Appeared: Communications of the ACM (CACM)
Vol. 40. No. 3., pp. 49-54, March, 1997

Virtual Objects in the Real World

Daniel G. Aliaga
Computer Science Department
University of North Carolina
Chapel Hill, NC 27599-3175

Virtual Reality usually refers to systems that generate a totally synthetic environment. The end-user is able to specify all the characteristics of the new environment. A head-mounted display and an orientation and position tracker enable the user to roam around the new virtual world looking up, down and walking into rooms that don't really exist! This capability allows the user to experience environments which are not yet created (e.g., architectural walkthroughs) or position himself in worlds he can never really visit (e.g., exploring a surface at an atomic level).

There are, however, situations where the user might want to remain in the real world and instead of completely replacing the real world with a virtual world, might wish to merge the virtual world with the real world. An Augmented Reality system (see figure 1) employs a see-through head-mounted display and an image generation system to present the user with a world consisting simultaneously of virtual and real objects. Such a system could allow an architect to make actual-size modifications to an existing building, a home-owner to decorate an empty (real) house, or children to design and build virtual toys which could be used simultaneously with real toys. Yet another application could allow a doctor to view ultrasound images of a fetus registered in place over a pregnant women's womb [Bajura92]. The KARMA system [Feiner93] uses an Augmented Reality system to assist the user in performing 3D tasks. This system combines an Intent-Based Illustration system with a prototype Augmented Reality system to explain simple laser printer maintenance tasks.

Figure 1: An Augmented Reality System. A see-through head-mounted display is connected to an image generation system to present to the user a world consisting of both virtual and real objects.

Objects in the real world are affected by phenomena such as gravity, friction and collisions. Future applications of merged virtual and real environments might wish to model such phenomena; otherwise, the interaction of the two worlds may not be convincing at all. Significant work must be done before a virtual environment convincingly simulates these phenomena. Consider an office environment where the user has a virtual notepad. The merged environment would not be convincing, if when the notepad is placed on the real table, it apparently falls through the table. Similarly, in the previous example of a home-owner decorating an empty house, the home-owner might desire the addition of a sliding door or venetian blinds. These virtual additions should properly interact with the surrounding (real) house.

VROC

The VROC (Virtual and Real Object Collisions) system uses computational power readily available today, for modeling interactive collision detection and collision response of moderately complex environments containing both virtual objects and real objects. A 3D mouse (or hand-held tracker) is provided with which the user can grab and control the linear and angular velocities of

the virtual objects. The system constantly performs collision detection and computes a classical mechanics-based collision response to model the interaction (e.g., collisions) between virtual and real objects, as well as the interactions among the virtual objects themselves. Figure 2 presents an outline of the entire VROC system.

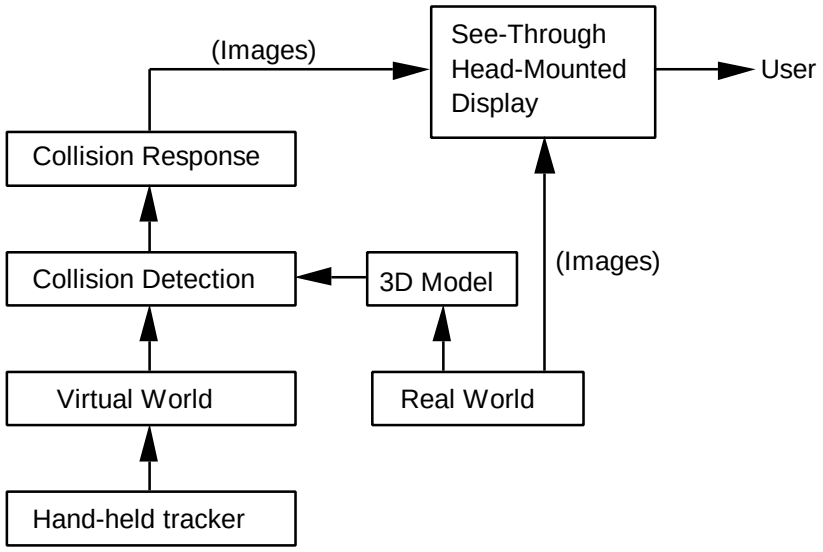


Figure 2: Schematic of entire VROC System. The user employs a 3D mouse (or hand-held tracker) to control the initial position and velocities of the virtual world objects. Collision detection and collision response is performed in the merged virtual and real world environment. Virtual imagery is combined with real world imagery in the see-through head-mounted display and presented to the user.

