Joohi Lee
lee@cs.unc.edu

Gentaro Hirota
hirota@cs.unc.edu

Andrei State
andrei@cs.unc.edu
University of North Carolina
at Chapel Hill
Chapel Hill, NC, 27599, USA

Modeling Real Objects Using Video See-Through Augmented Reality

Abstract

This paper presents an interactive "what-you-see-is-what-you-get" (WYSIWYG) method for creating textured 3-D models of real objects using video see-through augmented reality. We use a tracked probe to sample the objects' geometries, and we acquire video images from the head-mounted cameras to capture textures. Our system provides visual feedback during modeling 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 (and sometimes to paint them). The clay models are then digitized as computer models so that they can be animated. In mixed-reality systems, models of real-world objects are often required to render real and virtual objects with correct occlusion relationships. Thus, creating digital copies of real objects is an important technology, often referred to as *3-D digitizing*.

1.1 Related Work

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

1.1.1. 3-D Reconstruction from 2-D Images. These methods exploit computer vision techniques such as structure from motion, stereo triangulation, and structure from shading (Zhang, 1998). They try to mimic human vision by reconstructing 3-D information from 2-D images. Typical strategies such as structure from motion recover 3-D geometry of objects (and sometimes also camera motion) from images taken from different viewpoints. Detecting and matching common features among multiple images is crucial for such methods (Oliensis & Werman, 2000; Oliensis, 2000). (Some simple commercial products require manual assistance (www.canoma.com, www.photomodeler.com). There are no specific requirements for the light sources that illuminate the objects, but specularity and transparency of the object sur-

Presence, *Vol. 11, No. 2, April 2002,* 144–157

faces pose significant problems for feature detection and matching. Highly concave objects are also difficult to handle. For example, if an object has a deep hole, surface points inside the hole can be seen from only a very narrow angle (that is, not from all camera viewpoints), making it difficult to recover accurate coordinates for those points.

1.1.2. Structured Light Projection. Instead of using natural light sources, one can project light patterns onto the real-world objects (Besl, 1989; Hausler & Heckel, 1988; Horn & Brooks, 1989; Jarvis, 1983). In camera images, such patterns can be easier to detect than natural features. In most cases, the position of projectors and cameras are known (or determined by calibration techniques), and points on object surfaces can be determined by triangulation. Widely used commercial systems (Levoy et al., 2000) 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.

Because this method relies on the reflection of light, it cannot easily digitize specular or transparent objects. Furthermore, the concavities are even more problematic for this class of methods. A successfully sampled point must be visible (unoccluded) not only from the cameras but also from the light source (projectors).

1.1.3. 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 3-D tracking system. (See Martin (1998) for a description of how this is typically done for motion picture visual effects.) 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, 3-D reconstruction from 2-D images and structured light projection typically use a reconstruction algorithm (Hoppe, DeRose, Duchamp, McDonald, & Stuetzle, 1992) to build surface models. In the point sampling method, points are sampled manually; hence, typically fewer points are collected than in the other methods. In this case, surface models may be constructed by manually connecting the sample points or by using automatic methods.

Finally, Leibe et al. (2000) describe an interesting AR system that uses techniques from the first method (3-D reconstruction from 2-D images) together with computer-controlled infrared (IR) illumination to create coarse models of real-world objects from IR shadow contours. The system focuses on quick, automatic, approximate reconstruction of models that are suitable for certain forms of interaction in AR environments. Due to the significant errors inherent in the reconstruction method, such models are not (yet) adequate for the applications described in our introduction.

1.2 Motivation and Contribution

The methods involving 3-D reconstruction and structured light projection work well for objects with relatively simple geometry and diffuse reflective and opaque surfaces, and the second method 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 both fail to sample points on highly concave areas.

The third method (point sampling) is immune to specularity and transparency problems (as far as geometry acquisition is concerned) because it does not rely on image sensors. Furthermore, it is suitable for sampling concave parts because 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 exact location of 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. Also, such a separate display shows the locations of the sampled points in relation only to each other and not to the ob-

ject being modeled: 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. Furthermore, colors or textures cannot be acquired in this way.

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 (and the camera imagery could even be used to acquire textures). 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 a modeling system that is based on augmented reality (AR) and 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 (such as to scale, or deform), 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 3-D 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 that is 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 computes visibility parameters for each polygon of the model. These parameters are used to prevent acquisition of video textures that are either distorted or too small.

