Interactive Occlusion and Collision of Real and Virtual Objects in Augmented Reality

David E. Breen
Eric Rose
Ross T. Whitaker

ECRC-95-02

Technical report ECRC-95-02

Interactive Occlusion and Collision of Real and Virtual Objects in Augmented Reality

David E. Breen
Eric Rose
Ross T. Whitaker

European Computer-Industry Research Centre GmbH (Forschungszentrum)
Arabellastrasse 17
D-81925 Munich
Germany
Tel. +49 89 9 26 99-0
Fax. +49 89 9 26 99-170
Tlx. 52 69 10

For more information please contact : David E. Breen
david@ecrc.de

Abstract

We present several techniques for interactively performing occlusion and collision detection between static real objects and dynamic virtual objects in augmented reality. Computer vision algorithms are used to acquire data that model aspects of the real world. Either geometric models may be registered to real objects, or a depth map of the real scene may be extracted with computer vision algorithms. The computer vision-derived data are mapped into algorithms that exploit the power of graphics workstations, in order to interactively produce new effects in augmented reality. By combining live video from a calibrated camera with real-time renderings of the real-world data from graphics hardware, dynamic virtual objects occlude and are occluded by static real objects. As a virtual object is interactively manipulated collisions with real objects are detected, and the motion of the virtual object is constrained. Simulated gravity may then be produced by automatically moving the virtual object in the direction of a gravity vector until it encounters a collision with a real object.

1 Introduction

Augmented reality (AR) is a combination of technologies distinct from virtual reality (VR), that promises to support a wider range of applications. Interest in AR has substantially increased in the past few years, with research groups exploring diagnostic, manufacturing, medical and repair applications. In augmented reality, the computer provides additional visual information that enhances or augments a user's view of the real world. Instead of replacing the world with a completely virtual environment, as in VR, AR brings the computer out of the desktop environment and incorporates the computer into the reality of the user. The user can then interact with the real world in a natural way, with the computer providing graphical information and assistance.

In order for AR to become fully accepted the real and virtual objects[1] within the user's environment must be seemlessly merged. For the new reality to be convincing, real and virtual objects must interact realistically. Objects in AR may interact with each other in a variety of ways, which may be placed into two categories, visual and physical. Visual interactions between real and virtual objects are based on the inter-reflections, absorption, and redirection of light emitted from and incident on these objects. Effects that we see in reality and therefore expect in AR include shadows, occlusion, diffuse, specular, and internal reflections, refraction, and color bleeding.

Physical interactions between objects include kinematic constraints, collision detection and response, and full physically-based responses to external forces. Kinematic interactions involve one object's motion constraining or directly affecting the position and orientation of another object at some connection point or joint. For collision detection, calculations are performed to determine when one object strikes another, thus preventing them from occupying the same space. The most complex interactions, ones that are physically-based, involve the exchange of forces and momentum between real and virtual objects, producing virtual objects with realistic behavior. It is important to realize that most interactions are currently one-way, i.e., real objects can affect the virtual objects, but the virtual objects cannot usually affect the real ones. This would require computer-controlled visual and physical manipulators, which are currently beyond the scope of our available technology. Figure 1.1 provides a high-level diagram of our augmented reality environment, with solid lines representing the flow of interactions and data, and the dashed line representing the potential manipulation of the real environment.

The User Interaction and Visualization (UI&V) group at ECRC has begun to explore and develop the algorithms needed to produce real-time interactions between real and virtual objects within a larger project to develop a general-purpose augmented vision[2] (AV) capability. Our initial work in this

[1]The term *virtual objects* refers to geometric models and their associated rendered forms.

[2]We use the term *augmented vision* in order to highlight our interest in combining computer

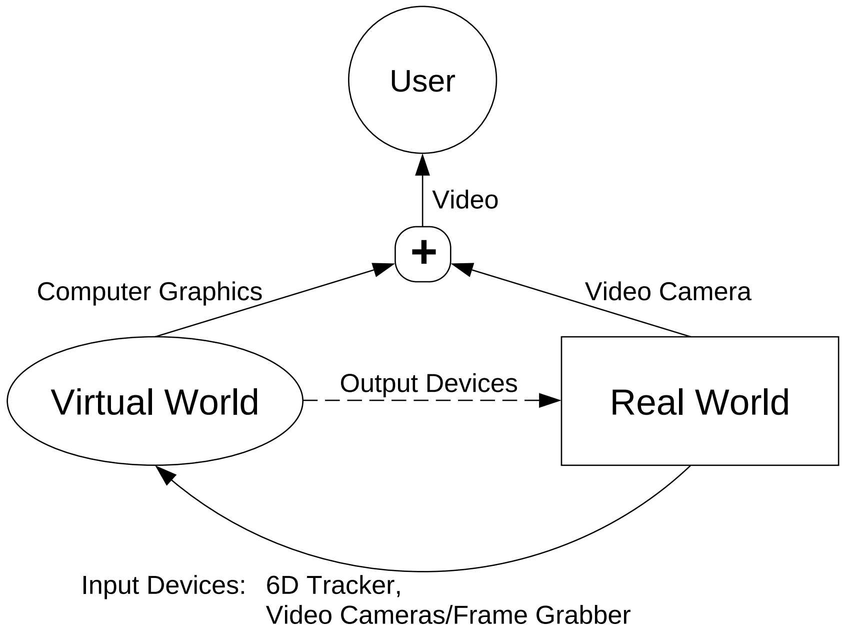

Figure 1.1: Augmented reality configuration.

area has focused on a single visual interaction, occlusion, and a single physical interaction, collision detection. Occlusion occurs when an object closer to the viewer obscures the view of objects further away along the line-of-sight. Collision detection and response prevent a virtual object from passing through a real one. Our system can also automatically move a virtual object until a collision is detected, allowing it to simulate virtual "gravity".

