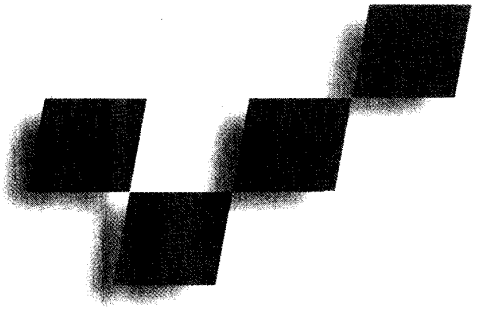
Dynamic Registration Correction in Video-Based Augmented Reality Systems



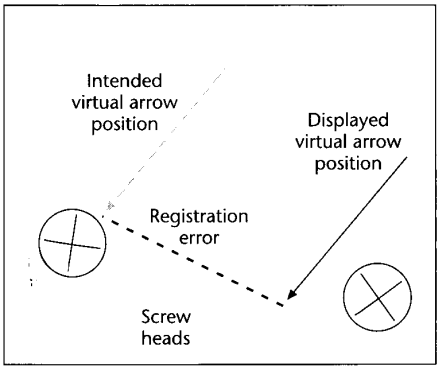
Michael Bajura
University of North Carolina at Chapel Hill

Ulrich Neumann
University of Southern California

AR systems must register images accurately to convey the perception that real and computer-generated objects occupy the same 3D space. Dynamic measurement of 2D misregistration helps reduce 3D registration errors.

Augmented reality (AR) systems allow users to interact with real and computer-generated objects by displaying 3D virtual objects registered in a user's natural environment. Applications of this powerful visualization tool include previewing proposed buildings in their natural settings, interacting with complex machinery for purposes of construction or maintenance training,[1-3] and visualizing in-patient medical data such as ultrasound.[4]

In all these applications, computer-generated objects must be visually registered with respect to real-world objects in every image the user sees. If the application does not maintain accurate registration, the computer-generated objects appear to float around in the user's natural environment without having a specific 3D spatial position. Figure 1 illustrates the registration problem in a case that uses a virtual arrow to indicate a particular screw of a mechanical assembly. The arrow is displayed near the lower right screw, but the application intended the arrow to indicate the left screw. Registration error is the observed displacement in the image between the actual and intended positions of virtual objects.



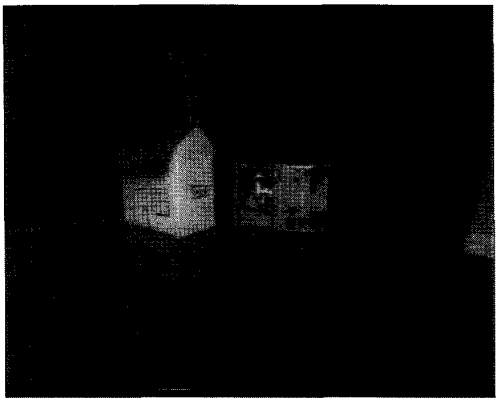
1 Registration error between intended and displayed virtual arrow positions.

Figure 2 is an image from our experimental AR system, which dynamically corrects image registration on a frame-by-frame basis. The figure shows a computer-generated television antenna registered correctly on a toy house and a direction arrow registered correctly on a disk drive screw. The antenna and arrow maintain correct registration in every image the user sees, even when the user moves. This article explains how we achieved this result and suggests future directions for dynamic registration correction.

Current model for AR systems

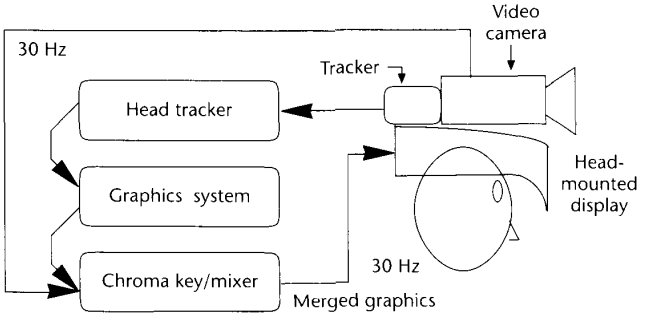
AR systems combine images of collocated real and virtual environments. In an optical see-through AR system, a user views the real world through an optical merging system that blends the virtual images into the real scene. A video see-through AR system presents the real world through head-mounted video camera images into which the virtual images are inserted electronically. Figure 3 shows a user wearing a head-mounted display (HMD) that presents combined images of both real and virtual

0272-17-16/95/$4.00 © 1995 IEEE



2 Experimental augmented reality system showing dynamic registration of a virtual antenna and annotation arrow that appear to be "nailed" in place.

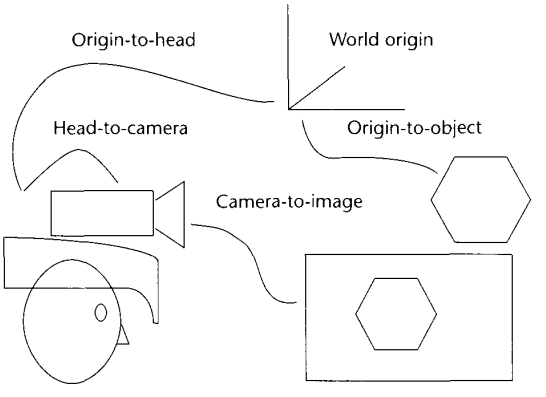
(computer-generated) objects. Images of real objects are obtained from a video camera mounted on the display helmet. Images of virtual objects are generated by a graphics system. A tracking system reports the user's head position to the graphics system so it can render images of the virtual world. The real and virtual images are typically merged with a chromakey or video mixer for display in the HMD.