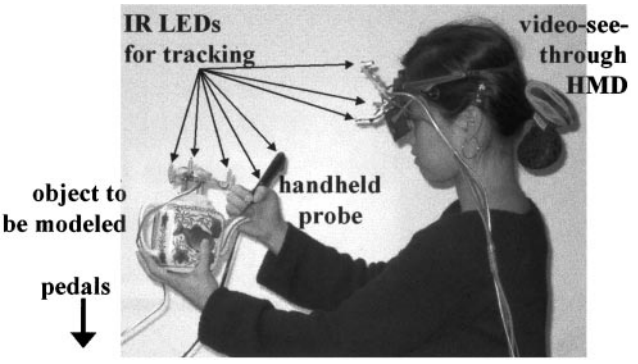

Figure 1. *User with AR HMD, modeling a teapot with the handheld probe. Probe, teapot, and HMD are equipped with IR LEDs for opto-electronic tracking.*

2 System Overview

Our current AR modeling system (an earlier version was described by State, Hirota, Chen, Garrett, and Livingston (1996)) integrates off-the-shelf components: a high-resolution HMD, miniature video cameras, an optical tracking system, a 3-D probe, foot pedals, a high-end graphics workstation, and several rigs equipped with IR LEDs for tracking.

The following list provides descriptions of the hardware configuration of our system.

- *Video see-through HMD:* We mounted two miniature Toshiba IK-SM43H cameras on a stereoscopic Sony Glasstron LD1-D100B head-mounted display with SVGA (800×600) resolution (figure 1). The two cameras are used to provide a stereoscopic view of the real world. Three IR LEDs are also mounted on the HMD for optical tracking. (See State, Ackerman, Hirota, Lee, and Fuchs (2001) for a detailed description of this device and its operation.)
- *Probe:* The probe has two IR LEDs to track the position of the probe tip (figure 1).

- *Object trackers:* Three IR LEDs are rigidly affixed to each of the objects to be modeled (such as the porcelain teapot in figure 1).
- *Optical sensors:* All IR LEDs are tracked by an Image Guided Technologies FlashPoint 5000 3-D Optical Localizer. Its sensor assembly holds three 1-D CCD arrays that triangulate the position of the LEDs.
- *Pedals:* We use foot pedal switches to assist with interactive operations.
- *Light:* Light is provided by various ambient 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 multichannel graphics pipe and two R12000 CPUs.

Figure 2. *Sampling the geometry of a Tiffany lamp shade with the probe. The dark dots represent the acquired points. A dark line segment is rendered over the tip of the acquisition probe.*

3 System Operation, Implementation Details, and Critical Issues

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 an HMD, as shown in figure 1.

We first describe how points are sampled and how triangles are created. 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 freely 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.

3.1 Point Sampling

The first step when modeling with our system is to sample the object geometry, which is quite simple. The user defines points on the object by physically touching the surface of the real object with the (optically) tracked probe and depressing a pedal (figure 2). When a point is sampled, the point in probe coordinates is transformed into the object's coordinate system, using the current transformations from the LED trackers of probe and object. Because the object is continually tracked just like the handheld probe, the newly acquired point can be rendered on top of the object's video image and appears to the HMD user as "attached" to the object's surface.

Overlaying points on the object helps the user to understand which points have been added, to determine which part of the object needs more sampling, and to sample the exact point desired. Some confusion may occur if two points happen to be rendered close to each other on the screen and if 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 have only a collection of points and no polygons that could be used to occlude the back points via *z*-buffering.) However, because points are displayed in stereo as is everything else, the user can still see which one is in front. Nevertheless, it can be visually confusing if the points on the back face are not occluded properly by the object. (This occlusion-depth conflict could be alleviated by using depth cueing (for example, by varying color) when displaying the collected points.) Our system allows the user to build a model incrementally, which reduces the possible confusion. Instead of defining triangles after all points of the object have been collected, the user can

Figure 3. *Modeling the concave part (handle) of a teapot.*

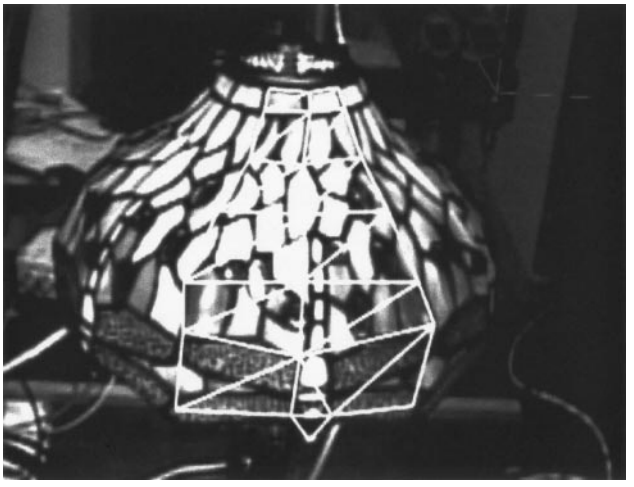

Figure 4. *Wireframe model of a 60 deg. segment of the lamp overlaid on the video image of the lamp.*

