Daniel G. Aliaga

Computing at the intersection of two realms.

Virtual Objects in the Real World

THE TERM *VIRTUAL REALITY* USUALLY REFERS TO SYSTEMS THAT generate a totally synthetic environment, in which 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, look up and down, and walk into rooms that don't physically exist. This capability allows users to experience environments that are not yet created (e.g., architectural walk-throughs) or position themselves in worlds they can never really visit (e.g., exploring a surface at an atomic level).

There are, however, situations in which 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 homeowner to decorate a real house, or children to design and build virtual toys that 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 woman's womb [3]. The KARMA sys-

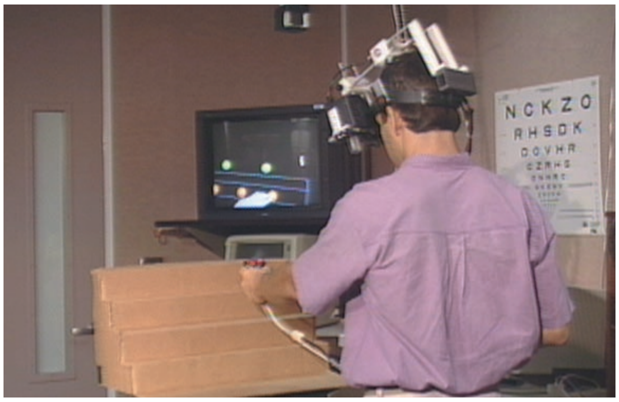
Figure 1. An augmented reality system. A see-through head-mounted display is connected to an image-generation system to present the user a world consisting of both virtual and real objects.

tem [5] 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.

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 at all convincing. 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 homeowner decorating an empty house, the homeowner might desire the addition of a sliding door or venetian blinds. These virtual additions should properly interact with the surrounding (real) house.

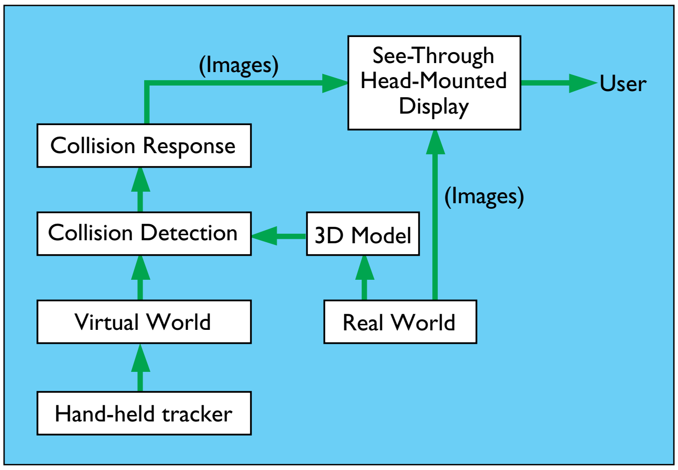

Figure 2. Schematic of entire VROC system. The user employs a 3D mouse (or hand-held tracker) to control the initial position and velocities of the virtual world objects. Collision detection and collision response is performed in the merged virtual and real-world environment. Virtual imagery is combined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

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 so 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.

Collisions

In order to maintain sufficient realism, 12 frames per second is the minimum acceptable frame rate. Thus a very fast collision detection method is necessary. The collision detection method will determine which objects intersect and how they intersect. 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 single collision detection or collision response algorithm can be said to be ideal. VROC uses a fast collision detection algorithm for 3D polygonal (convex) objects. The algorithm is based on Lin and Canny's [9] 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 [4]). 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. 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.

Once two objects have collided, a response must be computed. The VROC system assumes that all objects are rigid and have nearly inelastic properties. Furthermore, only a single point of contact between a pair of objects is modeled (since all objects used are convex, this is generally the case) [10]. These assumptions simplify the collision response computation. Based on the conservation of linear and angular momentum, the new velocities can be easily obtained:

$$m_1 \bar{v}_1 = m_1 v_1 + R \qquad I_1 \bar{w}_1 = I_1 w_1 + p_1 \times R$$
$$m_2 \bar{v}_2 = m_2 v_2 - R \qquad I_2 \bar{w}_2 = I_2 w_2 + p_2 \times R$$

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. R is the impulse transfer vector (computed by inverting the 15 x 15 matrix formed using these 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.

The collision detection and collision response algorithms are combined to form a dynamics simulation. 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 [8]), 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 [9]. 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.

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 simulated by the VROC system simulates, many of the objects (virtual or real) are not expected to move (e.g., real-world tables or monitors). 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. The images in Figure 3 depict a merged virtual and real environment.