Collisions

In order to maintain sufficient realism, we need to maintain highly interactive performance (12 frames per second is the minimum acceptable frame rate). Thus we need a very fast collision detection method. The collision detection method will determine which objects are intersecting and how they are intersecting. This information is then passed on to the collision response method which alters the trajectory of the objects according to the laws of classical mechanics.

Over the last decade, multiple approaches have been developed for collision detection and collision response. No one collision detection or collision response algorithm can be said to be optimal. VROC uses a fast collision detection algorithm for 3D polygonal (convex) objects. The algorithm is based on Lin and Canny's [1991,1992] work (since the implementation of the VROC system, the Lin and Canny collision detection algorithm has been further enhanced and used in larger scale environments [Cohen95]). A 3D polygonal model approximates an object as a collection of planar patches, typically triangles, which together form a 3D object. The algorithm provides efficient collision detection by assuming that an object's position and orientation will not drastically change from one frame to another (interframe coherency). The algorithm maintains a list of the closest features between all pairs of polygonal objects (see figure 3). A feature corresponds to either a face (2D polygonal patch) or to one of its edges or vertices. When the distance between the features is less than a minimum tolerance value, the objects are considered to have collided.

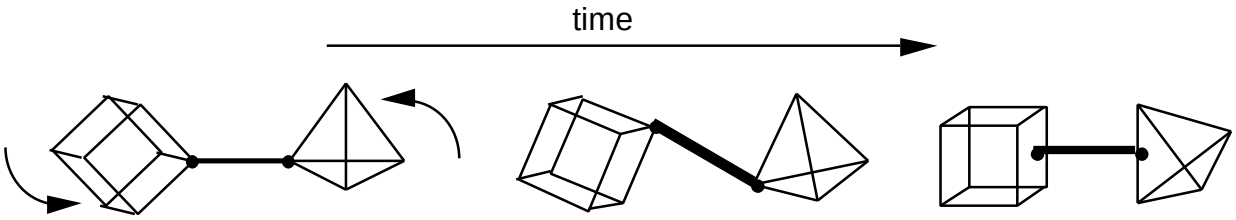


Figure 3: Closest features during 3 contiguous frames (time advances from left to right). Initially, the closest features are 2 vertices. As the objects rotate, the closest features become a face of the cube and an edge of the tetrahedron.

Once two objects have collided, a response needs to be computed. The VROC system assumes that all objects are rigid and have nearly inelastic properties. Furthermore, only single point contact between a pair of objects is modeled (since all objects used are convex, this is generally the case) [Moore88]. These assumptions simplify the collision response computation.

Based on the conservation of linear and angular momentum, the new velocities can be easily obtained:

$$m_1\bar{v}_1 = m_1v_1 + R \qquad I_1\bar{w}_1 = I_1w_1 + p_1 \times R$$
$$m_2\bar{v}_2 = m_2v_2 - R \qquad I_2\bar{w}_2 = I_2w_2 - p_2 \times R$$

The variables m, I, v, w describe each object's mass, inertia tensor matrix, linear velocity and angular velocity. The p vector is the relative vector from each object's center of mass to the point of contact (see figure 4). R is the impulse transfer vector (computed by inverting the 15x15 matrix formed using the above equations). Each object has its own elasticity coefficient. In order to simulate (slightly) elastic collisions, the computed R vector is scaled by the minimum of the 2 elasticity coefficients.