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 rasterizes triangles into the *z*-buffer, points behind the available, already defined triangles can be occluded. Of course, 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 3 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 concave parts of the object's surface with the probe. However, in our specific implementation, the user must be aware of the limitations of the optical tracking system, and not turn the object's or the probe's LEDs away from the optical sensors. The user also needs to be cautious not to occlude the LEDs from the optical sensors with body parts such as hands or fingers. Although we cannot eliminate this problem (which is typical for all opto-electronic trackers), we alleviate it through visual and audio feedback. The system does not accept a sample point if either the object or the probe loses tracking; it then warns the user of the tracking problem, thus preventing him or her from acquiring erroneous points.

3.2 Triangle Formation

After collecting points, the user is ready to create triangles by connecting the sampled points. In "triangle definition" mode, 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. If necessary, 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 subsection 3.3) and for backface culling (if desired) 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. (See figure 4.) This interactive feedback gives a good idea about how the model is being formed.

It is also possible to apply an automatic reconstruction algorithm (Hoppe et al., 1992) rather than manually constructing the triangle mesh. The user can invoke such an algorithm offline and then edit it interactively and/or continue on to the next step (texture acquisition).

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 each triangle's texture. The user can either specify the image (or, rather, the moment in time) manually or let the system decide using its own heuristics (discussed later). It usually is impossible for all triangles to receive their textures from a single image in the HMD camera video stream. Therefore, the user typically changes viewpoints and angles (typically by rotating 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 whether each triangle already has a texture and to assess the quality of each texture. The user can control the degree of transparency of the textures to see the real object through the textured model. Along with the textured variable-transparency model superimposed onto the object, a copy of the model with fully opaque textures is also rendered. This copy is displayed floating in space next to the real object. The real object (with or without the superimposed model) and the nearby copy of the model 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. (See figure 5.)

Figure 5. *Model with texture. The textured model is overlaid onto the real lamp on the left. A copy of the model is visible 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, the system rearranges triangle vertices in counterclockwise order.

Even though a triangle is visible in the image, it is not always a good idea to capture its texture. For instance, if the normal vector of the triangle is almost orthogonal to the viewing direction, the texture should not be 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 triangles. This mode is computationally expensive because screen-space data for all triangles must be processed. (We are of course taking advantage of the graphics pipeline feedback available on the IR system.) In manual mode, the system performs these computations only when the pedal is depressed; the smaller computational load in manual mode is of course 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. Together, manual mode and active group selection give the user better control when dealing with occlusion and inconsistent lighting conditions. These issues are discussed in detail in the next two subsections.

3.3.2 Occlusion. There are two kinds of occlusions. The first is caused by the very object that the user

is modeling. If this object is concave, parts of it may occlude other parts as seen through the HMD cameras. No textures should be acquired for triangles that are either fully or partially occluded. 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 capture textures, this technique does not work. We therefore adopted a heuristic method that relies on human intervention (or rather, operator awareness). As described in the previous subsection, the user can select a group of triangles, thus preventing all other triangles from taking any texture. It is easier to avoid this kind of occlusion if the selection group is small. Thus, by earmarking only a few triangles at a time for texture acquisition, concave objects can be easily managed.

The second type of occlusion is caused by the presence of 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 also pay attention to the position of his or her hands. Other researchers have investigated this issue (Inami, Kawakami, Sekiguchi, & Yanagida, 2000; Kanbara, Okuma, Takemura, & Yokoya, 2000; Yokoya, Takemura, Okuma, & Kanbara, 1999).

3.3.3 Lighting Conditions and Inconsistent Color. Adjacent triangles do not necessarily receive their textures at the same time from a 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 may be accepted by the system whereas adjacent triangles may receive their textures at different times—when the acceptance criteria previously described 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 between the adjacent triangles will be clearly visible, as shown in figure 6.

The lighting conditions include the position and brightness of the light and the orientation of the triangle. After failed experiments with an HMD-mounted light source (too much variation depending on the distance between HMD and the object being modeled), we switched to several light sources positioned around the working area to make the lighting as uniform as possible. It also helps if the user looks for consistent lighting for all (active) triangles when selecting the video images to be used for texturing in the manual texture capture mode.