Computer vision algorithms for acquiring information about the real world form the foundation of our occlusion and collision detection research. We are currently exploring several strategies for bringing the real world and the virtual world into a single computational framework. Once the computer can correctly model some aspect of the real world, virtual objects can interact realistically with it. Our strategies differ in their assumptions about the real world, and produce different kinds of representations of it. If geometric models of real objects exist, they may be registered to their corresponding objects in the real environment. Otherwise, if no specific geometric knowledge about the real environment is available, a depth map of the scene can be produced.

Our algorithms for occlusion and collision detection between real and virtual objects have been strongly influenced by the type of data that may be derived from computer vision (CV) techniques. The goal of our work is to develop computer graphics algorithms that utilize the information that is available from computer vision in order to produce new capabilities for augmented reality. The challenge here is to use CV-derived data in a way that exploits the power of graphics workstations. Camera and tracker calibration methods provide us with the ability to sense and measure the real world. Once object models are

vision and computer graphics techniques for augmented reality.

registered to objects in the real environment, the models may be passed directly to the graphics system to produce occlusions. A depth map of the real scene has several applications. It may either be used directly by the graphics system to produce occlusions, or used for collision detection between virtual objects and the real environment.

2 Previous Work

Several research groups are currently exploring augmented reality for a variety of applications. Feiner et al. [12] have developed a knowledge-based AR system for maintenance and repair instruction. Lorensen et al. [19] have focused more on AR for medical applications. At Boeing [27], AR is being developed to assist in manufacturing processes. A group at the University of North Carolina has also explored medical applications [6], and conducted research in tracking technologies [5] and collision detection [4] for AR. Deering [11] has studied the problems associated with achieving high resolution head-tracked stereo display for AR. Milgram et al. [21] have explored AR for enhancing telerobotic interactions.

Grimson et al. [14] present a method for automatically registering clinical data from MRI or CT scans with a patient's head on an operating table. Nakamae et al. [23] propose a technique for accurately overlaying computer-generated images on digital images of real outdoor scenes. Fournier [13] has posed the problems associated with common illumination when combining synthetic images with images of real scenes. Wloka and Anderson [33] are developing a new real-time "depth-from-stereo" algorithm to be used for producing occlusions in AR.

3 The Grasp Augmented Vision System

The efforts of ECRC's UI&V group to explore augmented vision to date have revolved around the development of the Grasp system [1, 2]. Grasp is an object-oriented system written in C++ which provides an environment for exploring the basic technologies of augmented vision and for developing applications that demonstrate the capabilities of these technologies. The major software components consist of classes that implement geometric models, rendering algorithms, calibration methods, file I/O, user interfaces, and input devices.

The hardware configuration is illustrated in Figure 3.1. A graphical image is generated by the workstation hardware and displayed on the workstation's high resolution monitor along with auxiliary control information. A scan converter takes the relevant portion of the graphical image and converts it to standard video resolution and format. The scan converter also mixes this

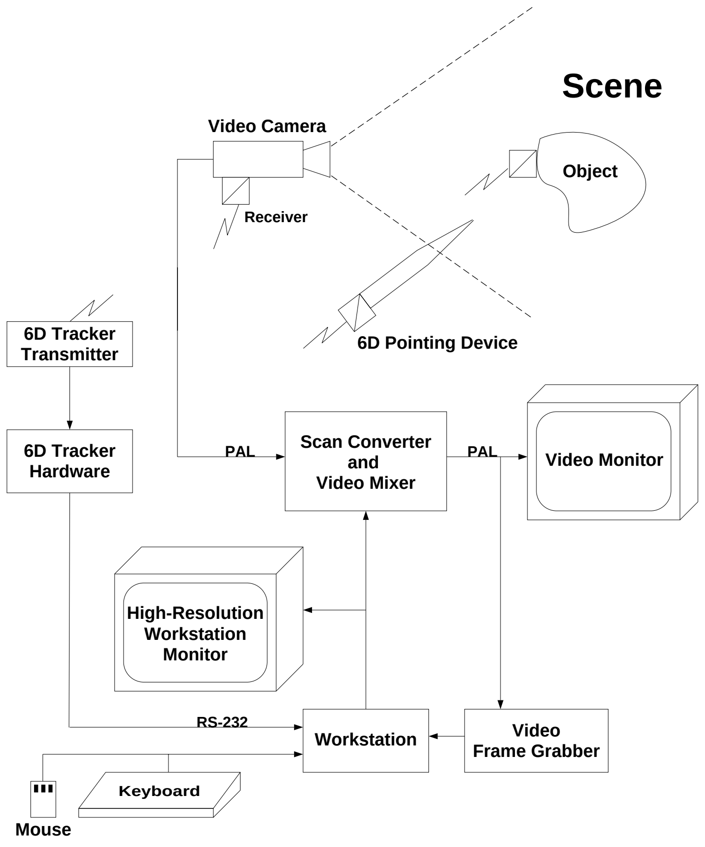

Figure 3.1: Grasp system hardware configuration.

generated video signal with the video signal input from the camera. A six degrees-of-freedom ($6D$) tracker, which is capable of sensing three translational and three rotational degrees of freedom, provides the workstation with continually updated values for the position and orientation of the video camera, a tracked object and the pointing device. The information about the camera and the tracked objects' position and orientation is used to keep the overlaid graphics in correct alignment with the visual image. A frame grabber is used during the initial calibration procedure as part of the process of determining the location and orientation of the camera, tracked objects, and real environment.

4 Modeling the Real World

In order for real and virtual objects to properly interact, they must be placed into the same computational framework. The geometric and physically-based modeling of virtual objects is a major research area that spans computer-aided design, computer animation, and virtual reality. Models of real objects must be created and brought into the virtual (computational) world before real and virtual objects may interact. The process of sensing the real world and automatically creating models of objects within it is traditionally a problem addressed by the computer vision community. Therefore the algorithms developed for computer vision play an extremely important role in augmented reality.

