Modeling Real Objects Using Video See-through Augmented Reality

Joohi Lee, Gentaro Hirota, Andrei State
University of North Carolina at Chapel Hill
{lee|hirota|andrei}@cs.unc.edu

Abstract

This paper presents a method for creating 3D models of real objects using video see-through augmented reality. We use a tracked probe to sample the objects' geometries and video images acquired from the head-mounted cameras to capture textures. Our system provides visual feedback by overlaying the model onto the real object in the user's field of view. This visual feedback makes the modeling process interactive and intuitive.

1 Introduction

Modeling is one of the most time-consuming tasks in computer graphics applications. Models are sometimes created from scratch, but in many cases, the objects to be modeled already exist in the real world. For example, in CAD applications, designers often want to reverse engineer real products. When creating computer animation, it is quite common for artists to model characters using clay. The clay models are then digitized as computer models so that they can be animated. Thus, creating digital copies of real objects is an important technology, often referred to as *3D digitizing*.

1.1 Related Work

Most (but not all) 3D digitizing techniques fall into three major categories:

1. **3D Reconstruction from 2D Images.** These methods exploit computer vision techniques such as structure from motion, stereo triangulation, and structure from shading [21]. They try to mimic human vision by reconstructing 3D information from 2D images. Typical strategies such as structure from motion recover 3D geometry of objects (and sometimes camera motion as well) from images taken from different viewpoints. Detecting and matching common features among multiple images is crucial for such methods [14]. (Some simple commercial products require manual assistance [13][15].) There is no specific requirement for the light sources that illuminate the objects, but specularity and transparency of the object surfaces pose significant problems for feature detection. Highly concave objects are also difficult to handle. For example, if an object has a deep hole, surface points inside the hole can only be seen from a very narrow angle (i.e. not from all camera viewpoints), making it difficult to recover accurate coordinates for those points.
2. **Structured light projection.** Instead of using natural light sources, one can project light patterns onto the real world objects [2][4][6][8]. In camera images, such patterns can be easier to detect than natural features. In most cases, the position of projectors and cameras are known, and points on object surfaces can be determined by triangulation. Widely used commercial systems [11] utilize laser stripes. In addition to using the camera for acquiring geometric patterns, a color CCD camera is often used to capture RGB colors on the objects' surfaces. The major advantage of these techniques is the high acquisition speed; some systems can sample nearly one million points per second. The acquired data is then passed on to a surface reconstruction algorithm.
   Since this method relies on the reflection of light, it cannot easily digitize specular or transparent objects. Furthermore, the concavity is even more problematic for this class of methods. A sample point can be visible (unoccluded) not only from the cameras but also from the light source (projectors).
3. **Point sampling with tracked probes.** Points are sampled by physically touching the surface of the object with a probe whose position is measured by a 3D tracking system [16]. Mechanical, electro-magnetic, ultrasonic, optical and inertial trackers can be used to measure the position of the probe.

Due to the large number of points acquired, (1) and (2) typically use a reconstruction algorithm [5] to build surface models. In the third method, points are sampled manually, hence typically fewer points than in the other methods are collected. Thus, surface models may be constructed by manually connecting the sample points.

1.2 Motivation and Contribution

Methods (1) and (2) above work well for objects with relatively simple geometry and diffuse reflective and

opaque surfaces. (2) is particularly efficient and has been used successfully in many areas. But as mentioned, neither can handle highly specular or transparent surfaces. In addition, they fail to sample points on highly concave areas.

Method (3) is immune to specularity and transparency problems since it does not rely on image sensors. Furthermore, it is suitable for sampling concave parts since it is easy to stick a thin probe into a concave area to sample a point.

However, digitizing with a tracked probe is often challenging in terms of hand-eye coordination. When collecting sample points, the user carefully touches a real object, looking at the probe's contact point on the object surface. On the other hand, one would like to see immediate visual feedback about the sampled point. But such information is usually available only on a computer screen, which one cannot see without looking away from the real object. After sampling many points, the user sees clouds of points on the computer screen, and it is hard to determine where on the real object each sampled point lies. Thus, even though the object exists in the user's real workspace, the user has to deal with extraneous virtual spaces in which the digitized models reside.