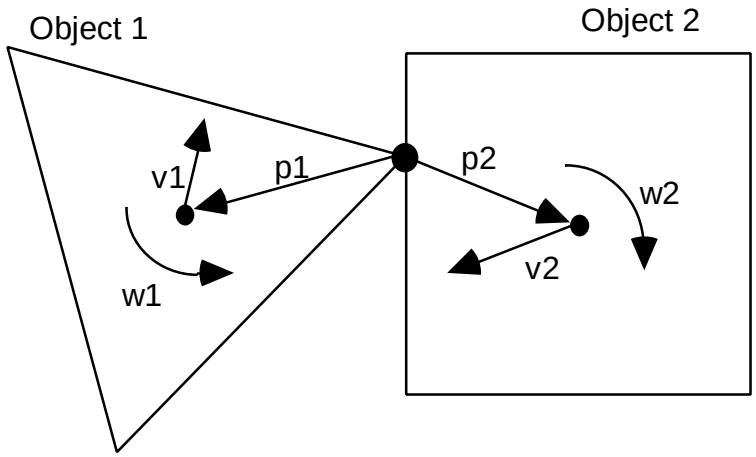


Figure 4: Two objects colliding. Arrows show linear velocity v, angular velocity w and vector p from collision point to each object's center of mass.

The collision detection and collision response algorithms are combined to form a dynamics simulation [Baraff89][Barraff92]. This implies that all the computations must be parametrized by time. The user must specify the time step to use to go from one frame to the next frame. The main problem with key-frame collision detection is that objects with large velocities might penetrate or pass through each other in one frame transition. Given the maximum linear velocity

and a collision distance (largest distance at which two objects are considered to have collided [Lin91]), the frame time step can be divided into internal time steps such that the internal time steps are small enough to guarantee that no two objects will totally pass through each other. Furthermore, the exact time of collision can be found by recursively subdividing the internal time step (binary search).

Optimizations

Since objects have continuous motion, it is possible to construct a sorted list of possible collision times [Lin92]. Given the distance between two objects, the bounds on the maximum linear velocity and linear acceleration, it is possible to predict the earliest time at which an object pair could collide. Since, in addition, objects have angular velocities, a safe prediction requires the difference between the radius of the inscribed sphere and the radius of the circumscribing sphere of each object to be subtracted from the inter-object distance.

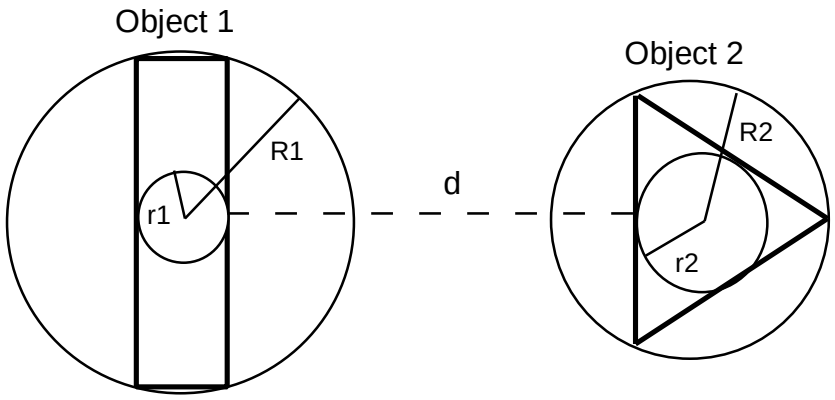


Figure 5: Inter-object distance, d, is reduced by $(R_2-r_2) + (R_1-r_1)$ to ensure that the collision prediction will never be incorrect due to angular rotation.

Collision checking must be performed on all object pairs because it is not known which objects will collide in an environment. This gives a maximum of n^2 collision pairs, where n is the number of objects. Fortunately, in the environments that the VROC system simulates, many of the objects (virtual or real) are not expected to move (e.g., real world tables, monitors, etc.).

These objects are considered *static* and no collision checking is done between two static objects. For example, if an environment uses 100 static objects to construct a desktop and only one moving (*dynamic*) object, then only 100 object pairs are checked as opposed to the more than 10,000 pairs that would have to be checked otherwise.

