the virtual conditions (PVE, HE, VFHE). The room had several virtual objects, including a lamp, a plant, and a painting, along with a virtual table that was registered with a real Styrofoam table. The enclosure in the RSE was also rendered in the VE, but was rendered with transparency.

All the VE conditions were rendered on an SGI Reality Monster. The PVE ran on one rendering pipe with four raster managers at a minimum of 20 FPS. The HE and VFHE ran on four rendering pipes at a minimum of 20 FPS for virtual objects, and 12 FPS for reconstructing real objects. The participant wore a Virtual Research V8 HMD (640x480 resolution in both eyes) that was tracked with the UNC HiBall tracking system.

We expect a participant's RSE (no VE equipment) performance would produce the best results, as the interaction and visually fidelity were optimal. We compared how closely a participant's task performance in VE was to their RSE task performance. We compared the reported sense-of-presence in the VEs to each other. The PVE is a plausible approach with current technology. The HE evaluates the effect of real objects on task performance and presence. The VFHE adds visually faithful self-avatars.

[FIGURE 5]

Measures. For *task performance* we measured the time (in seconds) for a participant to arrange the blocks to exactly match the target pattern. The dependent variable was the difference in a participant's task performance between the RSE condition and VE condition. For *sense-of-presence*, the dependent variable was the sense-of-presence scores from the Steed-Usoh-Slater Presence Questionnaire (SUS) [20].

Finally, we conducted a debriefing interview and administered the Guilford-Zimmerman Aptitude Survey, Part 5: Spatial Orientation and the Kennedy – Lane Simulator Sickness Questionnaire.

Experimental Procedure. After completing the initial forms and questionnaires, the participant did the task in the RSE. Participants performed a series of practice patterns, three small and then three large. Next, the participant completed six timed patterns, three small and three large. Then, the participant conducted the task in a VE condition. Following a period of adaptation to the VE, the participant practiced on two small and two large patterns, and then did two small and two large timed patterns. Finally, the participants were interviewed about their impressions of and reactions to the session, and completed the final series of questionnaires.

Hypotheses. *Task Performance*: Participants who manipulate real objects in the VE (HE, VFHE) will complete the spatial cognitive manual task significantly closer to their RSE task performance than will participants who manipulate virtual objects (PVE).

Sense-of-presence: Participants represented in the VE by a visually faithful self-avatar (VFHE) will report a higher *sense-of-presence* than will participants represented by a generic self-avatar (PVE, HE).

We expect interacting with real objects improves task performance regardless of self-avatar visual fidelity and generic self-avatars would have similar effects on presence regardless whether there were real or virtual objects.

Results and Discussion

Forty participants completed the study, thirteen each in the purely virtual environment (PVE) and hybrid environment (HE), and fourteen in the visually-faithful hybrid environment (VFHE). The participants were primarily male (thirty-three) UNC undergraduate students (thirty-one). They reported little prior VE experience (M=1.37), high computer usage (M=6.39), and moderate – 1 to 5 hours a week – computer/video game play (M=2.85), on [1..7] scales. There were no significant differences between the groups. We use a two-tailed t-test with unequal variances and an α=0.05 level.

[TABLE 1 and TABLE 2]

Task Performance. For small and large patterns, both VFHE and HE task performances were significantly better than PVE task performance. The difference in task performance between the HE and VFHE was not significant at the α=0.05 level. As expected, performing the block-pattern task took longer in any VE than it did in the RSE: The PVE participants took about three **times** as long to do the task as they did in the RSE. The HE and VFHE only took about twice as long (Table 1).

For the case we investigated, *interacting with real objects provided a quite substantial performance improvement over interacting with virtual objects for cognitive manual tasks*. Although task performance in all the VE conditions was substantially worse than in the RSE, the task performance of HE and VFHE participants was significantly better than for PVE participants. There is a slight difference between HE and VFHE performance (Table 2, p=0.055). We do not have a hypothesis for this result.