To alleviate the hand-eye coordination problem, one could set up a video camera at a carefully calibrated position and overlay the digitized models onto the captured video images of the real objects. The user will then see the correspondence between real and digitized models. However, if the user constructs a model by looking at the computer screen while manipulating the tracked probe and the real object, the user still has to mentally fuse two workspaces that differ in translation, rotation and scaling; thus hand-eye coordination continues to be impaired.

We propose an augmented reality (AR) based modeling system that removes the mental separation between real and virtual workspaces. The model being created is overlaid directly onto the user's view of the real object. Therefore the user never has to leave the (one and only) real workspace while modeling an object.

We use a head-mounted display (HMD) with video see-through AR technology, which enables virtual objects and real objects to coexist in the real workspace. Our user interface allows the user to modify (scale, deform, etc.) digital copies of real objects. One can assess the appearance of modified objects in the actual surroundings. The user can also replicate parts of the model to efficiently model objects whose shapes are bilaterally, rotationally or cyclically symmetric.

In conventional 3D scanners, the object to be digitized is usually rigidly mounted on a support. Our system tracks the rigid body motion of the real object, so the user can freely move the object undergoing modeling.

In video-see-through AR systems, video images are readily available. Our system extracts textures of real objects from video images. Using the known (tracked) poses of the head-mounted cameras and the object being modeled, the system automatically evaluates the image quality of the area of the video image available to texture each polygon of the model, and acquires only high-quality textures.

2 System Overview

Our current AR modeling system (an earlier version was described in [19]) integrates off-the-shelf components: a head mounted display, miniature video cameras, an optical tracking system, a 3D probe, foot pedals, a high-end graphics workstation, and several rigs equipped with infrared (IR) LEDs for tracking.

The following is a description of the hardware configuration of our system (Figure 1).

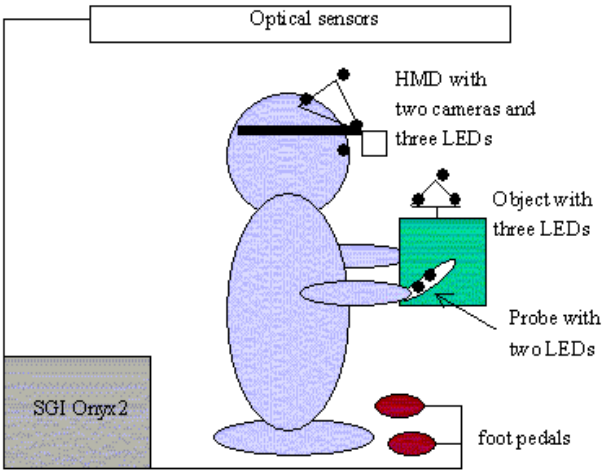

Figure 1. System Configuration.

Video see-through HMD: We mounted two miniature Toshiba IK-SM43H cameras on a stereoscopic Sony Glasstron LDI-D100B head-mounted display with SVGA (800×600) resolution. The two cameras are used to provide a stereoscopic view of the real world. Three IR LEDs are also mounted on it for the purpose of optical tracking. [10][18]

Probe: The probe has two IR LEDs to track the position of the probe tip.

Object trackers: Three IR LEDs are rigidly affixed to each of the objects to be modeled.

Optical sensors: All IR LEDs are tracked by an Image Guided Technologies Flash Point 5000 3D Optical Localizer. The sensor consists of three 1D CCD arrays.

Pedals: We use two foot pedal switches to assist with interactive operations.

Light: Various ambient light sources.

Graphics workstation: We use an SGI Onyx2 Reality Monster Graphics Supercomputer with multiple DIVO boards for real-time capture of HMD camera video streams. Our system makes use of one multi-channel graphics pipe and two R12000 CPUs.

3 System Operation

Our modeling process consists of three steps: sampling points, constructing a triangle mesh, and capturing textures. All of these are performed in an AR environment by a user wearing a HMD (Figure 2).

We first describe how points are sampled and a triangle mesh is generated. Then we discuss the use of video cameras on the HMD to obtain the textures of triangles. Generally, the user follows these three steps in this order. However, one can move back and forth between the steps. This allows the user to construct a complete model by incrementally adding small portions consisting of fully textured and *z*-buffered triangles.