(a) (b)

Figure 6: This example shows a simple merged virtual and real environment. The figures are 2 frames from a VROC video sequence where the user grabs a virtual ball and bounces it on a real staircase. Here we see a real miniature staircase with virtual balls superimposed over the staircase. The wireframe lines represent where the system expects the edges of the staircase to be. The user employs a 3D mouse to select one of the balls (a), pick it up and throw in on top of the stairs (b) (in the full video sequence, the ball bounces on the staircase and collides with the other balls). The video sequence was recorded by placing a small camera in front of the user's left eye.

Implementation

The see-through head-mounted display used by the VROC system was developed by the head-mounted display research group at the University of North Carolina at Chapel Hill Computer Science Department in the spring of 1992. The head-mounted display was built (with off-the-shelf components) to gain experience for a wide field-of-view model with custom optics and CRT.

It is an optical see-through head-mounted display which uses a pair of 2-inch LCD displays that project the computer generated image onto a pair of half-silvered mirrors (see figure 7). The user perceives a combined image of the real world and the computer generated world. In the background of figure 1 is a monitor displaying the computer generated image which the user sees superimposed on the real staircase.

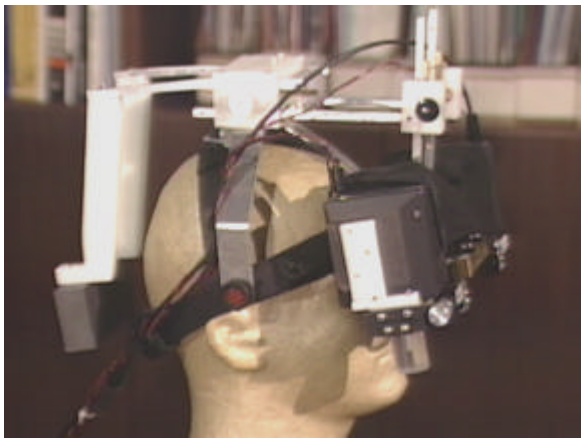


Figure 7: See-through Head-Mounted Display. The images from a pair of 2-inch color LCD displays is projected onto a pair of half-silvered mirrors placed in front of the user's eyes. The user perceives virtual imagery superimposed on real world objects.

The see-through head-mounted display is connected to Pixel-Planes 5 [Fuchs89]. Pixel-Planes 5 is a high-performance, scalable multi-computer for 3D graphics developed by the Computer Science Department of the University of North Carolina at Chapel Hill. Pixel-Planes 5 has a front-end of 10-40 Intel i860 processors. These general-purpose processors are programmed by the user to perform application computations and interact with the fast rendering hardware.

A portion of the VROC system runs on each processor. Recall that the collision detection scheme potentially requires a check to be performed between all possible object pairs. These checks are performed in parallel. Furthermore, each processor will compute the collision response for the object pairs it stores. The system is capable of achieving frame rates of up to 30

stereo frames per second. The multiple merged environments created with VROC range, on the average, from 14 to 28 stereo frames per second.

The set of object pairs that have to be checked for collisions is constructed based on the static model of the real world and on the set of virtual objects that "co-exist" with the real objects. The number of object pairs is typically significantly less than n^2, where n is the number of objects in the merged environment. The object pairs are distributed in a round-robin fashion among the multiple processors. Each processor will construct its sorted list of possible collision times. Consequently, each processor will only have to instantiate a subset of the total number of objects. An object may reside on multiple processors, but few objects will exist on all processors.