Sense-of-presence. We augmented the standard Steed-Usoh-Slater Presence Questionnaire with questions that focused on the participants' perception of their avatars. Although interviews showed visually faithful self-avatars (VFHE) were preferred**,** *there was no statistically significant difference in reported sense-of-presence compared to those presented a generic self-avatar (HE and PVE).*

There were no statistically significant differences at the α=0.05 level between any of the conditions for any of the sense-of-presence questions. Slater and Usoh cautions against the use of the SUS Questionnaire to compare presence across VE conditions, but also points out that no current questionnaire supports such comparisons [11].

Other Factors. Simulator sickness and spatial ability were not significantly different between the groups at the α=0.05 level. Spatial ability was moderately correlated (r=-0.31 for small patterns, and r=-0.38 for large patterns) with performance.

Participant Interviews. Participants in all groups responded that the self-avatar, block task, head tracking, and virtual objects improved their sense-of-presence.

We noticed a trend in comments on self-avatar realism. All PVE and HE participant comments related to motion accuracy, "Everything I did with my hands, it followed." All VFHE participant comments related to visual accuracy, "[It was] just the same as in reality… I didn't even notice my hands." We hypothesize *kinematic fidelity of the self-avatar is more important than visual fidelity for sense-of-presence*.

Most HE and VFHE participants noted the reconstruction system's noise and lag. Some of the HE and VFHE participants noted that the tactile feedback of handling real objects increased their sense of presence, while 43% of the PVE participants felt that the virtual blocks reduced their sense-of-presence. Almost all the participants (93%) of the PVE felt the interaction was unnatural, compared to only 13% in the HE. VFHE participants became comfortable interacting with the VE significantly more quickly (1.50 to 2.36 practice patterns) than PVE participants ($T_{26} = 2.83$, p=0.0044).

Interesting Results. Rotating the block, followed by selection and placement of blocks, dominated the difference in times between VE conditions. Both were improved through the natural interaction, motion constrains, and tactile feedback of real blocks.

Recall that while the PVE participant made a pinching gesture to pick up a block, visually they saw the avatar hand grasp a virtual block. This misregistration caused 25% of the participants to forget the pinching mnemonic and try a grasping action (which at times did not register with the pinch gloves).

Conclusions. Interacting with real objects significantly improves task performance over interacting with virtual objects in spatial cognitive tasks, and more importantly**,** it brings performance measures closer to that of doing the task in real space. *Handling real objects makes task performance and interaction in the VE more like the actual task.*

Motion fidelity is more important than visual fidelity for self-avatar believability. We believe that a visually faithful self-avatar is better than a generic self-avatar, but from a sense-of-presence standpoint, the advantages do not seem very strong.

CASE STUDY: NASA COLLABORATION

Driving Problems. To evaluate the potential of this technology in a real world task, we applied our prototype to an assembly verification task, we have begun collaborating with the NASA Langley Research Center (LaRC).

NASA LaRC payload designers are interested in examining models of payload subsystems for assembly verification and assembly training. They want to discern possible assembly, integration, and testing problems early in the project development cycle. Since different subsystems are separately subcontracted out, the integration stage always generates compatibility and layout issues. Layout issues result in schedule delays, equipment redesign, or makeshift engineering fixes.

Early in development, the payload designs are stored as CAD models, and the assembly procedure is a step-by-step instruction list. Later in development, simplified physical mock-ups are manufactured for design verification and layout. We believe hybrid VEs can be an effective tool between these stages to enable designers to test configurations using the final assembly personnel, real tools and parts.

Payload Spacing Experiment. We received CAD models of a photon multiplier tube (PMT), a weather imaging satellite subsystem, currently under development. Next, we abstracted a task similar to common assembly steps, such as attaching components and cables to connectors. The PMT model, along with two other payloads (payloads A and B), was rendered in the VE. The hybrid system performed collision detection between the virtual payloads and the real-object avatars. The task was to screw a cylindrical shield (mocked-up as a PVC pipe) into a receptacle and then plug a power connector into an outlet inside the shield (Figure 6). The task was to determine how much space was required between the *top* of the PMT box and the *bottom* of payload A.