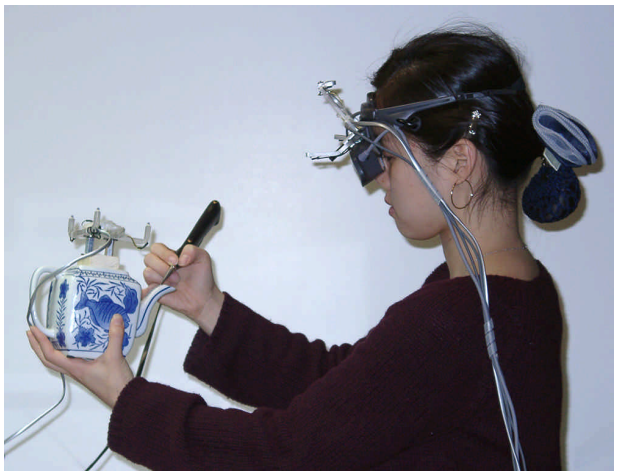

Figure 2. User with AR HMD, modeling teapot with handheld probe. Probe, teapot and HMD are equipped with IR LEDs for opto-electronic tracking.

3.1 Point Sampling

The first step of modeling in our system is to sample the object geometry.

This step is quite simple. The user picks points on the object by physically touching the surface of the real object with the (optically) tracked probe and depressing a pedal (Figure 3). When a point is sampled, the point in probe coordinates is transformed into the object's coordinate system. The object is also continually tracked by the optical tracker. As a result, the newly acquired point is "attached" to the object's surface.

Since we render the points on top of the object in the video image, the user sees both the real object and the points through the HMD. Overlaying points on the object helps the user to figure out which points have been added, to determine which part of the object needs more sampling, and finally to sample the exact point desired. Some confusion may occur if two points happen to be rendered close to each other on the screen while one point is on the front of the object and the other is on the back of the object. (At this stage in the modeling process we only have a collection of points and no polygons that could be used to occlude the back points via *z* buffering.) However, since points are displayed in stereo like everything else, the user can still see which one is in front. However, it is confusing if the points on the back face are not occluded properly by the object.

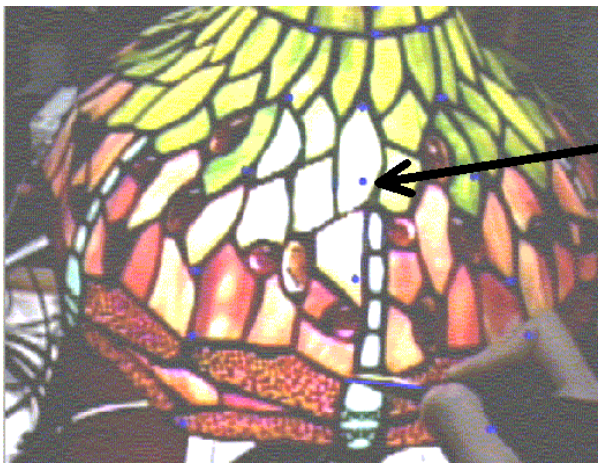

Figure 3. Sampling object geometry with the probe. The arrow indicates one of the acquired points.

As mentioned, our system allows the user to build a model incrementally, which can reduce the possible confusion. Instead of defining triangles after all points of the object have been collected, the user can work on a small area at one time. By creating triangles for part of an object before moving on to other parts, the system acquires partial information about the object geometry. Hence, when the system renders triangles into the *z*-buffer, points behind the triangles can be occluded. As long as the triangle mesh is incomplete, not all points on the back of the object can be occluded; nevertheless this technique provides a better sense of depth to the user and is therefore helpful when modeling complex objects.

Figure 4. Modeling the concave part (handle) of a teapot.

Figure 4 demonstrates that our system can handle concave objects without difficulty. Because both the real

object and the probe are tracked, the user can rotate the object and easily reach points with the probe. However, the user must be aware of the limitation of the optical tracking system, and not turn the object's or the probe's LEDs away from the optical sensors. One also needs to be cautious not to occlude the LEDs from optical sensors with body parts such as hands or fingers. Although we cannot eliminate this problem (typical for all opto-electronic trackers), we alleviate it using visual/audio feedback. The system does not accept a sample point if either the object or the probe loses tracking, and warns the user of the tracking problem, thus preventing him/her from acquiring erroneous points.

3.2 Triangle Formation

After collecting points, the user is ready to create triangles by connecting the sampled points. The point closest to the probe tip is added to a selection set when the user depresses the pedal. Once three points are selected, the system automatically adds a triangle to the model. The order of the three points is changed automatically to force the triangle to be counterclockwise from the user's point of view. Keeping every triangle in counterclockwise order is required to determine visible triangles for texture acquisition (see below) and for backface culling when rendering the finished model.

