Interactive Occlusion and Automatic Object Placement for Augmented Reality

David E. Breen† Ross T. Whitaker‡ Eric Rose* Mihran Tuceryan§

†Computer Graphics Lab, California Institute of Technology, Pasadena, CA 91125, USA, david@gg.caltech.edu
‡Electrical Engineering Dept., University of Tennessee, Knoxville, TN 37996, USA, whitaker@vision1.engr.utk.edu
*European Computer-Industry Research Centre, D-81925 Munich, Germany, erose@ecrc.de
§Corporate Research Center, Texas Instruments, Dallas, TX 75265, USA, tuceryan@csc.ti.com

Abstract
We present several techniques for producing two visual and modeling effects in augmented reality. The first effect involves interactively calculating the occlusions between real and virtual objects. The second effect utilizes a collision detection algorithm to automatically move dynamic virtual objects until they come in contact with static real objects in augmented reality. All of the techniques utilize calibrated data derived from images of a real-world environment.

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[1]. 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[†] within the user's environment must be seamlessly 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.

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 reality capability[2]. Our initial work in this 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. We are currently not investigating new collision detection algorithms for AR, but instead are using previously developed algorithms to implement new interaction effects. Given a collision detection capability, it is possible to automatically move a virtual object until a collision is detected, thus placing it in contact with real objects, and simulating virtual "gravity". In

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

this paper we present two approaches, model-based and depth-based, for calculating interactive occlusions, and performing automatic object placement. The advantages and disadvantages of each technique is discussed within the context of various AR applications.

Computer vision algorithms for acquiring information about the real world form the foundation of our occlusion and object placement work. 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 object placement for 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 automate many of the registration, tracking, and model generation tasks needed for AR, and also to use CV-derived data in a way that exploits the power of graphics workstations in order to produce interactive effects. Camera and tracker calibration methods provide us with the ability to sense and measure the real world. Once object models are registered to real-world objects or depth maps generated from images of a real scene, they both may be passed directly to a graphics system to produce occlusions. Calibrated object models and depth maps are also utilized in collision detection during automatic object placement.

In the remainder of the paper, we briefly describe related work, introduce our augmented reality system and the model acquisition and calibration techniques that are implemented within it. This is followed by more detailed descriptions of our occlusion and object placement algorithms, which utilize both a depth-based and a model-based approach. We close with discussion of the advantages and disadvantages of these two approaches.

2. Previous Work

Numerous groups around the world are now investigating augmented reality. For an excellent overview of augmented reality technologies, projects and applications, see the survey by Azuma[1]. Some of these groups are exploring the issue of real and virtual object interaction in augmented reality. Fournier[3] has posed the problems associated with common illumination when combining synthetic images with images of real scenes. Here, lighting information is shared between real and virtual environments. Wloka and Anderson[4] are developing a new real-time "depth-from-stereo" algorithm to be used for producing occlusions in AR. In their work they attempt to create depth maps of real scenes interactively. In the process, they sacrifice accuracy for performance. They then utilize the depth information to resolve occlusions in augmented reality. Their depth maps may be interactively generated, but their visual results are unsatisfactory with numerous errors. Aliaga[5] has introduced the concept of collision detection and response for augmented reality, within the context of the VROC system. He has implemented a collision detection algorithm by Lin and Canny[6] and has tested it within a VR and AR setting. Both the Wloka and the Aliaga work have introduced the concepts of occlusion resolution and collision detection for augmented reality, but they both suffer from the same weakness. Neither work involves calibrating a camera and the real-world objects used in their systems. In contrast, the UI&V group of ECRC has rigorously studied and implemented a spectrum of calibration methods for augmented reality[7]. These calibration methods, along with some new techniques described in this paper provide more accurate and improved visual results.

3. Modeling the Real World in the Grasp Augmented Reality System

We have conducted our augmented reality research within the Grasp system[2]. Grasp is an object-oriented system written in C++ which provides an environment for exploring the basic technologies of monitor-based augmented reality and for developing applications that demonstrate the capabilities of these technologies. Since Grasp is a monitor-based AR system, as opposed to a head-mounted display-based system, we capture live video with a camera, and merge it with real-time computer generated graphics using luminance keying. A magnetic tracking system is used to track the camera, a pointing device, and objects of interest. The tracking information is used to maintain the alignment of the video signal and the computer-generated graphics.