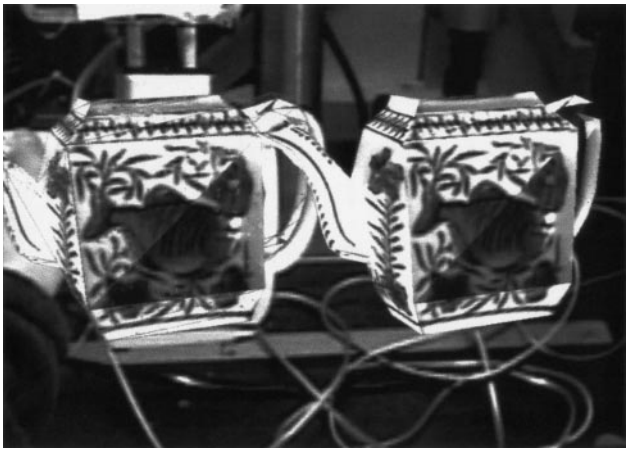

Figure 6. *The effect of different lighting conditions during texture acquisition. The lower right triangle in the front is darker than most other triangles.*

Specular objects require more attention due to the view-dependent effects. In our prototype, texture provides only view-independent colors. 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 in brightness also reveals the border between the two triangles. Such cases should be avoided.

3.3.4 Video Interlacing. The video camera captures odd and even fields at different moments in time. However, the digital video capture uses noninterlaced frames. Therefore, when the user or the modeled object move, the digitally captured video image exhibits staggered artifacts as shown in figure 7. In our system, it is inevitable to move the object or the camera to acquire

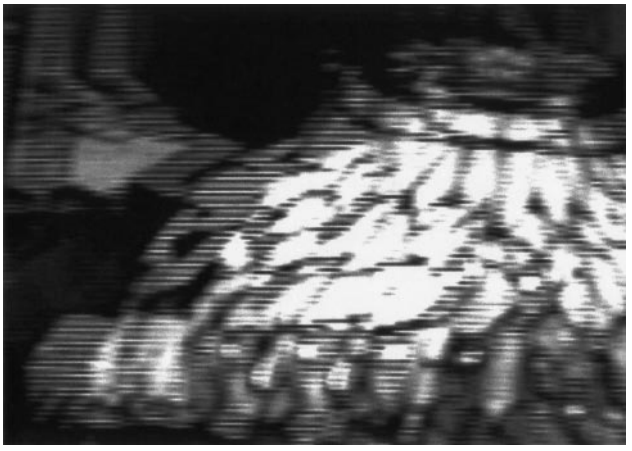

Figure 7. *Two staggered fields in an interlaced video image (enlargement).*

Figure 8. *Stored texture array. These are some of the textures that are used to render the teapot model interactively. Each rectangular texture corresponds to the bounding box of a triangle.*

textures for all triangles in the model. It is also hard for the user to keep both object and head absolutely still when capturing textures. Beyond that, tracker jitter—which is present to some extent in all commercial and experimental trackers we know of (except maybe the very best mechanically tethered ones)—also plays a role. Hence, using interlaced images degrades the quality of the acquired textures.

This problem can be addressed by using only one video field in the texture acquisition phase. The half-resolution image is vertically doubled in size by duplicating all scan lines. To keep latency at a minimum, 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 that receive their textures from different fields.

3.3.5 Storing Textures. Numerous techniques can be used to store textures of arbitrarily shaped triangles (Carr, Hart, & Maillot, 2000; Stalling, 1997). 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. See figure 8.) When copying a texture from the image, we enlarge their bounding boxes by one pixel in every direction so that, during rendering, bilinear texture interpolation will not sample texels outside a texture. (A texel is a pixel in a texture image.)

3.4 Interactive User Interface

Many 3-D modeling applications have complicated interfaces because they deal with 3-D models in 2-D 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 3-D together with the real object to be modeled, thus making the modeling process highly intuitive and greatly simplifying the user interface.