The triangle mesh is overlaid on the real object in the HMD view in the same fashion as the sampled points are drawn (Figure 5). This interactive feedback gives a good idea about how the model is being formed.

It is also possible to apply an automatic reconstruction algorithm [5] rather than manually constructing the triangle mesh. The user can invoke such an algorithm off-line and then continue on to texture acquisition with the automatically created mesh.

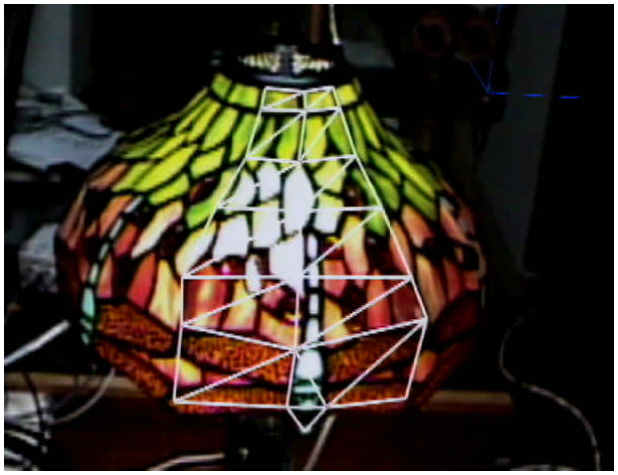

Figure 5. Wire frame model of a 60-degree segment of the lamp overlaid on the video image of the lamp.

3.3 Texture Acquisition

Taking advantage of the video-see-through HMD, we use video images to capture the textures of the model's triangles. Accurate registration between real objects and models in the AR view (which requires precise tracking of the real object as well as of the user's head) is crucial during this process.

To acquire textures for one or more visible triangles, a video image must be selected from the stream of incoming video images. In other words, a decision must be made as to when to acquire a triangle's texture. The user can either specify the image (or rather, moment in time) manually or let the system choose one using its own heuristics. Usually it is impossible for all triangles to receive their textures from a single image. Therefore, the user typically changes viewpoints and angles (i.e. typically rotates the object in front of the HMD's "eyes") until all textures of the model are acquired.

The triangle textures are also rendered transparently on top of the model in the HMD view. This allows the user to determine if each triangle already has a texture and assess the quality of each texture. The user can control the degree of transparency to see the real object through the textured model. Aside from the model superimposed onto the object, a copy of the model (with opaque textures) is also rendered. This copy is rendered next to the real object. The real object and its virtual copy move and rotate together. The copy is useful for examining how much texture data has already been acquired, and for comparing the overall appearance of the real object and the model being created (Figure 6).

Figure 6. Model with texture. The textured model is overlaid onto the real lamp on the left. A copy of the model is on the right.

3.3.1 Texture Acceptance Criteria

For a given video image, the system computes image coordinates of triangles that are possibly visible in that image. The poses of HMD cameras and triangles

influence image coordinates of triangle vertices as well as triangle visibility. As mentioned earlier, all triangle vertices are ordered counterclockwise, which allows the system to detect back-facing (invisible) triangles.

Even though a triangle is visible in the image, it is not always a good idea to capture its texture. If the normal vector of the triangle is almost orthogonal to the viewing direction, the texture is not captured.

As mentioned, the user can select automatic or manual texture capture. In the automatic capture mode, the system calculates the visibility and viewing angle of each triangle at every frame, and captures and assigns textures to selected triangles. This mode is computationally expensive. In manual mode, the system performs the computations only when the pedal is depressed. The smaller computational load is better for interactivity.

It is also possible to select a group of triangles (an *active group*) and restrict the system to capture textures for those triangles only. Manual mode and active group selection give the user better control when dealing with occlusion and inconsistent lighting conditions. These issues are discussed below.

3.3.2 Occlusion

There are two kinds of occlusions. The first kind is caused by the object to be modeled itself. If the object is concave, parts of the object may occlude other parts as seen through the HMD cameras. No textures should be acquired for (fully or partially) occluded triangles. Occlusion could be automatically detected by comparing depth values of triangles in the z-buffer. However, if the user does not first complete triangle definition before going on to texture capture, this technique does not work. We therefore adopted a heuristic method that relies on human intervention. As described in the previous section, the user can select a group of triangles and prevent others from taking any texture. It is easier to avoid this kind of occlusion when the number of triangles being modeled is smaller. By selecting only a few triangles for acquisition, concave objects can be easily managed.