4.1 Calibrating the Input Devices

The first stage of modeling the real world involves calibrating the devices that are used to sense and measure it. Our system has two devices for obtaining information about the real world, a video camera and a $6D$ pointer. The camera is calibrated using a semi-automatic method based on the work of Weng et al. [30]. Here, an image of a known calibration grid is grabbed. The user interactively picks grid points in the image whose locations are known. By correlating the pixel locations (r_i, c_i) and the $3D$ coordinates of the points (x_i, y_i, z_i) with the equation

$$\begin{aligned} \frac{r_i - r_0}{f_u} &= \frac{r_{11}x_i + r_{12}y_i + r_{13}z_i + t_1}{r_{31}x_i + r_{32}y_i + r_{33}z_i + t_3} \\ \frac{c_i - c_0}{f_v} &= \frac{r_{21}x_i + r_{22}y_i + r_{23}z_i + t_2}{r_{31}x_i + r_{32}y_i + r_{33}z_i + t_3} \end{aligned} \tag{1}$$

the extrinsic parameters of camera orientation $\mathbf{R}$ and translation T, and the intrinsic parameters of image plane origin (r_0, c_0) and aspect-ratio-adjusted focal lengths (f_u, f_v) are calculated. By using at least six data points, Equation 1 forms a system of non-linear equations that is iteratively solved with a constrained optimization technique [2].

The pointing device may be calibrated by placing its tip on a known location T_p, and reading the position T_i and orientation $\mathbf{R_i}$ of its tracking receiver at least three times. These values are plugged into

$$T_p = T_b\mathbf{R_i} + T_i \tag{2}$$

and solved with a least-squares method to produce T_b, the effective length of the pointer [2]. Once T_b is known and the tracker system is calibrated, the $6D$ location of the pointer tracking receiver may be read and the world coordinate position of the pointer tip calculated.

4.2 Acquiring Models of Real-World Objects

Techniques for acquiring models of the real world typically fall into one of two classes as described in the computer vision literature. The first class consists of model-based techniques which assume some model of the world and try to align (register) that model to data from the real world, thereby inferring the pose of that object in world coordinates. If the object model can be represented in a form that is consistent with the other $3D$ models in the graphics system, then the object model can be used to simulate the effects of the real object it represents. The registration of models to match features in

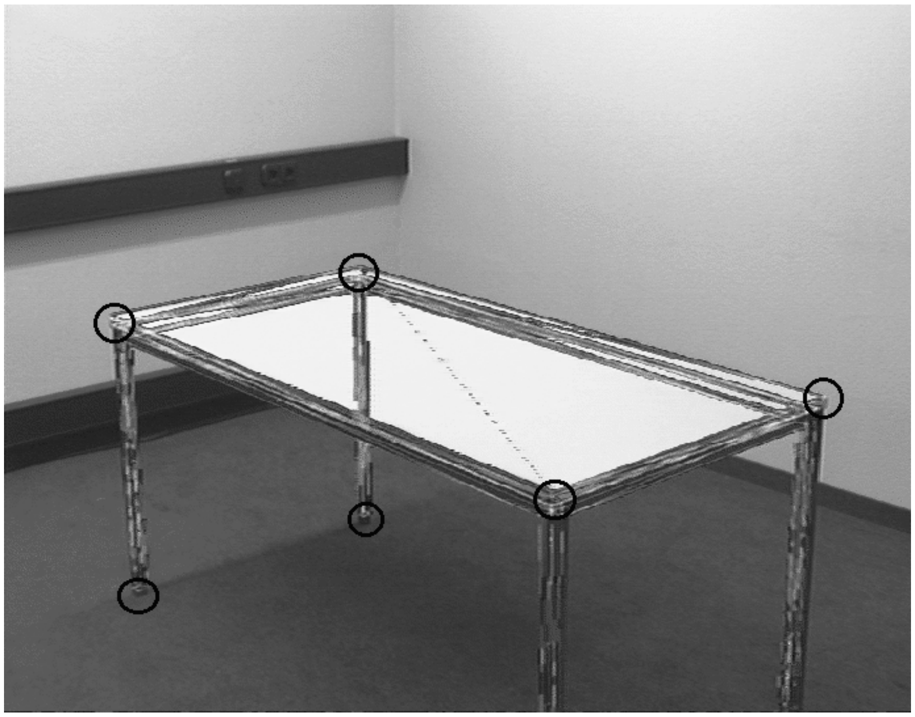

Figure 4.1: A wireframe model is overlaid on a real object after registration.

images is an on-going research problem in computer vision. However, applications of augmented reality do not necessarily require fully automated techniques. For the examples in the paper we have provided a registration tool that allows users to align objects by clicking on a number of points (r_i, c_i) in an image whose location (x_i, y_i, z_i) on the object are known (landmarks). Given this minimal user input the computer is then able to calculate the transformation $(\mathbf{R}, T)$ between the object's local coordinate system and the camera's coordinate system using Equation 1. At this point the camera's intrinsic parameters ((r_0, c_0), (f_u, f_v)) have been determined and are fixed; thus simplifying the system of equations needed to be solved. A constrained optimization technique is used that maintains the orthonormality of rotation matrix $\mathbf{R}$. Figure 4.1 presents the results of such a registration. The corresponding model is overlaid in wireframe on an image of the real object. The circles identify the landmarks that are picked by the user during the registration procedure.

The $6D$ pointing device may also be utilized to register geometric models to real objects. The pointer is used to acquire the world coordinates P_i^w of at least four landmarks known in the object's local coordinate system P_i^l. Plugging P_i^w and P_i^l into

$$P_i^w = P_i^l \mathbf{R} + T \tag{3}$$

and solving the resulting system of non-linear equations with a constrained optimization method produces the rotation matrix $\mathbf{R}$ and translation vector T. This local-to-world transformation provides the mapping, needed for object registration, from the object's local coordinate system to the world coordinate