Using direct 3-D manipulation (via the tracked probe and the two foot pedals), the user can select, copy (to a clipboard), paste (from the clipboard), translate, and rotate a point or a group of points. These operations can also be performed on triangles. To define the cur-

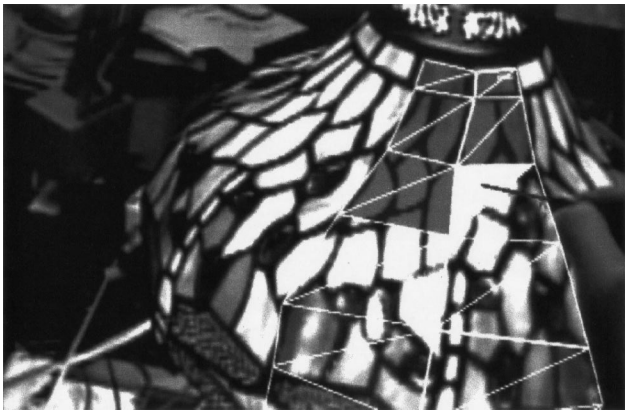

Figure 9. *Selecting a number of triangles (darker color) with the probe.*

Figure 10. *Copied and pasted triangles for a segment of the lamp shade (darker color) are being rotated with the probe. Some of the copies overlap and intersect the original triangles (shaded strip in the middle).*

rent axis for the rotation operator, the user first selects three points defining a triangle. The rotation axis is then calculated to be perpendicular to the plane of the triangle and to pass through the center of the triangle's circumscribing circle. Figures 9 and 10 show two HMD view snapshots during a select paste-rotate operation on triangles.

When moving triangles to a new position, our system automatically merges two vertices if they are in close proximity to each other. If the copied triangles already have textures, the texture ID as well as the texture coordinates are also copied to the new triangles so that they use the same texture as the original ones. The user can of course reassign the textures of copied triangles (using the active group method previously described). Finally, the user can also deform the object by selecting and moving individual vertices. (See figure 11.)

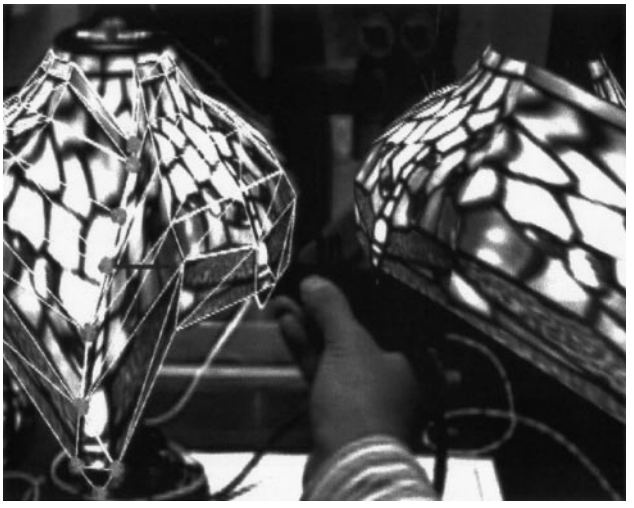

Figure 11. *Deforming the model by translating selected vertices (marked by gray blobs).*

Figure 12 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 free "static" haptic feedback. (It also avoids hand/arm fatigue caused by holding the probe still in the air in menu selections, as is the case with floating toolboxes.) The user simply touches the table with the probe tip when picking an item from the menu.

4 Accuracy of Acquired Models

Two separate issues must be considered with respect to the accuracy of the models created with such a

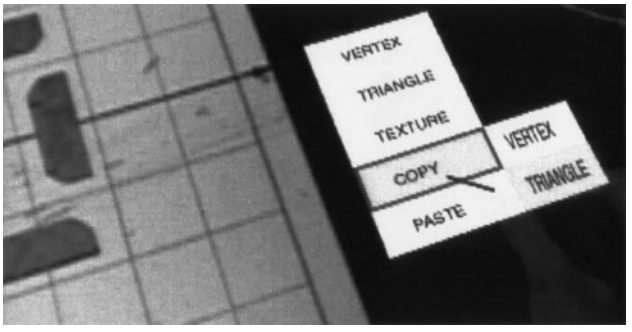

Figure 12. *Selecting a menu item with the probe. (The grid on the left is used to calibrate our system.)*