[FIGURE 6 AND FIGURE 7]

Experimental Procedure. Four NASA LaRC payload designers participated in the case study. Before attempting the task, we provided task information in approximately the same manner as they receive it in actual design evaluation and surveyed them on the space needed between the PMT and payload A. Each participant then performed the pipe insertion and power cable attachment procedure in the hybrid system. After a period of VE adjustment, participants picked up the pipe and eased it into the center cylindrical assembly while trying to avoid colliding with any of the virtual payloads. After the pipe was lowered into the cylindrical shaft of the PMT, they snaked the power cord down the tube and inserted it into the outlet.

As the participant asked for more or less space, the experimenter could dynamically adjust the space between the PMT and payload A. With this interaction, different spatial configurations of the two payload subassemblies could be quickly evaluated.

Results. Given that the pipe had a length of 14 cm and a diameter of 4 cm:

[TABLE 3]

Participant #1 was able to complete the task without using any tool, as the power cable was stiff enough to force into the outlet. Since an aim was to impress upon the participants the possibility of requiring unforeseen tools in assembly or repair, we used a more flexible cable for the remaining participants. While trying to insert the power cable, participants #2, 3, and 4 asked for a tool to assist. They were handed a set of tongs and were then able to complete the power cable insertion task. This required increasing the spacing between the PMT and Payload A from 14 cm to an average of 24 cm.

Whereas in retrospect it was obvious that the task would not be easily completed without a tool, none of the designers anticipated this requirement. We believe the way the assembly information was provided (diagrams, assembly documents and drawings), made it difficult for designers, even though each had substantial payload development experience, to catch subtle assembly integration issues. On average, the participants allocated 5.6 cm too little space between the payloads on their pre-experience surveys.

The hybrid VE system provided identifiable benefits over purely virtual approaches. Participants interacted with virtual objects as if they were real, including avoiding contact of the payload. "You just don't touch flight hardware". Accommodating tools extemporaneously, without additional modeling or development, enabled easy evaluation of multiple layouts, approaches, and tools. The pipe threads and cable socket provided important motion constraints that aided in interacting with these objects.

Debriefing. The participants were *extremely* surprised that both a tool and substantial additional space were required. The participants commented that the financial cost of the spacing error could range from moderate (keeping personnel waiting until a design fix was implemented) to extreme (launch delays). Time is the most precious commodity in payload development, and identifying the spacing error would save days to weeks. NASA LaRC payload designers remarked that VEs and object reconstruction VEs would be useful for assembly training, hardware layout, and design evaluation.

SUMMARY

This work investigated the methods, usefulness, and application of a hybrid environment capable of incorporating dynamic real objects. We developed real-time algorithms for generating virtual representations of real objects and collision management algorithms to handle interactions between real and virtual objects

We evaluated the algorithms from an engineering, theoretical, usability, and application standpoint. We conducted studies to examine the effects of interaction modality and avatar fidelity on task performance and sense-of-presence. We found that interacting with real objects significantly improves task performance for spatial cognitive tasks. We did not find the avatar visual fidelity affected sense-of-presence. We have begun applying our system to a NASA LaRC an assembly verification task. Initial trials show promise on the applicability of hybrid environments to aid in payload development.

ACKNOWLEDGEMENTS

I would like to thank Dr. Frederick P. Brooks, Jr. for his support and guidance. I want to acknowledge Samir Naik and Professor Mary Whitton for their invaluable contribution

to this research. Finally, I would like to thank the LINK Foundation and the University of North Carolina at Chapel Hill for the financial support.