In order for real and virtual objects to properly interact, they must be placed into the same computational framework. Models of real objects must first be created and brought into the virtual (computational) world. All of the models are then related to a single coordinate system. Towards this end, a series of model acquisition and calibration procedures have been developed within Grasp[7, 8]. These procedures allow us to sense, acquire, and calibrate models of real-world objects. These models are essential components in all of our augmented reality applications and algorithms, including those involving occlusion and automatic object placement.

The first stage of modeling the real world involves calibrating the devices that are used to sense and measure it[7]. Our system has two devices for obtaining information about the real world, a video camera and a 6-DOF pointer. The camera is calibrated with an automatic method that utilizes the mathematics derived by Weng et al.[9]. Calibration provides the pose of the camera and its internal imaging parameters. The pointing device may be calibrated by placing its tip on a known location several times with different orientations. This allows us to calculate its length and the transformation relating the pointer to the tracking system.

Techniques for acquiring models of the real world[8] 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. For the examples in the paper we have utilized a registration tool that matches a number of points (r_i, c_i) in an image with landmarks whose location (x_i, y_i, z_i) in the environment are known. The points (r_i, c_i) may be manually identified by a user or automatically located by the system if the environment is labeled with a pre-determined pattern of black squares. Given this minimal 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. Figure 1 presents the results of a manual 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. Figure 2 presents the results of an automatic registration. The virtual furniture has been automatically registered to the real room.

The 6-DOF 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. A local-to-world transformation may then be calculated to provide the mapping, needed for object registration, from the object's local coordinate system to the world coordinate system. Error information for both these techniques may be found in[7].

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. Various techniques are described in the literature including shape from stereo[10], shading[11], and texture[12]. These "shape from X" algorithms all produce some kind of depth information about the environment, either sparse or dense. In our work we generate a dense depth map, which 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. Figures 3 and 4 present the right view from a stereo pair of images, and the depth map generated from the stereo pair using Weng's algorithm[10].

4. Occlusion of Real and Virtual Objects

4.1. 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. Similarly to a number of other groups[13, 14, 15], 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 7. 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.