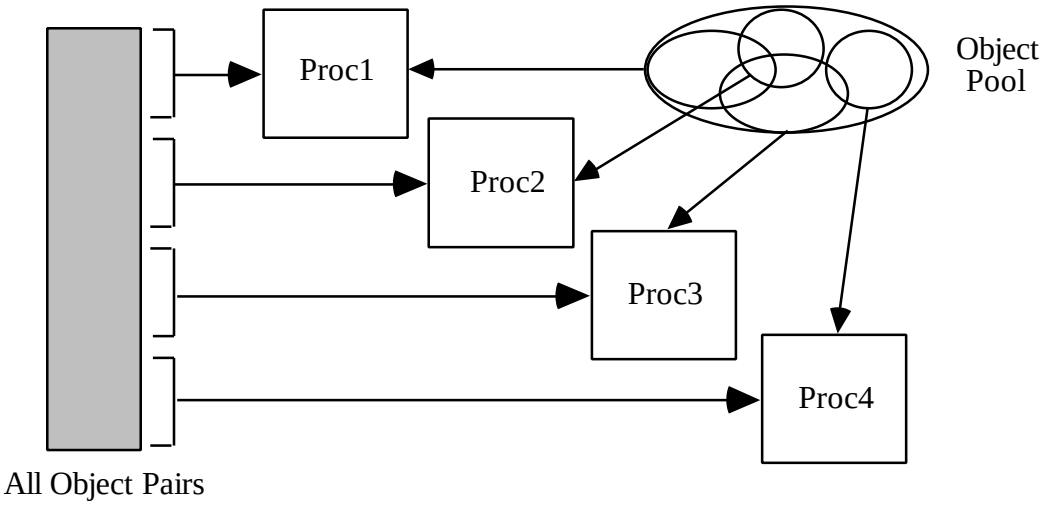
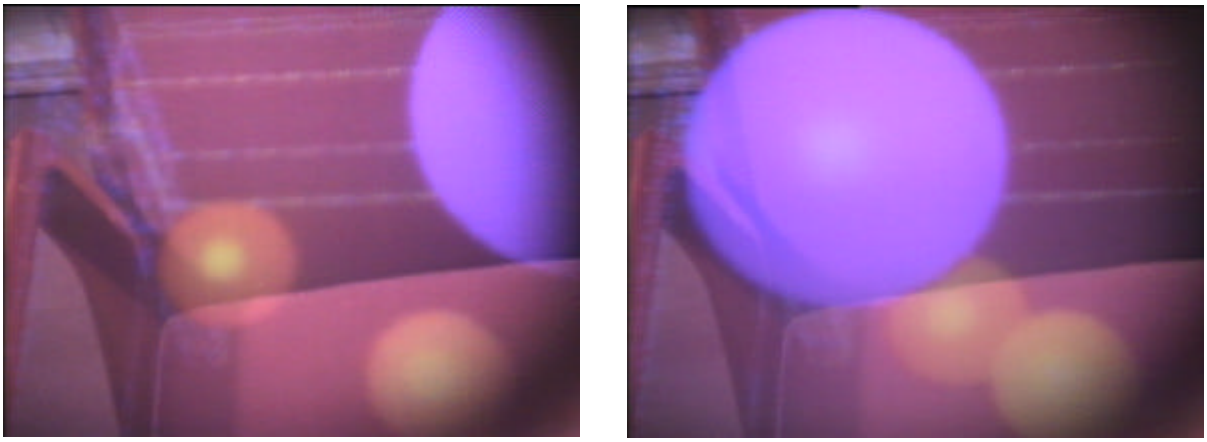


Figure 8: Object pair and object instantiation distribution. Each processor receives its partition of the total number of object pairs. Based on the object pairs it receives, each processor only needs to instantiate a subset of the objects from the object pool.

If an object pair is determined to be in collision, the associated processor will compute the collision response. Typically, each processor will only need to compute a single collision response. If other processors encounter a collision, they will compute their own collision response. Afterwards, processors that computed a collision response must broadcast the new velocities to all processors that have a copy of the objects involved in the collision. Hence, on average, the

collision response computations for multiple simultaneous collisions across the system take the same amount of time as one collision response (though some additional time is needed for the update messages sent between processors).



(a) (b)

Figure 9: This example shows a close-up view of a merged virtual and real environment. The user's hand is not visible in this sequence. The user has thrown the virtual blue ball from a distance towards the real chair. The ball bounces off the multiple sides of the stair, causing the balls resting on the chair bottom to disperse in various directions (a-b). In the video sequence, some of balls fall off the chair while others, including the blue ball, come to rest on the chair bottom.