The other type of the occlusion is caused by physical objects whose geometry is unknown. The user's hand is a typical example. It is inevitable to hold or touch an object while capturing the texture, because one needs to acquire several different video images to get all textures, in particular for highly concave objects. The user should therefore pay attention to the position of the hands. Other researchers have investigated this issue [7][9][20].

3.3.3 Lighting Conditions and Inconsistent Color

Adjacent triangles do not necessarily receive their textures from the same video image. For example, only a specific set of triangles may be included in the active group, or the textures for the one set of triangles are accepted by the system while adjacent triangles receive their textures at different times—when the acceptance criteria described above are met. In such cases, adjacent triangles will receive textures from two different video images. If the lighting conditions are different for the two images, the border of the two triangles will be clearly visible as shown in Figure 7.

The lighting conditions include position and brightness of the light and the orientation of the triangle. We use as many light sources as possible to make the lighting uniform within the working area. It also helps if the user uses consistent lighting for all triangles when selecting the video images.

Specular objects require more attention due to the view-dependent effects. In our system, texture is supposed to provide view-independent colors only. Hence the user should choose a viewing direction that minimizes highlights. If a triangle uses an image with a highlight, and an adjacent triangle uses an image without a highlight from a different viewing direction, the discontinuity of intensity also reveals the border between the two triangles; such cases should be avoided.

Figure 7. The effect of different lighting conditions during texture acquisition. The lower right triangle in the front is darker than most other triangles.

3.3.4 Interlacing

The video camera captures odd and even fields at different points in time. The digital video capture however uses non-interlaced frames. Therefore, when the user or the object move, the digitally captured video image exhibits staggered artifacts as shown in Figure 8. In our system, it is inevitable to move the object or the camera to acquire textures for all triangles in the model. It is also hard for the user to keep object and head absolutely still when capturing textures. Hence, using interlaced images degrades the quality of the texture.

Figure 8. Two staggered fields in an interlaced video image (enlargement).

This problem is handled by using only one video field in the texture acquisition phase. The half resolution image is doubled by duplicating all scan lines. To reduce the latency, we use the latest field available, which is either even or odd. We shift the image half a scan line up or down depending on which field is used. This shifting operation compensates for vertical translation due to the scan line duplication, and reduces the slight texture discontinuity at the border between any two triangles which receive their textures from different fields.

3.3.5 Storing textures

There are numerous techniques to store textures of arbitrarily shaped triangles [3][17]. Our system uses a very simple method: We find the bounding boxes of the triangles in the image and copy the pixels within the bounding boxes into a large texture array. The texture array is an aggregate of texture tiles with the same dimension (Figure 9). When copying a texture from the image, we make the bounding boxes one pixel larger in every direction so that during rendering, bilinear texture interpolation does not sample texels outside a texture.

3.4 Interactive User Interface

Many 3D modeling applications have complicated interfaces because they deal with 3D models in 2D displays. Using the tracked AR video see-through HMD, we overlay the model onto the real object. This allows the user to manipulate the model in 3D together with the real object to be modeled, thus making the modeling process highly intuitive.

Using direct 3D manipulation via the tracked probe, the user can select, copy, translate, and rotate a point or a group of points. The user can also select a group of triangles and move, rotate or mirror them (Figures 10 and 11). When copying a triangle to a new position, our system automatically merges two vertices if they are in close proximity to each other. If copied triangles already

Figure 9. Stored texture array. These are the textures used to render the model interactively. Each rectangular texture corresponds to the bounding box of a triangle.

Figure 10. Selecting a number of triangles (darker color) with the probe.