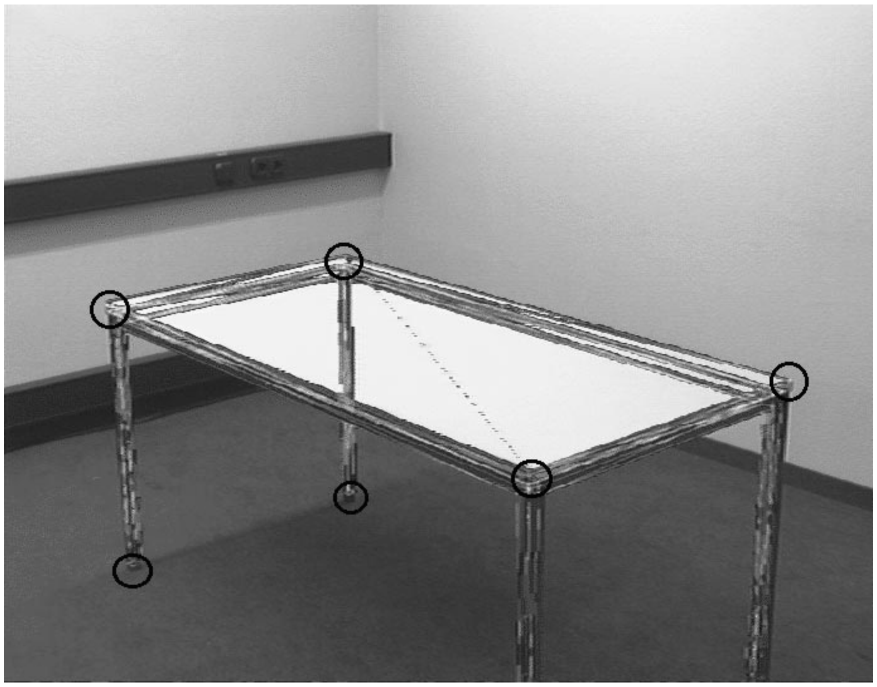

Figure 1: *A wireframe model is overlaid on a real object after registration.*

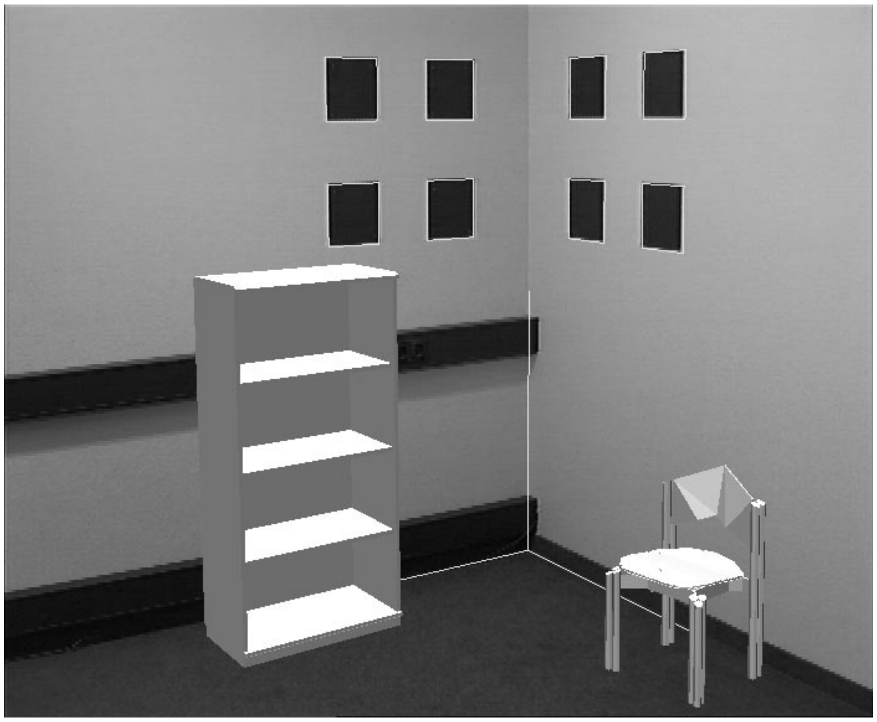

Figure 2: *Virtual furniture accurately merged with a real room after automatic calibration with black squares.*