system. The first of them is the accuracy of the point coordinates; it depends on the accuracy of the tracking system. The FlashPoint 5000 specifications mention an average error for LED position readings of approximately 0.5 mm over the entire tracking volume. Considering the construction of the handheld probe (the measurement tip and two LEDs are collinear, mounted approximately 100 mm apart), the average error at the collected points is of course higher (at least 1 mm, which is twice the individual LED value). This error directly affects the accuracy of collected points. In addition, the error in tracking the object being modeled also enters the equation, given that the two transformations (object and probe) must be concatenated to obtain object coordinates. (If one were to implement such a system with separate trackers for object and probe, the calibration transformation between the two trackers would be a source of additional errors.) The object tracker is a three-LED, six-degree-of-freedom tracker. Hence, it has not only position error (order of magnitude 0.5 mm) but also orientation error, thus reducing the accuracy of points sampled far away from the object tracker LED locations. The actual angular error is obviously dependent on the size of the LED triangle used to track each object. This is the reason we mount larger LED triangles on the larger objects. The LED triangle for the teapot in figure 1 is approximately 100 mm on the side, whereas the Tiffany lamp in figure 4 has a 240 mm LED triangle. In both cases, the LED triangles are roughly as large as the objects, keeping the orientation component of the error at the same order of magnitude as the position error (approximately 1 mm). In conclusion, we can assume average total point sampling errors of about 3 mm. These could be reduced by using more accurate trackers (for example, tethered mechanical arms) or by modeling fixed objects instead of tracked ones. Also, as it turns out, our FlashPoint tracker has better accuracy in the central area of its tracking volume (down to 0.23 mm), where we constructed most of our models.

The second accuracy issue relates to the captured textures. If point position error depends on the combined error in object and probe trackers, texture alignment error depends on the combined errors of these two trackers and of the HMD tracker (because the textures are acquired via HMD-mounted cameras). In addition, HMD camera calibration errors can also lead to texture alignment problems, which are potentially visible as seams. (Details regarding our HMD hardware and software techniques for HMD calibration are given in State et al. (2001).) Texture resolution is a direct function of camera resolution (low-end NTSC only, given the state of the art for miniaturized devices suitable for HMD mounting). Both resolution errors and additional distance-dependent texture alignment errors (resulting from angular errors in the HMD tracker) can be alleviated by holding the object being modeled close to the HMD at texture capture time. Another possibility (which we have not investigated) would be to use a rigidly mounted and very accurately calibrated high-resolution camera (it can be bulky and heavy as opposed to the HMD cameras) for texture acquisition. Its frustum and depth of field could be displayed as wireframe elements to the AR user, who would have to hold the model inside it (and not in front of the HMD) to acquire textures. This would have the added benefit of easily controlled uniform lighting within the camera's acquisition area. Finally, the specific problems resulting from video interlacing, another camera-related error, were previously discussed.

Both point position and texture alignment accuracy can be severely affected by the relative lag between tracking devices if different kinds of trackers are used for the three tracked components: HMD, object, and probe. This is not the case in our implementation, how-

Figure 13. *Complete Tiffany lamp shade model.*

Figure 14. *Complete teapot model.*

ever. Also, relative lag between HMD camera image capture and tracker readings will affect texture alignment. In recent work not yet submitted for publication, the relative lag between these two input streams to our system was measured to be approximately 20–30 msec, a fairly low value that can nevertheless lead to visible errors if the user moves the object or the HMD quickly during texture acquisition.

5 Results

Figure 13 shows a complete model of the Tiffany lamp shade. The lamp is illuminated by its own light bulb. Because the lamp is rotationally symmetric, we made a polygonal model of a 1/6 segment of the lamp and acquired texture for that part. After defining the rotation axis to coincide with the axis of the lamp, we copied and pasted the segment five times (see figure 10), each time with a 360°/6 rotation, to make a complete model. Because the lamp is not exactly symmetric, the texture shows a subtle seam between the original and the copied triangles. (There are no cracks in the geometry, however.) This model contains 96 vertices and 156 triangles and was created in approximately 5 min.

Figure 14 shows a model of a shiny teapot. The handle and the spout demonstrate our system's ability to deal with concave objects. Because these parts can be easily occluded or can themselves 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 create the other. The color of the (physical) 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 14 shows some visible seams between triangle textures.

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

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

Figure 15 shows the (untextured) model of a hand-

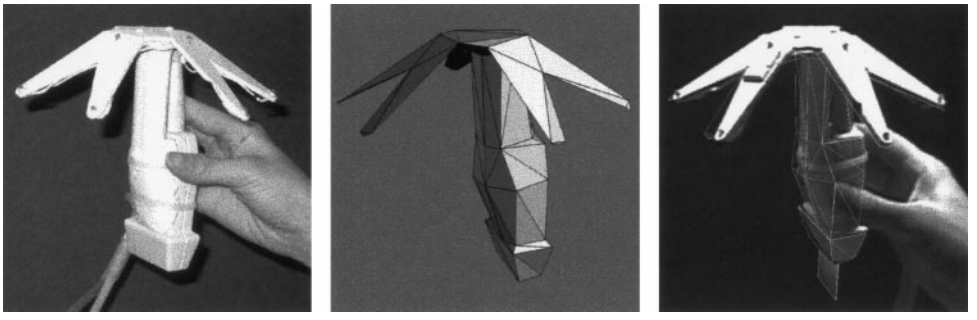

Figure 15. *Left: tracked, handheld ultrasound probe used in medical AR system. Center: probe model created in about one hour with the AR modeling system (65 vertices, 105 triangles). Right: probe with registered model, as seen in the HMD by a user of the medical AR system.*