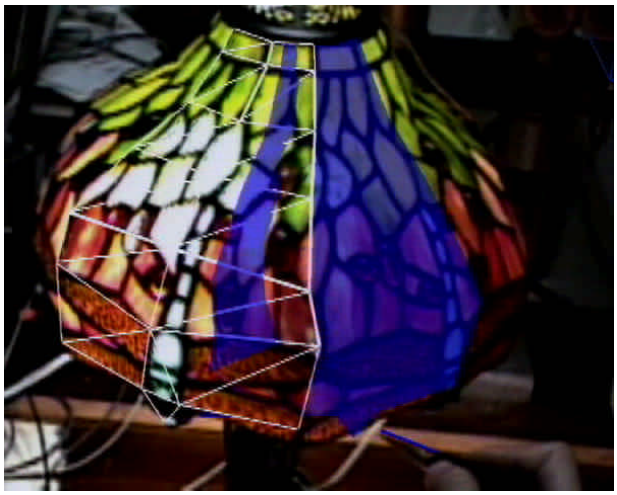

Figure 11. Copied triangles (darker color) are being rotated with the probe. Some of copies overlap with original triangles (shaded strip in the middle)

have textures, the texture coordinates are also copied to the new triangles so that they use the same texture as the original ones. The user can also reassign the textures of copied triangles using the active group method described above. Finally, the user can also deform the object by moving individual vertices (Figure 12).

Figure 12. Deforming the model by translating selected vertices.

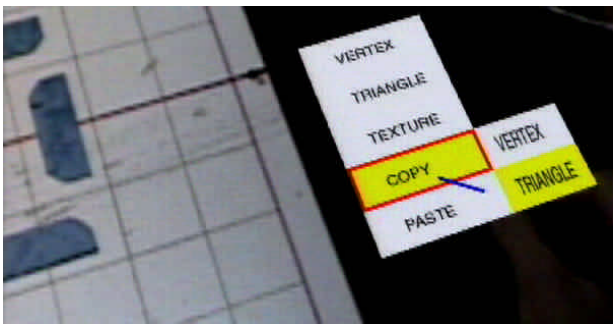

Figure 13. Selecting a menu item with the probe. (The grid on the left is used to calibrate our system.)

Figure 13 shows an interactive menu. The user can choose one of the modeling modes—sampling points, constructing a triangle mesh, acquiring textures and deforming or copying the model—by pointing at the menu with the probe and depressing the pedal. The menu is placed on the desktop working surface (table), which provides haptic feedback. It also avoids additional fatigue caused by holding the probe still in the air in menu selections. The user simply touches the table with the probe tip when picking an item from the menu.

4 Results and Discussion

Figure 14 shows a complete model of a Tiffany lamp. The lamp is illuminated by its own light bulb. Because the lamp is rotationally symmetric, we made a polygonal model of 1/6 of the lamp and acquired texture for the part. A rotation axis was defined as follows; First, we selected three points on the lamp defining a triangle. The rotation axis is perpendicular to this plane and passes through the center of the triangle's circumscribing circle. After defining the rotation axis, we copied and pasted the 1/6 part five times, each time with a 360/6 degrees rotation around the axis, to make a complete model. Since the lamp is not exactly symmetric, the texture shows a subtle seam between original and copied triangles. (There are no cracks however.)

The model contains 96 vertices and 156 triangles. It took about 5 minutes to create it.

Figure 14. Complete lamp model

Figure 15 shows a model of a shiny teapot. The handle and spout demonstrate our system's ability to deal with concave objects. Because these parts can be easily occluded or occlude other parts of the model, we worked with small parts of the object and added them to the model one at a time. Taking advantage of the symmetry of the teapot, only one side of the model was modeled manually and then mirrored to the other. The color of this teapot is sensitive to lighting conditions because it is made of shiny china. We tried to keep the lighting conditions similar for every triangle, with some (limited) success. Figure 15 shows some visible seams between triangle textures.

The teapot model consists of 111 vertices and 206 triangles. It took 30 minutes to create this model including texture, longer than for the lamp because the teapot requires special attention for the handle, the spout and variations in lighting conditions. Nevertheless, these results show that a trained user can create good photo-textured models of real objects in a short amount of time.

The frame rate was 20 frames per second during point sampling and triangle construction. During texture capture, the frame rate slows down significantly, depending on the number of triangles, especially in the automatic mode (down to 2 frames per second). In manual mode, there is only a momentary slowdown (glitch) when the capture is activated.

Figure 15. Complete teapot model.

5 Conclusions

We have presented a video-see-through augmented reality system for modeling real objects. Our system allows the designer to work in the same space as the real object, providing an intuitive and user-friendly interface.

By using a 3D probe, the user samples points of a real object and constructs triangles using these points. Textures are then extracted from video images in an interactive manner. We also provide interactive methods to manipulate the model, to create new models or to deform existing models.