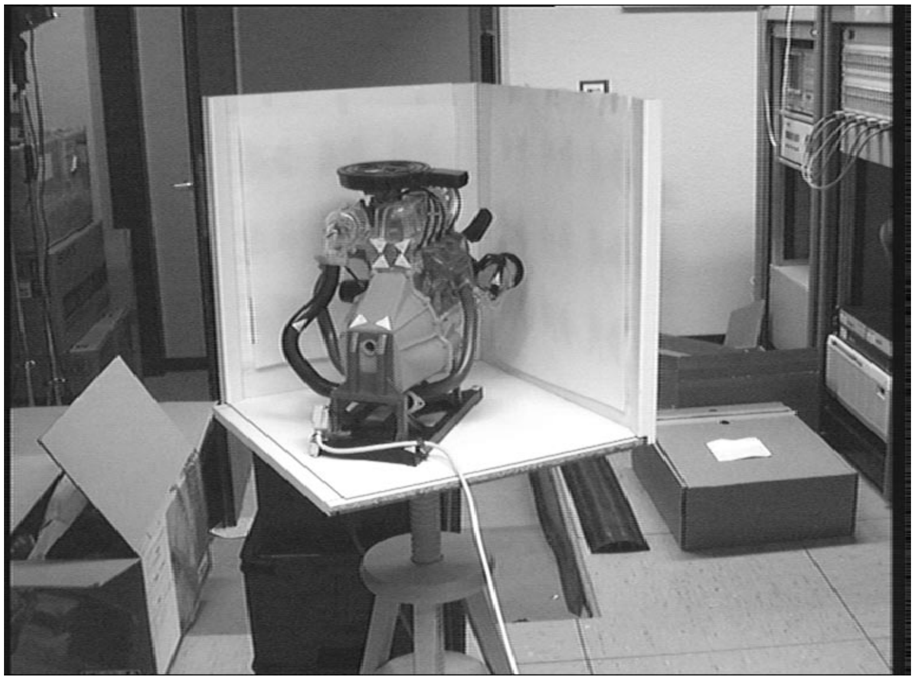

Figure 3: *The right view of a stereo pair.*

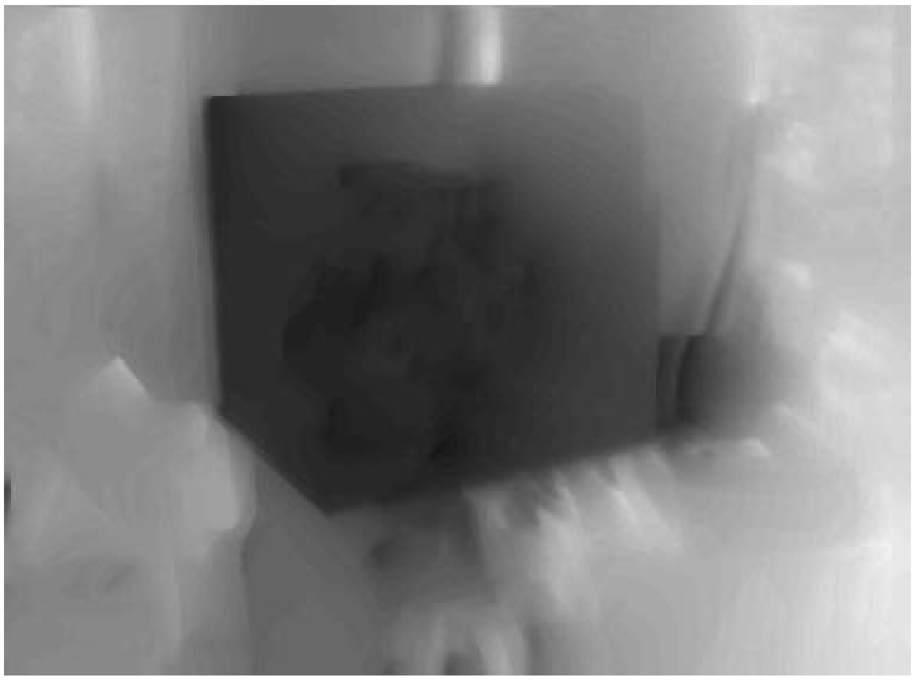

Figure 4: *Depth map corresponding to Figure 3, where brighter pixels represent points farther away.*

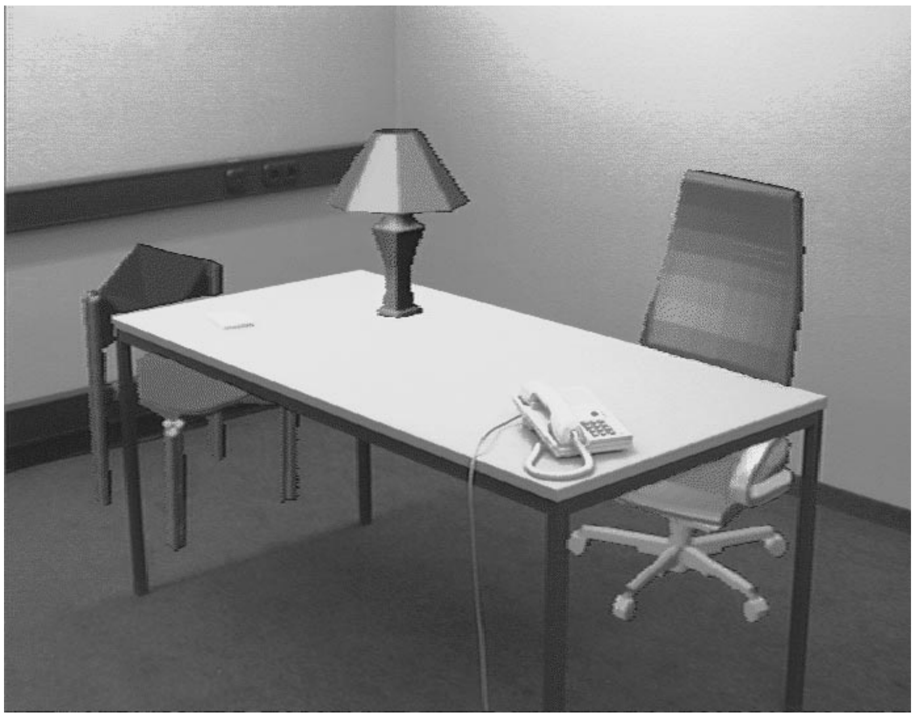

Figure 5: *A real table occludes two virtual chairs in augmented reality, while being occluded by a virtual lamp, using the model-based technique.*

4.2. 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 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 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 presents two virtual chairs 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 1.

4.3. Real Occluding Virtual: Depth-Based Method

A second approach utilizes a depth map of the world to produce occlusions. The depth map may be tessellated and then decimated[16] 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. Another 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[4]. 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