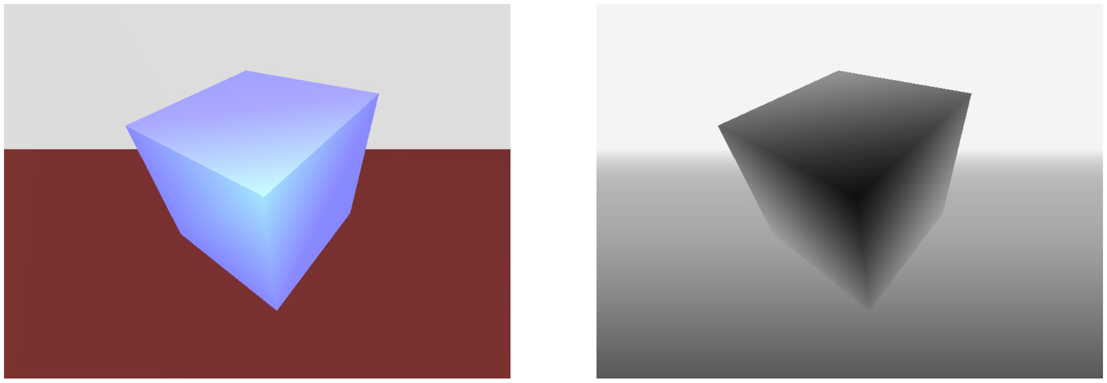

Figure 4.2: Rendered image and its corresponding depth map.

system.

A second class of techniques for acquiring models of the real world are those that reconstruct the depth of the real environment from the point of view of the camera (depth-based). While these techniques might use models of the real world, often they make more general assumptions, such as piecewise continuity or smoothness of surfaces in the scene. The absence of hard assumptions about the environment promises to produce algorithms that can be applied to a more general class of applications without having to construct geometric models of all the relevant real-world objects. Various techniques are described in the literature including shape from stereo [16], motion [31], shading [15], and texture [8]. These "shape from X" algorithms all produce the same kind of output, a depth image from the point of view of a camera. A depth map stores for each pixel in the real-world image the distance from the XY (image) plane of the camera to the real-world surface projected into that pixel. This is effectively a "$2\text{-}\frac{1}{2}D$" representation that stores a single value at each pixel of the image. Given the range of algorithms that produce depth maps, the interaction of objects with arbitrary depth maps is an important topic in virtual and real object interaction.

Since our group's research on producing depth maps from stereo image pairs is still in progress, "artificially-generated" depth maps were used to thoroughly test our depth-based algorithms. An "artificially-generated" depth map from a rendered image is shown in Figure 4.2. It is produced by first rendering a geometric model, and then reading the depth values computed by the graphics hardware from the Z-buffer. Finally, the hardware specific Z values are transformed into camera coordinates.

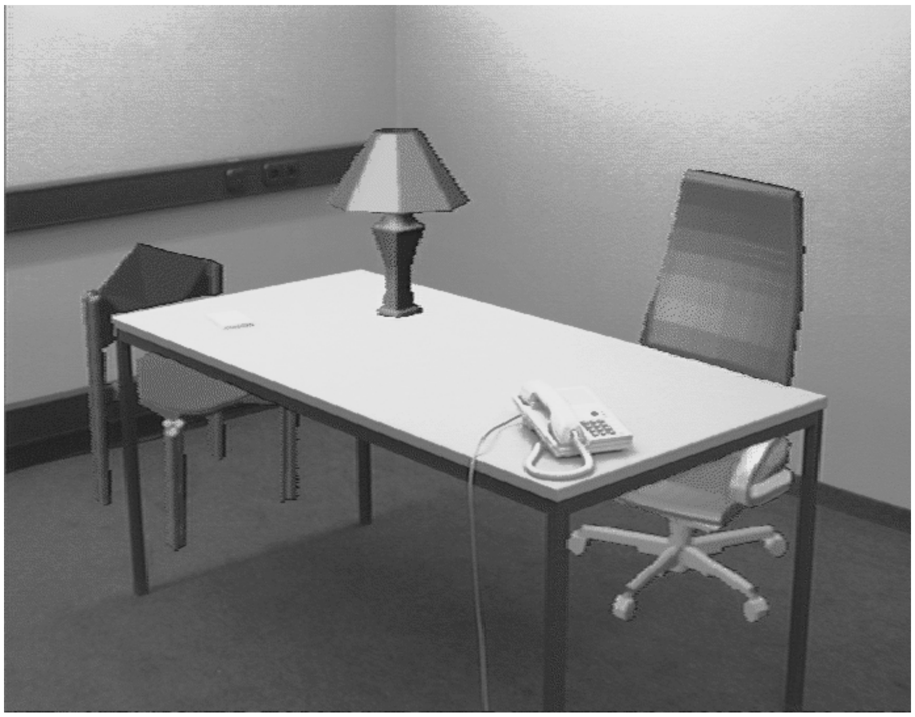

Figure 5.1: A real table occludes two virtual chairs in augmented reality, using the model-based technique.

5 Occlusion of Real and Virtual Objects

5.1 Virtual Occluding Real

In the Grasp system, live video and real-time computer graphics are merged and the results are displayed on a video monitor. Virtual objects that are to be included in the video signal are rendered on a black background. The video output of the graphics workstation is combined with the output of a video camera using luminance keying. In the final video output, the live signal is displayed in the black regions of the computer-generated signal. In this scenario, virtual objects always occlude the real ones in the final output, as seen in Figure 6.2. Displaying a non-black virtual object immediately hides the real object displayed in the same pixels in the live video signal. Virtual objects occlude real ones by default. Given our two approaches to modeling the real world, a variety of methods may now be employed to produce the interactive occlusion of virtual objects by real ones.