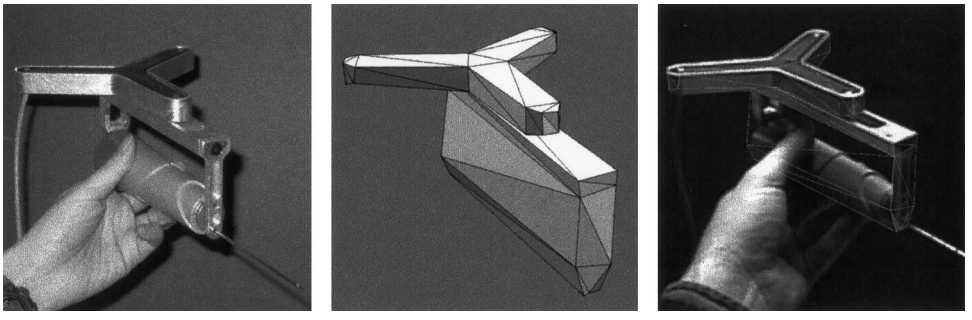

Figure 16. *Left: tracked biopsy needle used in medical AR system. Center: model (one hour, 48 vertices, 88 triangles). Right: needle with registered model, as seen in the HMD by a user of the medical AR system.*

held ultrasound transducer (used for medical ultrasound scans), also equipped with infrared LEDs. This model is used in an experimental AR system for ultrasound-guided needle biopsy (Rosenthal et al., 2001), as is the biopsy needle (also attached to an LED tracker) shown in figure 16. Figure 17 shows a still frame from the video see-through HMD (the same one used for this work) of the medical AR system; the two objects (transducer and needle) are visible in the foreground. Behind them is a virtual ultrasound monitor. The occlusion relationship between the two objects and virtual imagery (such as the ultrasound monitor) can only be established if the video images of the tracked, handheld transducer and needle also have appropriate depth values in the z-buffer. In our medical AR system, this is accomplished via z-buffer-only rasterization of the models created with the AR modeler described here.

6 Conclusions and Future Work

We have presented an interactive, video see-through, augmented-reality system for modeling real objects in a WYSIWYG manner. 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 3-D probe, the user samples points on the surface of a real object and constructs triangles using these points. Textures are then extracted from video images in an interactive manner. We also provide inter-

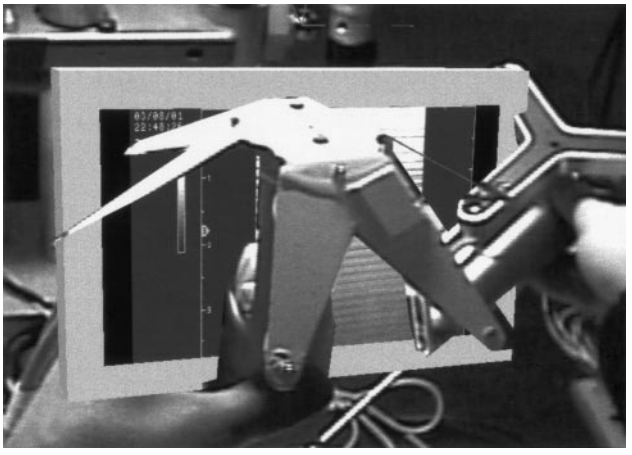

Figure 17. *Tracked ultrasound probe and needle as seen by medical AR system user, correctly occluding virtual ultrasound monitor (background). The user's hands are incorrectly occluded by the monitor because the system has no model for them (or knowledge of their position).*

active methods to manipulate the model, to create new models, or to deform existing ones. Methods to handle potential problems such as occlusion, inconsistent lighting, and interlaced video capture were also discussed.

Possible future enhancements include the interactive editing of surface normals, use of progressive scan cameras to avoid resolution loss from interlaced video, better texture storage management, and texture postprocessing algorithms to even out lighting and color inconsistencies, to improve alignment (for example, through texture coordinate error distribution), and to improve texture quality (the use of super-resolution techniques may be possible with jitter-free trackers).