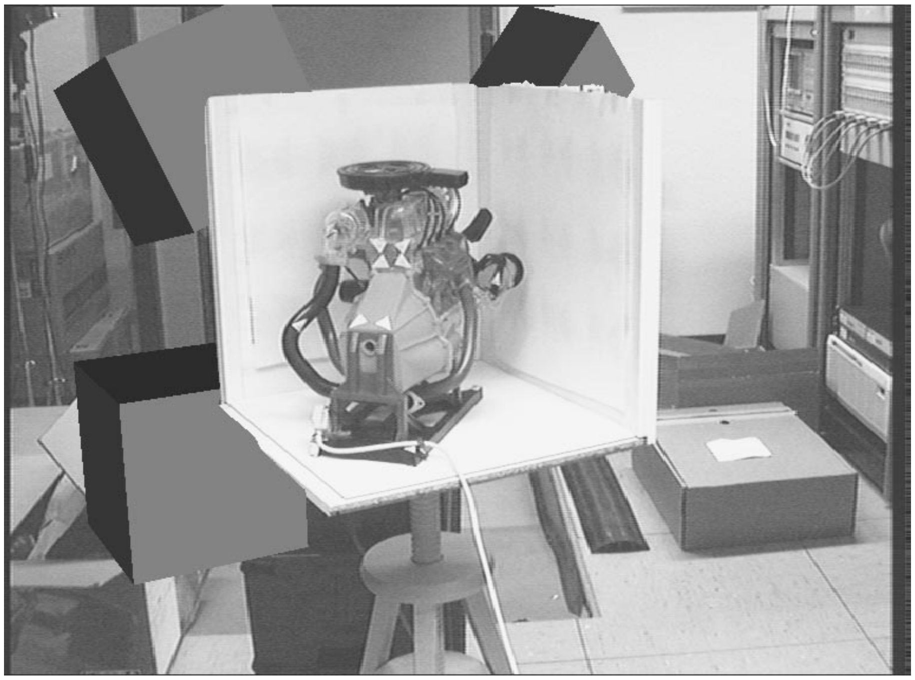

Figure 6: *Three virtual cubes occluded by a wooden stand and occluding the room background, using the depth-based technique.*

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 6 presents three virtual cubes occluded by a wooden stand and occluding the other objects in a real room, using the depth-based approach that writes depth values into a workstation's Z-buffer. The depth map used in this example is presented in Figure 4.

5. Automatic Object Placement

Once models of the real world are acquired, they may be utilized to produce other effects in augmented reality besides occlusion. Another interactive feature that we have implemented within the Grasp system allows us to automatically place virtual objects on top of real ones within augmented reality. This can be a useful component of an augmented reality interior design system[17], for example. This feature lets the user interactively move furniture, with collision detection, around a real room into the approximately correct position. Automatic object placement would then ensure that the virtual furniture rested on the real floor, or up against a real wall. This could also be used to place virtual items on real tables.

5.1. Collision Detection

The fundamental capability supporting automatic object placement is collision detection. If we know where the real world is, collisions may be detected between virtual objects and real objects. As a virtual object moves, its $3D$ position may be checked against the $3D$ position of the objects in the real world. Once a collision is detected, a variety of responses may be calculated[5]. As with occlusion, we have identified two approaches for collision detection: model-based and depth-based. The model-based approach involves calculating collisions between the 3-D geometric models that describe the real and virtual objects in the augmented environment. In the depth-based approach, the geometric model of the virtual object is checked against the depth map representing the real environment.

Once geometric models are registered to real-world objects for the model-based approach, conventional object space methods for collision detection may be employed. This is a widely studied area of research[6, 18, 19, 20, 21], which we have only begun to explore, and is beyond the scope of this paper. Currently our group is implementing a hierarchical collision detection scheme similar to the ones proposed by Moore and Wilhelms[18], and Hahn[19]. To date, we have focused our efforts on utilizing camera-derived depth maps for 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'[22], 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. This has proven sufficient for our applications. The hierarchical bounding boxes of subparts, convex hull vertices, or all model vertices may also be utilized. 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 with a visible surface has occurred.

5.2. Placing Virtual Objects in Contact with Real Objects

Automatic object placement has been implemented using our ability to detect collisions between real and virtual objects with depth maps. This involves incrementally moving virtual objects in the direction of a "gravity" vector until a collision is detected. The gravity vector may be arbitrarily defined by the user in screen or object space. Currently, we define gravity in the positive Y direction in screen space, i.e., straight down on the screen. Our object placement function incrementally translates and rotates a virtual object such that it changes its position in image space one pixel at a time. We then check its bounding box vertices for collisions. Moving one pixel at a time ensures that the virtual object does not pass through any objects.