5.2 Real Occluding Virtual: Model-Based Method

Utilizing a model-based approach, we register geometric models of real objects to their real-world counterparts. Assuming that the model accurately represents the real object, this registration produces the modeling transformation that

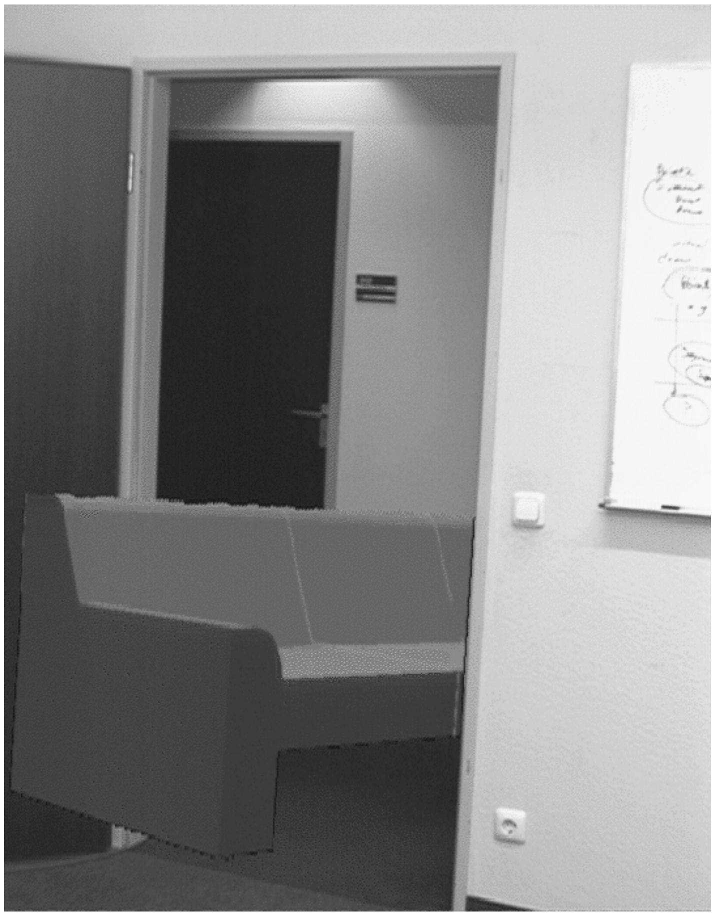

Figure 5.2: A virtual sofa coming through a real doorway, using the depth-based technique.

places the geometric model in the correct position in camera coordinates to produce an image identical to the live video image of the real object. In other words, when the geometric model is rendered, it will appear at the same location and orientation in the computer-generated image as the real object in the live video. Once the model of the real object is correctly positioned in our virtual world, the model can be used to produce occlusions by drawing it in black. Visible parts of the object will render in black, showing the live video through on the output monitor, due to our system's luminance keying. This effectively overlays the video image of the real object on top of its corresponding rendered model. As another a virtual object moves behind a model of a real object, the graphics hardware calculates the occlusion, but draws the forward visible model in black. This therefore produces the illusion that the real object is occluding the virtual one. Figure 5.1 presents two virtual chairs[3] placed around a real table. The surface and legs of the table correctly occlude the chairs using the model-based approach. A geometric model of the table has been registered to the real table as seen in Figure 4.1.

[3]The models used here are CAD models of real chairs taken from a telemarketing CD.

A second approach utilizes a depth map of the world to produce occlusions. The depth map may be tessellated and then decimated [25, 29] to produce a polygonal surface. The polygonal model represents the combined surfaces of the real-world objects seen from the video camera. This becomes in effect a geometric “model” of the current scene, and can be used with the method described above for occlusion. A more clever usage of the depth map information involves writing the camera-derived depth values directly into the Z-buffer of the graphics hardware, in an approach similar to Wloka and Anderson’s [33]. If at the beginning of each rendering cycle the hardware Z-buffer is initialized with the real-world depth values, occlusion of the virtual objects is performed automatically. When the virtual object is rendered, pixels that are further away from the camera than the Z values in the depth map are not drawn. By setting the background color to black, the real objects present in the original video are displayed in these unmodified pixels. Figure 5.2 presents a virtual sofa occluding and being occluded by a real doorway, using the depth-based approach. In this figure, we have first registered a geometric model of the room to the doorway, and then extracted the depth map from the graphics hardware’s Z-buffer.

The depth-based approach has the advantage that the scene can be arbitrarily complex, while the processing time remains a constant time function of image resolution. Additionally, no geometric model of the environment is needed during the interaction. Even if models of the environment are available, it may be more advantageous to convert them to a depth map, once the number of objects becomes too high. However, the depth map is dependent on the camera’s position and orientation, as well as the geometry of the environment. Once the camera or the real environment changes, the depth map becomes invalid. The model-based approach has the advantage that the occlusion information is stored as $3D$ geometry, making it valid from any viewpoint. One could also use object tracking to maintain the correspondence of the model to its real geometry in order to continue correct occlusion as the real object moves. It should also be emphasized that since we are directly utilizing the capabilities of our graphics hardware, the occlusions in Figures 5.1 and 5.2 are correctly calculated while the virtual objects are interactively manipulated.

6 Collision Detection Between Real and Virtual Objects

Once models of the real world are acquired, they may be utilized to produce other effects in augmented reality besides occlusion. If we know where the real world is, collisions may be detected between virtual objects and real objects. As a virtual object is interactively manipulated, its $3D$ position is checked against the $3D$ position of the objects in the real world. If a collision is detected, the virtual object is constrained and the user’s manipulation is ignored. Here again, we have identified two approaches for collision detection: model-based and depth-based. Once geometric models are registered to real-world objects, conventional object space methods for collision detection may be employed. This is a widely studied area of research [4, 7, 9, 10, 22, 28], which we have not yet begun to explore, and is beyond the scope of this paper. We have instead focused our efforts on utilizing camera-derived depth maps for collision detection.

6.1 Depth-Based Collision Detection

The first step in the depth-based collision detection process involves registering a virtual camera to the real camera. As previously stated, a variety of techniques may then be utilized to produce a view-dependent depth map of the real world from the video camera. Once completed, the depth of the real objects in our augmented reality environment from the camera's viewpoint is known. Applying a method similar to Shinya and Forgues' [26], a virtual object may be checked against the depth map for collisions as it is interactively manipulated. We currently check just the bounding box vertices of our virtual objects against the depth map in order to achieve interactive performance. Convex hulls, hierarchical bounding boxes, or polygonal vertices with $3D$ sorting may also be used to produce more accurate, but slower results.