The models we create are only as good as the tracking system used; their texture quality depends heavily on HMD camera resolution and calibration accuracy. The latter is directly affected by tracking accuracy. Thus, most limitations of our specific implementation stem from today's low-accuracy 3-D tracking technology, which remains the principal inhibiting factor for the widespread use of AR, including systems such as the one described here. The remaining problems should be overcome by future low-lag video capture and rendering engines and high-resolution, noninterlaced digital video cameras.

Finally, combining our technique—which is robust but only moderately accurate (depending on the tracker)—with shape from shading and stereo correlation algorithms—which are accurate but not particularly robust)—may lead to a vastly superior hybrid technique that could be both robust and accurate.

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

Besl, P. (1989). Active optical range imaging sensors. In J. Sanz (Ed.), *Advances in machine vision: Architectures and applications* (pp. 1–63). Berlin: Springer-Verlag.

Carr, N., Hart, J., & Maillot, J. (2000). The solid map: Methods for generating a 2D texture map for solid texturing. *ACM SIGGRAPH 2000 Course Notes—Approaches for Procedural Shading on Graphics Hardware*.

Hausler, G., & Heckel, W. (1988, December). Light sectioning with large depth and high resolution. *Applied Optics,* 5156–5159.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., & Stuetzle, W. (1992). Surface reconstruction from unorganized points. *Proceedings of ACM SIGGRAPH 92,* 71–78.

Horn, B. K. P., & Brooks, M. (1989). *Shape from shading.* Cambridge, MA: The MIT Press.

Inami, M., Kawakami, N., Sekiguchi, D., & Yanagida, Y. (2000). Video-haptic display using head-mounted projector. *Proceedings of Virtual Reality 2000,* 233–240.

Jarvis, R. A. (1983). A perspective on range-finding techniques for computer vision. *IEEE Trans. Pattern Analysis,* 122–139.

Kanbara, M., Okuma, T., Takemura, H., & Yokoya, N. (2000). A stereoscopic video see-through augmented reality system based on real-time vision-based registration. *Proceedings of Virtual Reality 2000,* 255–262.

Leibe, B., Starner, T., Ribarsky, W., Wartell, Z., Krum, D., Weeks, J., Singletary, B., & Hodges, L. (2000). Toward

spontaneous interaction with the perceptive workbench. *IEEE Computer Graphics and Applications, 20*(6), 54–65.

Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., & Fulk, D. (2000). The digital Michelangelo project: 3D scanning of large statues. *Proceedings of ACM SIGGRAPH 2000,* 131–144.

Martin, K. (1998). The sound and the fury. *Cinefex, 74,* 82–106.

Oliensis, J. (2000). Direct multi-frame structure from motion for hand-held cameras. *ICPR 2000,* 889–895.

Oliensis, J., & Werman, M. (2000). Structure from motion using points, lines, and intensities. *CVPR 2000,* 599–606.

Rosenthal, M., State, A., Lee, J., Hirota, G., Ackerman, J., Keller, K., Pisano, E. D., Jiroutek, M., Muller, K., & Fuchs, H. (2001). Augmented reality guidance for needle biopsies: A randomized, controlled trial in phantoms. *Proceedings of Medical Image Computing and Computer-Assisted Intervention—MICCAI 2001,* 240–248.

Stalling, D. (1997). LIC on surfaces. *ACM SIGGRAPH97 Course Notes—Texture Synthesis with Line Integral Convolution,* 51–62.

State, A., Ackerman, J., Hirota, G., Lee, J., & Fuchs, H. (2001). Dynamic virtual convergence for video see-through head-mounted displays: Maintaining maximum stereo overlap throughout a close-range work space. *Proceedings of the International Symposium on Augmented Reality (ISAR) 2001,* 137–146.

State, A., Hirota, G., Chen, D. T., Garrett, W. F., & Livingston, M. A. (1996). Superior augmented reality registration by integrating landmark tracking and magnetic tracking. *Proceedings of ACM SIGGRAPH 96,* 429–438.

Yokoya, N., Takemura, H., Okuma, T., & Kanbara, M. (1999). Stereo vision based video see-through mixed reality. *The First International Symposium on Mixed Reality (ISMR '99),* 131–145.

Zhang, Z. (1998). Modeling geometric structure and illumination variation of a scene from real images. *Proceedings of 6th Int'l Conference On Computer Vision,* 1041–1046.