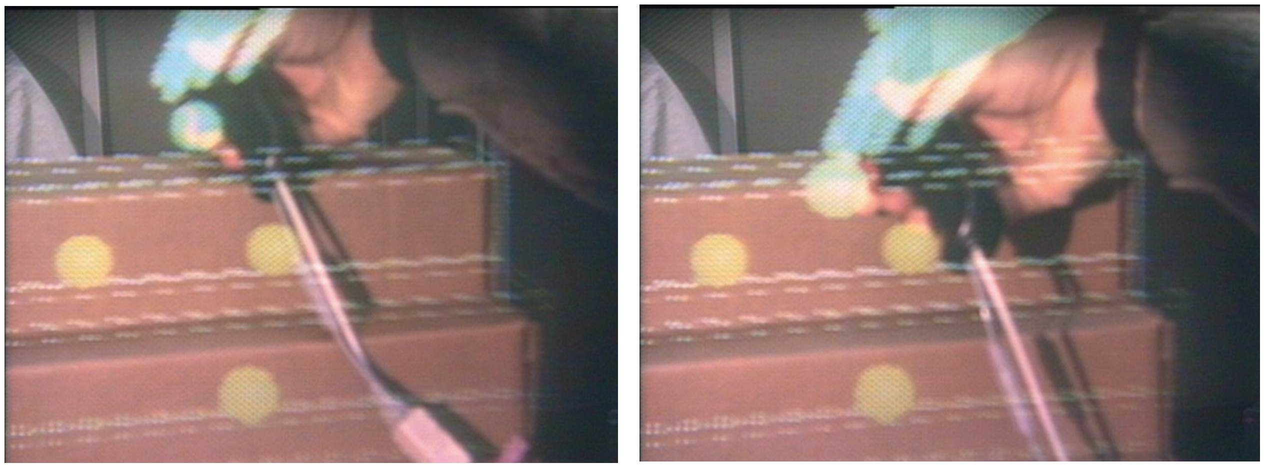
Figure 3 (a) (b). This example shows a simple merged virtual and real environment. The figures are two frames from a VROC video sequence where the user grabs a virtual ball and bounces it on a real staircase. Here we see a real miniature staircase with virtual balls superimposed over the staircase. The 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 that uses a pair of 2-inch LCD displays to project the computer-generated image onto a pair of half-silvered mirrors (see Figure 4). 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 that the user sees superimposed on the real staircase.

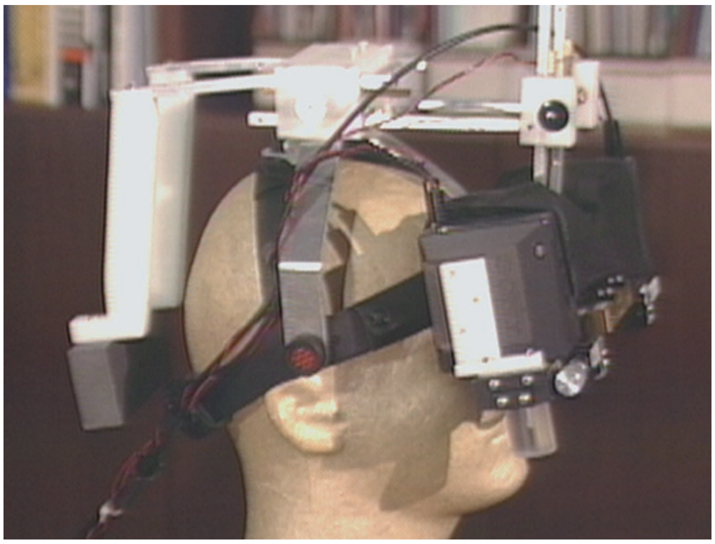
Figure 4. See-through head-mounted display. The images from a pair of two-inch color LCD displays are projected onto a pair of half-silvered mirrors placed in front of the user's eyes. The user perceives virtual imagery superimposed on real-world objects.

The see-through head-mounted display is connected to Pixel-Planes 5 [6]. Pixel-Planes 5 is a high-performance, scalable multi-computer for 3D graphics developed by the Chapel Hill Computer Science Department. 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 "coexist" 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.

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 responses. Afterward, 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).

Real-World Models and Calibration

In order to create convincing interactions between the virtual and real worlds, 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 (see Figure 5). 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 [11], 3D scene analysis [7], Plenoptics [1], among others.

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.

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 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. Let's consider a user turning his or her head at a rate of 200 degrees/second with a total latency of only 50 milliseconds; thus the generated image will be off by approximately 10 degrees. A typical head-mounted display, with 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 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 or totally) by the real objects in front of them. For example, Wloka [12] 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. Removing this assumption is difficult. Information about the movement of real objects could be