Several steps are required to calculate the transformation that will move a virtual object down one pixel in the final output image. First, each bounding box vertex is transformed into camera coordinates, where the origin is located at the camera focal point, and the x and y axes define the image plane, and the z-axis is the viewing direction. The vertex closest to the camera is chosen, noting its distance to the camera focal point, i.e., its z coordinate. This point will appear to move fastest due to the perspective projection. Therefore, all transformation calculations are performed with this point to ensure that no point on the virtual object moves more than one pixel per step. The pixel coordinates, (i, j), of the closest vertex P_0, (x_i, y_j, z), are calculated by projecting (x_i, y_j, z) onto the image plane. Pixel $(i, j+1)$ is then projected back into camera coordinates, assuming the same z coordinate as the previous point, to give point P_1, (x_i, y_{j+1}, z). $P_1 - P_0$ gives the "gravity" vector in camera coordinates, G_{cc}. In other words, adding G_{cc} to P_0 produces P_1. This change in camera coordinates position moves point P_0 from pixel (i, j) to pixel $(i, j+1)$ on the image plane. G_{cc} is then transformed into the coordinate system of the virtual object's parent object to give G_{pc}. G_{pc} is successively added to the local transformation of the "dropping" virtual object, moving it one pixel at a time, until a collision is detected at one of the bounding box vertices.

Once one collision is detected, a torque calculation is performed around the collision point P_c to produce a rotation. The individual torques associated with each non-constrained bounding box vertex are summed,

$$N = \sum_i (P_i - P_c) \times G_{pc}, \tag{1}$$

to produce the axis of rotation N. All calculations are performed in the coordinate system of the virtual object's parent. Again, a transformation is needed that does not move any of the unconstrained bounding box vertices more than one pixel. We assume that the vertex, P_{max}, with the longest moment arm, i.e., the maximum $(P_i - P_c)$, will move the greatest distance in the image when the virtual object is rotated. A "gravity" vector G_{cc} is once again calculated by determining the distance vertex P_{max} must move in order for it to move one pixel in the Y direction on the image plane. The angle to achieve this one pixel move is calculated by

$$\theta = \arccos((P_{max} \cdot (P_{max} + G_{cc}))/(|P_{max}| * |P_{max} + G_{cc}|)). \tag{2}$$

The virtual object is then incrementally rotated by θ around the axis N which runs through the constrained