For each analyzed point on the virtual object, collision detection is performed by first transforming the point into camera coordinates. This provides the mapped pixel location of the point, and the corresponding real-world depth may then be read from the depth map. If the point's Z-value is greater than the Z-value stored in the depth map for that particular pixel, a collision has occurred. During interactive manipulation, an object's previous transformation is stored at each rendering update. If a collision is detected at one of the object's bounding box vertices, the previous transformation is restored, preventing the virtual object from passing through the real object in augmented reality. Figure 6.1 is a sequence of images which demonstrates an object being interactively manipulated. As the object "strikes" the wall, i.e., its depth is greater than the wall's, it is stopped. The bounding box in the final image shows where the user wanted to place the object.

6.2 Simulating Gravity in Augmented Reality

Given that collisions can be detected between real and virtual objects, a simulated "gravity" capability may be implemented. This involves moving virtual objects in the direction of a gravity vector until a collision is detected. Currently, we define gravity in the positive Y direction in screen space, i.e., straight down on the screen. This vector is transformed first into camera coordinates, then into the local coordinate system of the virtual object to be

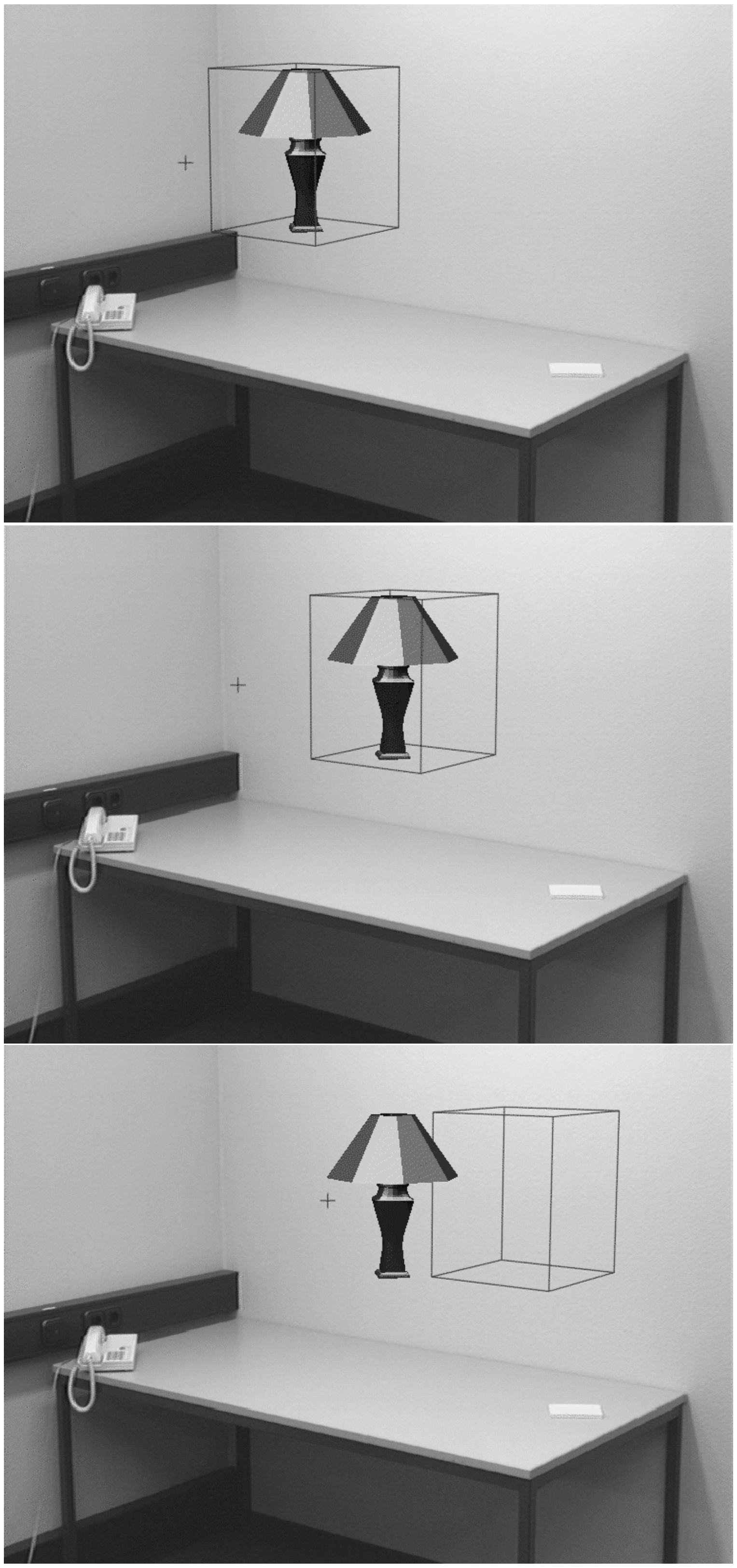
Figure 6.1: A virtual lamp hits a real wall in augmented reality.