Lessons Learned

Real World Models and Calibration

In order to create convincing interactions of the virtual and real world, the computer system must know exactly where the static real world objects lie; otherwise, visual anomalies will occur (for example, virtual objects might intersect or penetrate surrounding real objects). This requires the creation of a precise model of the real world. For complex environments, this is a time-consuming task. In addition to creating a geometrical model of the real world, the virtual

objects that interact with the real objects must be created with equally realistic colors and textures. New techniques are being explored for obtaining real world object models, such as 3D reconstruction from range images [Turk94], 3D scene analysis [Koch93], Plenoptics [Adelson91], among others.

Proper calibration of the static real objects and their computer generated counterparts depends not only on a properly measured model but also on:

- Compensation for the optical distortion generated by the see-through head-mounted display.
- The approximate perspective computations used for the virtual objects and computer models of real objects.
- The latency introduced by the trackers, refresh delay of the head-mounted display and the graphics pipeline. This latency (or lag) causes the virtual objects to apparently swim around the real world.

Even though the latency is only on the order of 60-150ms [Mine93] per frame, it is very noticeable especially with a see-through head-mounted display. Consequently, the illusion of virtual objects lying on real objects is less convincing. Studies have shown that people regularly turn their head at speeds above 300 degrees/second; in fact, fighter pilots turn their heads at speeds in excess of 2,000 degrees/second. Lets consider a user turning his head at a rate of 200 degrees/second [Bishop84] with a total latency of only 50 milliseconds; thus, the generated image will be off by approximately 10 degrees. A typical head-mounted display, which a field-of-view of 60 degrees, will display the image shifted to one side by one sixth the display resolution!

Solving the calibration problem would help to improve the apparent location of real objects from within the see-through head-mounted display, thus enabling precise collision responses and other feedback (sound, force, tactile, etc.). Furthermore, virtual objects could be accurately obscured (partially obscured or totally obscured) by the real objects in front of them. For example, Wloka [1995] constructed a system capable of resolving occlusion between virtual

and real world objects using a (video) see-through head-mounted display and depth inferred from stereo images.

So far we have assumed a static model of the real world. This limits the range of possible environments. It is not clear how to go about removing this assumption. Information about the movement of real objects could be obtained from a tracker placed in each of the moving objects. The KARMA system [Feiner93] places individual trackers on some of the dynamic real-world objects. Unfortunately, this scheme is subject to the inaccuracies of the trackers but it is a potential solution. Another approach is to use imaging technology and to continuously compute from a 2D camera view of the world, the position and orientation of the 3D objects contained in it [Tomasi92]. In any case, this is certainly a difficult problem.

Feedback

How real the interaction of the virtual and real world appears to the user is not totally dependent on the visual cues from the head-mounted display. Other sensory input, such as sound, is also needed [Astheimer93]. When two real world objects collide, a specific sound is produced dependent on the objects involved. Similarly, when a virtual and a real object collide, a different sound should be emitted. Stereo or 3D sound would improve the realism of the merged worlds. Current audio technology is advanced enough to produce such effects reasonably well.

A more important (and difficult to implement) type of feedback is force feedback. The user may have a large virtual object in his hand. It would be helpful if the user could feel when the virtual object's surface has collided with a real object. For example, an application which allows the user to place virtual furniture in an empty real room could give the user force feedback when the virtual sofa being placed hits a wall.

Force feedback is not only useful for virtual and real object interaction but also for user and virtual object interaction. The user may wish to touch and feel the contours of a virtual object. The illusion of realism would certainly be improved if the user could run his hand along the stairs of the environment of figure 6 and feel the presence of the virtual balls. Another

example is a sculpting environment. Tactile feedback is essential to provide an effective sculpting tool.

Conclusions

To summarize, the intent of VROC is to create an integrated system to model the interaction of virtual and real objects in a merged environment. A see-through head-mounted display system is combined with a graphics engine to present to the user the merged environment while performing collision detection and collision response computations. Hopefully the observations and lessons learned will benefit further research of such systems.