BIBLIOGRAPHY

[1] F. Brooks. "What's Real About Virtual Reality?" *IEEE Comp. Gr. and App.*, Vol 19, No. 6, pp. 16-27. (1999)

[2] Sutherland, I. "The Ultimate Display", In *Proceedings of IFIP* 65, Vol 2, pp 506. (1965)

[3] K. Daniilidis, J. Mulligan, R. McKendall, G. Kamberova, D. Schmid, and R. Bajcsy. "Real-Time 3-D Tele-immersion", In *The Confluence of Vision and Graphics*, A Leonardis *et al*. (Eds.), Kluwer Academic Publishers, pp. 253-266. (2000)

[4] W. Matusik, C. Buehler, R. Raskar, S. Gortler and L. McMillan. "Image-Based Visual Hulls", In *Proceedings of ACM SIGGRAPH 2000*, Annual Conference Series, pp. 369-374. (2000)

[5] W. Matusik, C. Buehler, and L. McMillan. "Polyhedral Visual Hulls for Real-Time Rendering", In *Proceedings of 12th Eurographics Workshop on Rendering*, London, England, pp. 115-125. (2001)

[6] H. Hoffman. "Physically Touching Virtual Objects Using Tactile Augmentation Enhances the Realism of Virtual Environments", In *Proceedings of the IEEE Virtual Reality Annual International Symposium '98*, Atlanta GA, p. 59-63. IEEE Computer Society, Los Alamitos, California. (1998)

[7] S. Baba, H. Saito, S. Vedula, K.M. Cheung, and T. Kanade. "Appearance-Based Virtual-View Generation for Fly Through in a Real Dynamic Scene", In *Proceedings of VisSym '00* (Joint Eurographics – IEEE TCVG Symposium on Visualization). (2000)

[8] S. Ehmann and M. Lin. "Accurate Proximity Queries between Polyhedra Using Surface Decomposition", *Computer Graphics Forum (Proceedings of Eurographics).* (2001)

[9] K. Hoff, A. Zaferakis, M. Lin, and D. Manocha. "Fast and Simple 2-D Geometric Proximity Queries Using Graphics Hardware", *2001 ACM Symposium on Interactive 3-D Graphics*, pp. 145-148. (2001)

[10] D. Breen, E. Rose, R. Whitaker. "Interactive Occlusion and Collision of Real and Virtual Objects in Augmented Reality", Munich, Germany, European Computer Industry Research Center. (1995)

[11] M. Slater and M. Usoh. "The Influence of a Virtual Body on Presence in Immersive Virtual Environments", *VR 93, VR International, Proceedings of the Third Annual Conference on Virtual Reality*, London, Meckler, pp 34-42. (1993)

[12] M. Usoh, K. Arthur, *et al*. "Walking > Virtual Walking> Flying, in Virtual Environments", *Proceedings of SIGGRAPH 99*, pp. 359-364, Computer Graphics Annual Conference Series. (1999)

[13] D. Bowman and L. Hodges. "An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments", In *Proceedings 1997ACM Symposium on Interactive 3-D Graphics*. eds. by M. Cohen and D. Zeltzer, pp. 35-38. ISBN 0-89791-884-3. (1997)

[14] R. Lindeman, J. Sibert, and J. Hahn. "Hand-Held Windows: Towards Effective 2D Interaction in Immersive Virtual Environments", In *IEEE Virtual Reality*. (1999)

[15] K. Hinckley, R. Pausch, J. Goble, and N. Kassell. "Passive Real-World Interface Props for Neurosurgical Visualization", In *Proceedings of the 1994 SIG-CHI Conference*, pp 452-458. (1994)

[16] B. Lok. "Online Model Reconstruction for Interactive Virtual Environments", In *Proceedings 2001 ACM Symposium on Interactive 3-D Graphics*, Chapel Hill, N.C., 18-21, pp. 69-72, 248. (2001)

[17] A. Laurentini. "The Visual Hull Concept for Silhouette-Based Image Understanding", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 16, No. 2, 150-162. (1994)

[18] W. Niem. "Error Analysis for Silhouette-Based 3D Shape Estimation from Multiple Views", *Proceedings on International Workshop on Synthetic - Natural Hybrid Coding and Three Dimensional Imaging* (IWSNHC3DI'97), Rhodos. (1997)

[19] D. Wechsler. The Measurement of Adult Intelligence, 1st Ed., Baltimore: Waverly Press Inc. (1939)

[20] M. Usoh, E. Catena, S. Arman, and M. Slater. "Using Presence Questionnaires in Reality", *Presence: Teleoperators and Virtual Environments*, Vol. 9, No. 5, pp. 497-503. (2000)