Methods to handle issues such as occlusion, inconsistent lighting and interlaced video capturing are also discussed.

Acknowledgments

We thank Henry Fuchs, Kurtis Keller, David G. Harrison, Karl Hillesland and Jeremy Ackerman. Funding was provided by NIH grant P01 CA47982 and by NSF Cooperative Agreement no. ASC-8920219: "Science and Technology Center for Computer Graphics and Scientific Visualization."

References

[1] Ronald T. Azuma, "A Survey of Augmented Reality." *Presence: Teleoperators and Virtual Environments 6*, 4 (August 1997), 355 - 385.

[2] P. Besl, "Advances in Machine Vision, Chapter 1—Active optical range imaging sensors." *Springer-Verlag*, 1989, 1-63.

[3] N. Carr, J. Hart, J. Maillot, "The Solid Map: Methods for Generating a 2D Texture Map for Solid Texturing." *ACM SIGGRAPH 2000 course notes ¾ Approaches for Procedural Shading on Graphics Hardware.*

[4] G. Hausler, W. Heckel, "Light sectioning with large depth and high resolution." *Applied Optics*, Dec 1988, 5156-5159.

[5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, "Surface reconstruction from unorganized points." *Proceedings of ACM SIGGRAPH `92*, July 1992, 71-78.

[6] B. K. P. Horn, M. Brooks, "Shape from Shading." MIT Press, *Cambridge, MA*, 1989.

[7] M. Inami, N. Kawakami, D. Sekiguchi, Y. Yanagida, "Video-Haptic Display Using Head-Mounted Projector." *Proceedings of Virtual Reality 2000 Conference*, March 2000, 233-240.

[8] R. A. Jarvis, "A perspective on range-finding techniques for computer vision." *IEEE Trans. Pattern Analysis Mach. Intell*, March 1983. 122-139.

[9] M. Kanbara, T. Okuma, H. Takemura, N. Yokoya, "A Stereoscopic Video See-through Augmented Reality System Based on Real-time Vision-Based Registration." *Proceedings of Virtual Reality 2000 Conference*, March 2000, 255-262.

[10] K. Keller. http://www.cs.unc.edu/~keller/lapro/Augmented_Reality_HMDs/gallery.html, University of North Carolina at Chapel Hill

[11] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. koller, L. Pereira, M. Ginzton, S. Anderson , J. Davis, J. Ginsberg, J. Shade, D. Fulk, "The Digital Michelangelo Project: 3D Scanning of Large Statues." *Proceedings of ACM SIGGRAPH 2000.* (July 2000), 131-144.

[12] D. K. McAllister, L. Nyland, V. Popescu, A. Lastra, C. McCue, "Real-Time Rendering of Real World Environments." *Proceedings of Eurographics Workshop on Rendering*, 1999.

[13] MetaCreations CANOMA, http://www.metacreations.com/products/, MetaCreations Corp.

[14] J. Oliensis, M. Werman, "Structure from Motion using Point, Lines, and Intensities." *CVPR 2000*, 599-606 and J.Oliensis, "Direct Multi-Frame Structure from Motion for Hand-Held Cameras." *ICPR 2000*.

[15] PhotoModeler, http://www.photomodeler.com/, Eos Systems Inc.

[16] Kevin H. Martin, "The Sound and the Fury" Cinefex 74, *Valley Printers, INC. July 1998, 82-106.*

[17] D. Stalling, "LIC on Surfaces", *ACM SIGGRAPH97 course notes ¾ Texture Synthesis with Line Integral Convolution*, 51-62.

[18] A. State. http://www.cs.unc.edu/~us/web/headmounts.htm, University of North Carolina at Chapel Hill

[19] A. State, G. Hirota, D. T. Chen, W. F. Garrett, M. A. Livingston. "Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking." *Proceedings of ACM SIGGRAPH 96*, (August 1996). 429-438

[20] N. Yokoya, H. Takemura, T. Okuma, M. Kanbara, "Stereo Vision Based Video See-through Mixed Reality." *Mixed Reality – Merging Real and Virtual Worlds, Ohmasha, Ltd. & Springer-Verlag*, 131-145.

[21] Z. Zhang, "Modeling geometric structure and illumination variation of a scene from real images." *Proceedings of 6th Int'l Conference On Computer Vision*, 1998, 1041-1046