Acknowledgments

I would like to thank my professors, Henry Fuchs, Gary Bishop and Dinesh Manocha for their patience and wisdom. I would like to thank the many people who have helped me throughout this project; Anselmo Lastra for answering my questions regarding Pixel-Planes 5, Rich Holloway for assisting with the see-through head-mounted display hardware and Rik Faith, Mark Mine, Marc Olano, William Mark, Amitabh Varshney, Yunshan Zhu. Grants from the National Science Foundation's Supercomputing Center for Science and Technology (ASC-8920219) and Advanced Research Projects Agency (DAEA 18-90-C-0044, DABT 63-92-C-0048) made this work possible.

References

[Adelson91] Adelson E.H., Bergen J.R., "The Plenoptic Function and the Elements of Early Vision", Computational Models of Visual Processing, Chapter 1, Edited by Michael Landy and J. Anthony Movshon. The MIT Press, Cambridge, Mass. 1991.

[Astheimer93] Astheimer P., "What you See is What you Hear - Acoustics Applied in Virtual Worlds", *IEEE Symposium on Research Frontiers in Virtual Reality*, pp. 100-107, Oct. 25-26, San Jose, CA, 1993.

[Bajura92] Bajura M., Fuchs H., Ohbuchi R., "Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery within the Patient", *Computer Graphics (Proc. SIGGRAPH)*, vol. 26, no. 2, July 1992.

[Baraff89] Baraff D., "Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies", *Computer Graphics (Proc. SIGGRAPH)*, vol. 23, no. 3, pp. 223-232, July 1990.

[Baraff92] Baraff D., "Rigid Body Dynamics", *Computer Graphics Course Notes: An Introduction to Physically Based Modeling (Proc. SIGGRAPH)*, pp. H3-H29, 1992.

[Bishop84] Bishop G., "Self-Tracker: A smart Optical Sensor on Silicon", Ph.D. Dissertation, University of North Carolina at Chapel Hill, TR 84-002, 1984.

[Cohen95] Cohen J., Lin M., Manocha D., Ponamgi K., "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments", *Symposium on Interactive 3D Graphics (SIGGRAPH)*, Monterey, California, pp. 189-196, April 1995.

[Feiner93] Feiner S., MacIntyre B., Seligmann D., "A Knowledge-Based Augmented Reality", *CACM*, Vol. 36, No. 7, pp. 53-62, 1993.

[Fuchs89] Fuchs H., Poulton J., Eyles J., Greer T., Goldfeather J., Ellsworth D., Molnar S., Turk G., Tebbs B., Israel L., "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories", *Computer Graphics (Proc. SIGGRAPH)*, vol. 23, no. 3, pp. 79-88, July 1989.

[Koch93] Koch R., "Dynamic 3D Scene Analysis through Synthesis Feedback Control", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 6, pp. 556-568, 1991.

[Lin91] Lin M., Canny J., "A fast algorithm for incremental distance calculations", *IEEE International Conference Robotics and Automation*, pp. 1008-1014, 1991.

[Lin92] Lin M., Canny J., "Efficient Collision Detection for Animation", *Third Eurographics Workshop*, Cambridge, England, September 1992.

[Mine93] Mine M., "Characterization of End-to-End Delays in Head-Mounted Displays", University of North Carolina at Chapel Hill, TR 93-001, 1993.

[Moore88] Moore M., Wilhelms J., "Collision Detection and Response for Computer Animation", *Computer Graphics (Proc. SIGGRAPH)*, vol. 22, no. 4, pp. 289-297, August 1988.

[Tomasi92] Tomasi C., Kanade T., "Shape and Motion from Image Streams under Orthography: a Factorization Method", *Intl. Journal of Computer Graphics*, vol. 9, no. 2, pp. 137-154, 1992.

[Turk94] Turk G., Levoy M., "Zippered Polygon Meshes from Range Images", *Computer Graphics (Proc. SIGGRAPH)*, Annual Conference Series, pp. 311-318, July 1994.

[Wloka95] Wloka M., Anderson B., "Resolving Occlusion in Augmented Reality", *Symposium on Interactive 3D Graphics (SIGGRAPH)*, Monterey, California, pp. 5-12, April 1995.