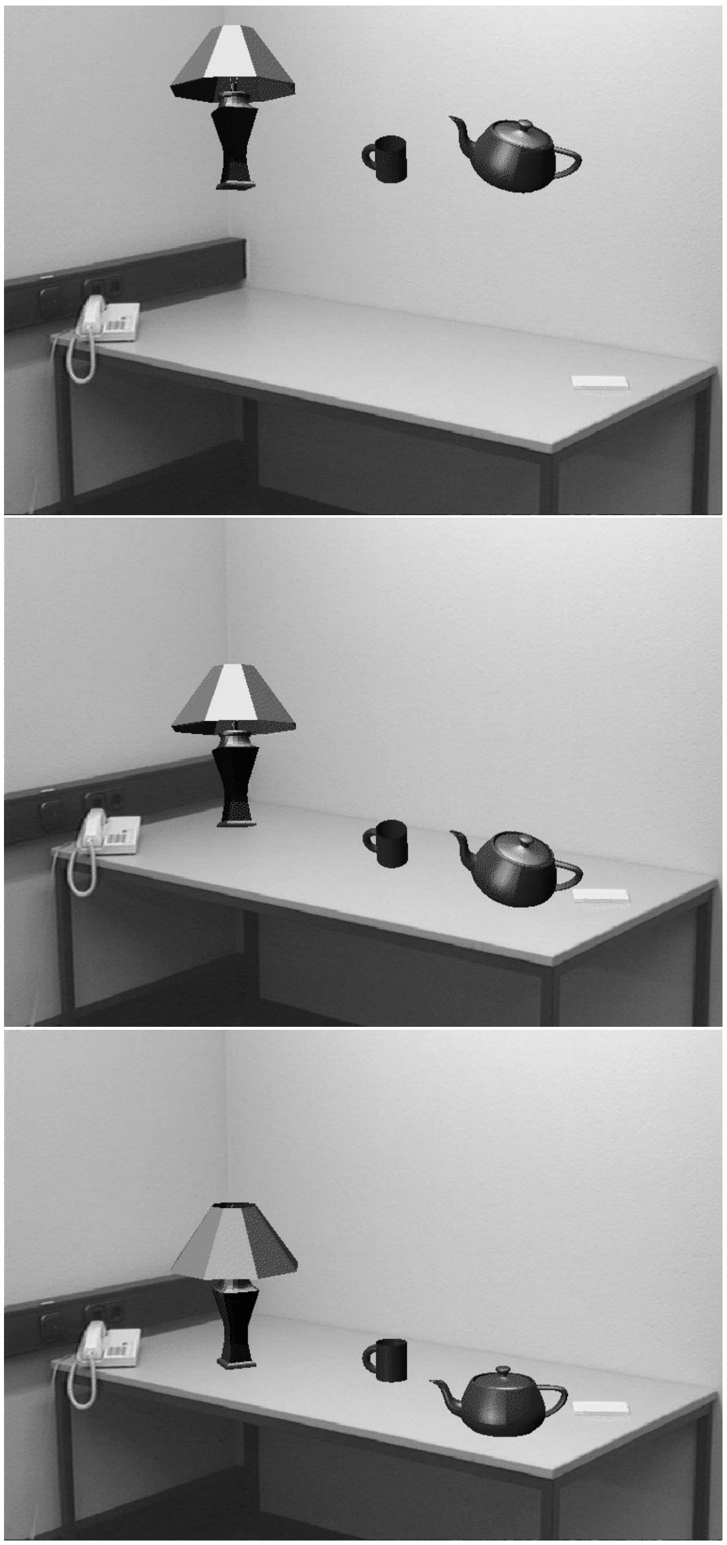

Figure 7: *Three virtual objects automatically placed on a real table.*

	Occlusion	Collision Detection
Depth-Based	ECRC Wloka & Anderson	ECRC
Model-Based	ECRC	ECRC (in progress) Aliaga

Figure 8: *Matrix of augmented reality effects and methods.*

point P_c, until a second collision is detected at another bounding box vertex. At this point similar calculations are performed to incrementally rotate the virtual object around the axis formed by the two constrained vertices until a third and final collision is detected. The virtual object is now fully at rest on a real object. An example of automatic object placement is presented in Figure 7. Three virtual objects are translated down until their bounding boxes come into contact with the real table. They are then rotated until three of their bounding box vertices touch the table.

6. Evaluation of Approaches

One of the goals of the UI&V group is the development of a general set of techniques that addresses the problem of occlusion and automatic object placement in augmented reality. To date, our group has developed two AR applications, one for mechanical diagnostics[23], the other for interior design[17]. Each application consists of its own set of requirements, a priori data, and geometric models. Therefore, a variety of algorithms are needed to provide occlusion and object placement/collision detection in each of them. As shown in Figure 8 we have presented four different techniques which may be placed into a 2×2 matrix. The chart also lists the other groups working in this area.

In each AR 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 other applications geometric models of the relevant objects may not be available. In these cases 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, such as the background, and use model-based techniques, along with tracking, for modeling the dynamic objects in the real world.

The depth-based and the model-based approaches each have their own advantages and disadvantages. The depth-based algorithms have the advantage that their processing time is order O(n), where n is the number of vertices checked against the depth map. The depth checking procedure takes constant time, being independent of scene complexity. All operations simply become table look-ups and a compare. This may be ideal for scenes with a static background. 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. Additionally, aliasing may be a problem when utilizing the depth map. Object features that map into a single pixel will be lost and their depth may not be captured.

Since the depth-based approaches are view-dependent, the algorithms can only account for surfaces that can be seen. In the depth-based object placement algorithm, the environment is assumed to be solid behind the visible surfaces. This is obviously not always true and prevents virtual objects from being placed behind real objects in this specific scenario. Therefore, occlusion and depth-based object placement may not be performed simultaneously. Instead collision detection must either be suspended to place a virtual object behind a real one, or an object-based collision detection technique should be utilized.

The model-based and depth-based approaches have complementary advantages and disadvantages. The model-based approach has the advantage that the occlusion information is stored as $3D$ geometry, making it valid from any viewpoint. This will support more complex and accurate interactions. Aliasing is also not a concern, with all aspects of the real world theoretically capable of being modeled. In practice, though, generating a detailed geometric model of a complex environment may be difficult or impossible. It requires matching pre-existing parametric geometric models to real objects. These parameterized models may not exist for all objects in a real scene. As real scenes become more complex, so do their geometric model counterparts. As the geometric models grow in size, interactive processing of them becomes more difficult, unlike in the depth-based approach. Ultimately, both approaches have the advantage of producing data that may be directly utilized by graphics workstations to produce interactive results.

It should be emphasized that since we are directly utilizing the capabilities of our graphics hardware, the occlusions in Figures 5 and 6 are correctly calculated while the virtual objects are interactively manipulated. Since our occlusion and object placement 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 SparcStation 10 with ZX graphics hardware and XGL software). We expect the performance to improve as manufacturers realize the importance of writing into the Z-buffer interactively.

7. Conclusion

We have presented several techniques for calculating interactive occlusions and performing automatic object placement with 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 produce interactive 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. By utilizing collision detection algorithms, virtual objects may be automatically placed on real objects in augmented reality. Simulated gravity is produced by automatically moving the virtual objects in the direction of a gravity vector until it encounters a collision with a real object.