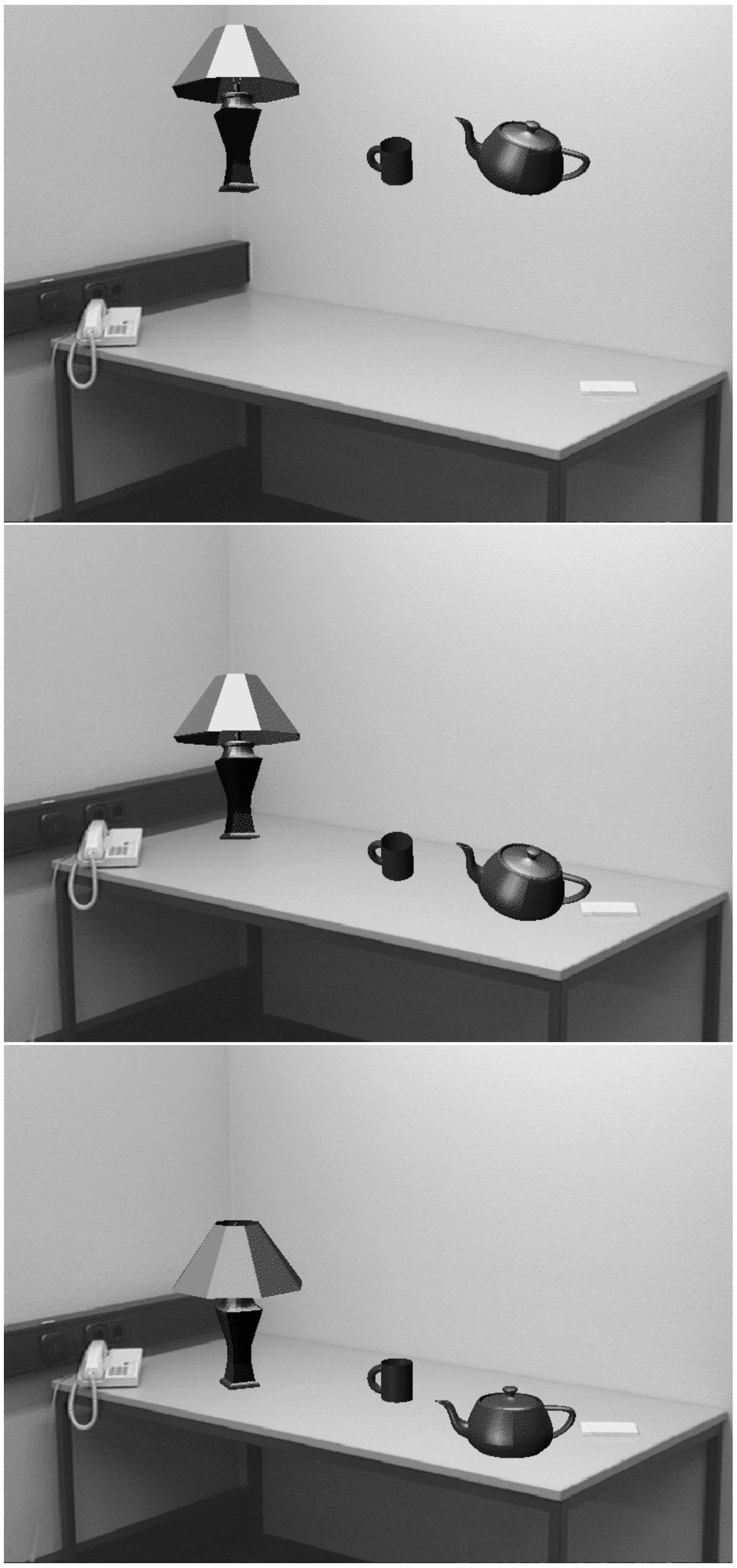

Figure 6.2: Three virtual objects falling on a real table.

"dropped". The incremental transformation that translates the virtual object down one pixel is calculated, and this transformation is successively applied until a collision is detected. Once one collision is detected, a simple torque calculation is performed around that collision point to produce a rotation. Again, a transformation is calculated that does not move any of the unconstrained bounding box vertices more than one pixel. The virtual object is rotated until another collision is detected. A final rotation is performed around the axis formed by the two constrained vertices until a third collision is detected. This capability allows us to place virtual items on a real table in augmented reality, as seen in Figure 6.2. Similar techniques can be used to move virtual objects to walls or other arbitrary surfaces. An arbitrary "gravity" vector may be defined that allows the user to specify in which direction virtual objects should fall. This promises to be an important tool in the Grasp system, allowing users to automatically place virtual objects in contact with real ones.

7 Discussion

Our goal has been to develop a general set of techniques that addresses the problem of occlusion and collision detection in augmented reality. The UI&V group is exploring a number of AR applications, including mechanical diagnostics [24], interior design [3], and computer-assisted surgery. Each of these application areas consists of its own set of requirements, a priori data, and geometric models. A variety of algorithms are needed to provide occlusion and collision detection in each of them. For each application the trade-off between the model-based and depth-based approaches must be considered. In a mechanical diagnostics application model-based approaches may be more suitable because CAD models of the examined objects may already be available. In an interior design application, generating depth maps may be the most straightforward way to acquire a model of the environment. In this case depth-based techniques are more appropriate. It is also possible to mix the approaches by using depth-based techniques for parts of the environment that remain static, and use model-based techniques, along with tracking, for modeling the dynamic objects in the real world.

The model-based approach can work well for scenes that appear complex; sometimes a simple model can sufficiently represent a complex-looking scene. Unfortunately, modeling small details is currently not possible because of the error ($\pm$1cm) introduced by camera calibration and object registration. Dealing with complex scenes or unknown objects is another shortcoming of the model-based approach. Deformable and parameterized models are a possible solution in this case. These are models that, with some interactive input, can be fit to objects in images. Utilizing parameterized models would allow one general model of a table to be fit to real tables of varying height or length during the registration process [18, 20]. Unknown objects in the scene can be modeled with deformable models, possibly using multiple camera views

[17, 32].

The techniques described here markedly improve the “reality” of augmented reality. Users often have difficulty grasping the scale and spatial location of virtual objects when they appear in the real scene. Occlusion and collision detection give the user cues about the virtual object in relation to its real environment. Simulated gravity and other similar techniques promise a better interaction model by transmitting forces and constraints from the real world to objects in the virtual world.

Since our occlusion and collision techniques rely on specific hardware capabilities, the performance and functionality of the available graphics hardware must also be considered. We obtained good interactive speeds (> 15 frames/sec) using the model-based occlusion and depth-based collision detection. Using the depth map directly for occlusion by loading the Z-buffer values is slow (1-2 frames/sec) due to the limitations of our workstation’s graphics software (Sun ZX graphics hardware and XGL software). We expect this problem to be solved as manufacturers realize the importance of writing into the Z-buffer interactively.

8 Conclusion

We have presented several techniques for interactively performing occlusion and collision detection between static real objects and dynamic virtual objects in augmented reality. Computer vision algorithms are used to acquire data that model aspects of the real world. Either geometric models may be registered to real objects, or a depth map of the real scene may be extracted with computer vision algorithms. The computer vision-derived data are mapped into algorithms that exploit the power of graphics workstations, in order to interactively produce new effects in augmented reality. By combining live video from a calibrated camera with real-time renderings of the real-world data from graphics hardware, dynamic virtual objects occlude and are occluded by static real objects. As a virtual object is interactively manipulated collisions with real objects are detected, and the motion of the virtual object is constrained. Simulated gravity may then be produced by automatically moving the virtual objects in the direction of a gravity vector until it encounters a collision with a real object.