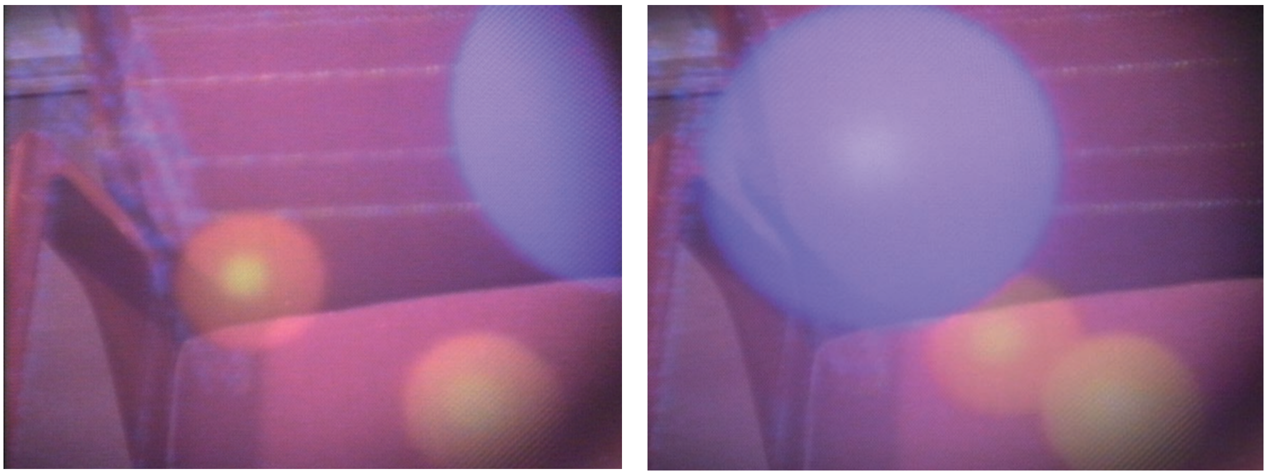

Figure 5 (a) (b). This example shows a close-up view of a merged virtual and real environment. The user's hand is not visible in this sequence. The user has thrown the virtual blue ball from a distance toward the real chair. The ball bounces off the multiple sides of the chair, causing the balls resting 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.

obtained from a tracker placed in each of the moving objects. The KARMA system [5] 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 would be 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. In any case, this is certainly a difficult problem.

Feedback

How real the interaction of the virtual and real worlds 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 [2]. 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 that allows the user to place virtual furniture in an empty real room could give the user force feedback when the virtual furniture 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 users could run their hands along the stairs of the environment of Figure 3 and feel the presence of the virtual balls. Another example is a sculpting environment—tactile feedback is essential to provide an effective sculpting tool.

Conclusion

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 the merged environment to the user while performing collision detection and collision response computations. The observations and lessons learned should benefit further research of such systems. **C**

Acknowledgments

I would like to thank 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 conferring with me 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, and Yunshan Zhu.

References

1. Adelson, E.H. and Bergen, J.R. The Plenoptic function and the elements of early vision. *Computational Models of Visual Processing*, M. Landy and J.A. Movshon, Eds. The MIT Press, Cambridge, Mass., 1991.
2. Astheimer, P. What you see is what you hear—acoustics applied in virtual worlds. In *Proceedings of the IEEE Symposium on Research Frontiers in Virtual Reality*, (San Jose, Calif., Oct. 25–26, 1993), pp. 100–107.
3. Bajura, M., Fuchs, H., and Ohbuchi R. Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. *Computer Graphics (Proceedings of SIGGRAPH) 26*, 2 (July 1992).
4. Cohen, J., Lin, M., Manocha, D. et al. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the Symposium on Interactive 3D Graphics (SIGGRAPH)*, (Monterey, California, Apr. 1995), pp. 189–196.
5. Feiner, S., MacIntyre, B., and Seligmann, D. A knowledge-based augmented reality. *Commun. ACM 36*, 7 (July 1993), 53–62.
6. Fuchs, H., Poulton, J., Eyles, J. et al. Pixel-Planes 5: A heterogeneous multiprocessor graphics system using processor-enhanced memories. *Computer Graphics (Proceedings of SIGGRAPH) 23*, 3 (July 1989), pp. 79–88.
7. Koch, R. Dynamic 3D scene analysis through synthesis feedback control. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 6 (1991), 556–568.
8. Lin, M. and Canny, J. A fast algorithm for incremental distance calculations. In *Proceedings of the IEEE International Conference on Robotics and Automation* (1991), pp. 1008–1014.
9. Lin, M. and Canny, J. Efficient collision detection for animation. In *Proceedings of the Third Eurographics Workshop* (Cambridge, England, Sept. 1992).
10. Moore, M. and Wilhelms, J. Collision detection and response for computer animation. *Computer Graphics (Proceedings of SIGGRAPH) 22*, 4 (Aug. 1988), 289–297.
11. Turk, G. and Levoy, M. Zippered polygon meshes from range images. *Computer Graphics (Proceedings of SIGGRAPH Annual Conference Series,* July 1994*)*, pp. 311–318.
12. Wloka, M. and Anderson, B. Resolving occlusion in augmented reality. In *Proceedings of the Symposium on Interactive 3D Graphics (SIGGRAPH)*, (Monterey, California, Apr. 1995), pp. 5–12.

Daniel Aliaga (aliaga@cs.unc.edu) is a Ph.D. candidate and research assistant in the Department of Computer Science at the University of North Carolina at Chapel Hill.

This work was made possible by 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).