8. Acknowledgements

This research was conducted at the European Computer-Industry Research Centre. We would like to thank the remainder of the UI&V Group, Klaus Ahlers, Chris Crampton, Doug Greer, Gudrun Klinker, Dieter Koller, and Steven Prestwich, who have all contributed to the development of the Grasp system. This work was financially supported by Bull SA, ICL PLC, and Siemens AG.

References

1. R. Azuma, "A survey of augmented reality", in *ACM SIGGRAPH '95 Course Notes #9 - Developing Advanced Virtual Reality Applications*, (August 1995).

2. K. Ahlers, D. Breen, C. Crampton, E. Rose, M. Tuceryan, R. Whitaker, and D. Greer, "An augmented vision system for industrial applications", in *Telemanipulators and Telepresence Technologies*, vol. 2351, pp. 345–359, SPIE Proceedings, (October 1994).

3. A. Fournier, "Illumination problems in computer augmented reality", in *Journée INRIA, Analyse/Synthèse D'Images*, pp. 1–21, (January 1994).

4. M. Wloka and B. Anderson, "Resolving occlusion in augmented reality", in *Symposium on Interactive 3D Graphics Proceedings*, (New York), pp. 5–12, ACM Press, (April 1995).

5. D. Aliaga, "Virtual and real object collisions in a merged environment", in *Virtual Reality Software & Technology (Proc. VRST '94)* (G. Singh, S. Feiner, and D. Thalmann, eds.), (Singapore), pp. 287–298, World Scientific Publishing Co., (1994).

6. M. Lin and J. Canny, "Efficient collision detection for animation", in *Third Eurographics Workshop on Animation and Simulation Proceedings*, (Cambridge, UK), (September 1992).

7. M. Tuceryan, D. Greer, R. Whitaker, D. Breen, C. Crampton, E. Rose, and K. Ahlers, "Calibration requirements and procedures for a monitor-based augmented reality system", *IEEE Transactions on Visualization and Computer Graphics*, **1**(3), pp. 255–273 (1995).

8. R. Whitaker, C. Crampton, D. Breen, M. Tuceryan, and E. Rose, "Object calibration for augmented reality", in *Proceedings of Eurographics '95 Conference*, (Maastricht, NL), pp. C–15–C–27, (August 1995).

9. 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), pp. 965–980 (1992).

10. J. Weng, T. 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), pp. 451–476 (1989).

11. B. Horn and M. Brooks, *Shape from Shading.* Cambridge, MA: MIT Press, (1989).

12. 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), pp. 1233–1251 (1989).

13. 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), pp. 203–210 (1992).

14. W. Lorensen, H. Cline, C. Nafis, R. Kikinis, D. Altobelli, and L. Gleason, "Enhancing reality in the operating room", in *Visualization '93 Conference Proceedings*, (Los Alamitos, CA), pp. 410–415, IEEE Computer Society Press, (October 1993).

15. 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*, (Yokohama, Japan), pp. 1467–1472, (July 1993).

16. W. Schroeder, J. Zarge, and W. Lorensen, "Decimation of triangle meshes", *Computer Graphics (Proc. SIGGRAPH)*, **26**(2), pp. 65–70 (1992).

17. 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", in *Proceedings of Eurographics '95 Conference*, (Maastricht, NL), pp. C–03–C–14, (August 1995).

18. M. Moore and J. Wilhelms, "Collision detection and response for computer animation", *Computer Graphics (Proc. SIGGRAPH)*, **22**(4), pp. 289–298 (1988).

19. J. Hahn, "Realistic animation of rigid bodies", *Computer Graphics (Proc. SIGGRAPH)*, **22**(4), pp. 299–308 (1988).

20. D. Baraff, "Curved surfaces and coherence for non-penetrating rigid body simulation", *Computer Graphics (Proc. SIGGRAPH)*, **24**(4), pp. 19–28 (1990).

21. 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)*, pp. 321–334, (August 1993).

22. M. Shinya and M.-C. Forgue, "Interference detection through rasterization", *Journal of Visualization and Computer Animation*, **2**, pp. 132–134 (1991).

23. E. Rose, D. Breen, K. Ahlers, C. Crampton, M. Tuceryan, R. Whitaker, and D. Greer, "Annotating real-world objects using augmented reality", in *Computer Graphics: Developments in Virtual Environments (Proceedings of CG International '95 Conference)*, (Leeds, UK), pp. 357–370, (June 1995).