9 Acknowledgements

We would like to thank Chris Crampton for assisting with the creation of the geometric models, and Mihran Tuceryan and Dieter Koller for providing computer vision references. This work is financially supported by Bull SA, ICL Plc, and Siemens AG.

Bibliography

[1] K. Ahlers, D. Breen, C. Crampton, E. Rose, M. Tuceryan, R. Whitaker, and D. Greer. An augmented vision system for industrial applications. In *SPIE Photonics for Industrial Applications Conference Proceedings*, October 1994.

[2] K. Ahlers, C. Crampton, D. Greer, E. Rose, and M. Tuceryan. Augmented vision: A technical introduction to the grasp 1.2 system. Technical Report ECRC-94-14, ECRC, Munich, Germany, 1994.

[3] K. Ahlers, A. Kramer, D. Breen, P.-Y. Chevalier, C. Crampton, E. Rose, M. Tuceryan, R. Whitaker, and D. Greer. Distributed augmented reality for collaborative design applications. Technical Report ECRC-95-03, ECRC, Munich, Germany, 1995.

[4] D. Aliaga. Virtual and real object collisions in a merged environment. In G. Singh, S. Feiner, and D. Thalmann, editors, *Virtual Reality Software & Technology (Proc. VRST '94)*, pages 287–298, Singapore, 1994. World Scientific Publishing Co.

[5] R. Azuma and G. Bishop. Improving static and dynamic registration in an optical see-through display. In *Computer Graphics (Proc. SIGGRAPH)*, pages 194–204, July 1994.

[6] M. Bajura, H. Fuchs, and R. Ohbuchi. Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. *Computer Graphics (Proc. SIGGRAPH)*, 26(2):203–210, July 1992.

[7] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics (Proc. SIGGRAPH)*, 24(4):19–28, August 1990.

[8] D. Blostein and N. Ahuja. Shape from texture: Integrating texture-element extraction and surface estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1233–1251, 1989.

[9] J. Boyse. Interference detection among solids and surfaces. *Communications of the ACM*, 22(1):3–9, January 1979.

[10] J. Canny. Collision detection for moving polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):200–209, 1986.

[11] M. Deering. High resolution virtual reality. *Computer Graphics (Proc. SIGGRAPH)*, 26(2):195–202, July 1992.

[12] S. Feiner, B. Macintyre, and D. Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7):53–62, July 1993.

[13] A. Fournier. Illumination problems in computer augmented reality. In *Journée INRIA, Analyse/Synthèse D'Images*, pages 1–21, January 1994.

[14] W. Grimson, T. Lozano-Perez, W. Wells, G. Ettinger, S. White, and R. Kikinis. An automatic registration method for frameless stereotaxy, image guided surgery, and enhanced reality. In *IEEE Conference on Computer Vision and Pattern Recognition Proceedings*, pages 430–436, Los Alamitos, CA, June 1994. IEEE Computer Society Press.

[15] B. Horn and M. Brooks. *Shape from Shading*. MIT Press, Cambridge, MA, 1989.

[16] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. In *IEEE International Conference on Robotics and Automation Proceedings*, pages 1088–1095, Los Alamitos, CA, April 1991. IEEE Computer Society Press.

[17] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[18] D. Koller, K. Daniilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 1993.

[19] W. Lorensen, H. Cline, C. Nafis, R. Kikinis, D. Altobelli, and L. Gleason. Enhancing reality in the operating room. In *Visualization '93 Conference Proceedings*, pages 410–415, Los Alamitos, CA, October 1993. IEEE Computer Society Press.

[20] D. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:441–450, 1991.

[21] P. Milgram, S. Shumin, D. Drascic, and J. Grodski. Applications of augmented reality for human-robot communication. In *International Conference on Intelligent Robots and Systems Proceedings*, pages 1467–1472, Yokohama, Japan, July 1993.

[22] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Computer Graphics (Proc. SIGGRAPH)*, 22(4):289–298, August 1988.

[23] E. Nakamae, K. Harada, and T. Ishizaki. A montage method: The overlaying of the computer generated images onto a background photograph. *Computer Graphics (Proc. SIGGRAPH)*, 20(4):207–214, August 1986.

[24] E. Rose, D. Breen, K. Ahlers, C. Crampton, M. Tuceryan, R. Whitaker, and D. Greer. Annotating real-world objects using augmented vision. Technical Report ECRC-94-41, ECRC, Munich, Germany, 1994.

[25] W. Schroeder, J. Zarge, and W. Lorensen. Decimation of triangle meshes. *Computer Graphics (Proc. SIGGRAPH)*, 26(2):65–70, July 1992.

[26] M. Shinya and M.-C. Forgue. Interference detection through rasterization. *Journal of Visualization and Computer Animation*, 2:132–134, 1991.

[27] D. Sims. New realities in aircraft design and manufacture. *IEEE Computer Graphics and Applications*, 14(2):91, March 1994.

[28] J. Snyder, A. Woodbury, K. Fleischer, B. Currin, and A. Barr. Interval methods for multi-point collisions between time-dependent curved surfaces. In *Computer Graphics (Proc. SIGGRAPH)*, pages 321–334, August 1993.

[29] G. Turk. Re-tiling polygonal surfaces. *Computer Graphics (Proc. SIGGRAPH)*, 26(2):55–64, July 1992.

[30] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-14(10):965–980, 1992.

[31] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(5):451–476, May 1989.

[32] R. Whitaker. Algorithms for implicit deformable models. Technical Report ECRC-94-42, ECRC, Munich, Germany, 1994.

[33] M. Wloka and B. Anderson. Resolving occlusion in augmented reality. In *ACM Symposium on Interactive 3D Graphics Proceedings*, April 1995.