3 Typical video-based augmented reality system components.

For clarity in this article, we discuss only the monocular case with one video camera mounted on the HMD. A stereo system adds a second video camera and treats it as a second independent monocular system. Edwards, Rolland, and Keller[5] address the problem of constructing a binocular HMD that presents correct stereopsis.

In a monocular system, the perceived registration between real and virtual objects depends on how accurately the virtual camera models the real one. Figure 4 shows a detailed transformation model for the virtual camera. An origin-to-object transformation specifies the position and orientation, or *pose*, of virtual objects relative to a coordinate system origin. Two transformations, origin-to-head and head-to-camera, determine the position of the virtual camera. The origin-to-head transformation, reported by the tracking system, specifies the location of a tracking element's position on the user's HMD. The head-to-camera transformation sets the effective viewpoint of the real camera relative to the tracking element. Virtual camera images are produced by a perspective projection onto a virtual image plane. A nonlinear camera-to-image mapping models the field of view and lens distortion of the real camera.

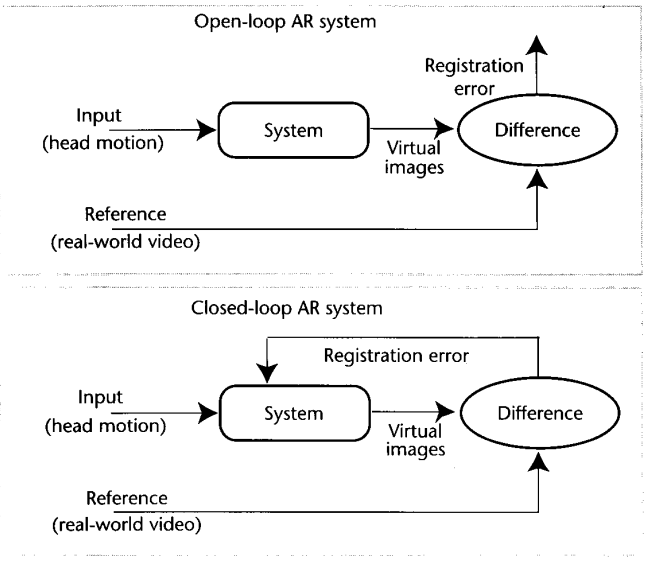


4 Transformations from object to image.

The head-to-camera and camera-to-image transformations are typically determined by a calibration procedure such as the one described below under "Calibration." Note that this model does not address the correction for distortion in the HMD optics, which is a separate problem from generating correctly registered images.[5]

There are four causes of registration errors in combined real and virtual images:

1. *The tracking system's origin is not aligned with the world coordinate system origin.* This error causes all virtual objects to appear displaced from their proper positions.
2. *The virtual origin-to-object transformation is not the*



5 Open- and closed-loop AR systems: Error is fed back to the system for correction in a closed-loop system.

same as the real origin-to-object transformation for a particular object. This error causes individual objects to appear out of position.

3. *The virtual camera position is not the same as the real camera position*. This can be caused by errors in either the static head-to-camera transformation or the dynamic origin-to-head transformation reported by the tracking system. The tracking system exhibits two error types: temporal and pose. Pose errors cause misregistration in all cases, while temporal errors cause misregistration only during user movement. Temporal errors are caused by delays in the sensing and reporting of tracking information to the computer graphics system and in the computer graphics system's generation of appropriate virtual images.[6]
4. *The virtual camera-to-image mapping doesn't accurately model the real camera*. The camera-to-image mapping abstraction asserts that any real camera can be modeled by an idealized pinhole camera with a particular center of projection, viewing direction, field of view, and distortion function. The distortion function is a 2D warp that accounts for the nonlinearities found in lens-based projection systems. Errors in the camera-to-image mapping cause misregistration to vary with screen position.

Correcting registration error

The new idea presented here is to dynamically measure the registration error, or misregistration, in each combined image and to use this information to correct the system errors that caused the misregistration. In the model in Figure 3, correct image registration depends on absolute correctness of all the transformations shown in Figure 4. This is analogous to an open-loop system, like the one shown in Figure 5, whose output is perceived to have error. The only way to improve such systems is to make each component more accurate. Alternatively, a closed-loop system, also shown in Figure 5, senses its own output and corrects errors dynamically. A closed-loop AR system can tolerate transformation inaccuracies because it senses misregistration and applies corrections to the transformations. The type of correction performed depends on two factors: (1) the method used for detecting and measuring image misregistration and (2) the uncertainty and image-space sensitivity of the various transformation parameters to be adjusted. Both are described below.

The method of measuring image misregistration dictates what corrections can be performed. One method identifies a recognizable point on each object to be registered. The image coordinates of each point are then located in both the real and uncorrected virtual images. The difference between each point's position in each real image and corresponding uncorrected virtual image is the registration error, or misregistration, for the object corresponding with that point. This measure of misregistration can correct for errors such as camera orientation and sometimes camera position. A drawback, however, is that it cannot estimate either the distance between an object and the camera or an object's orientation.

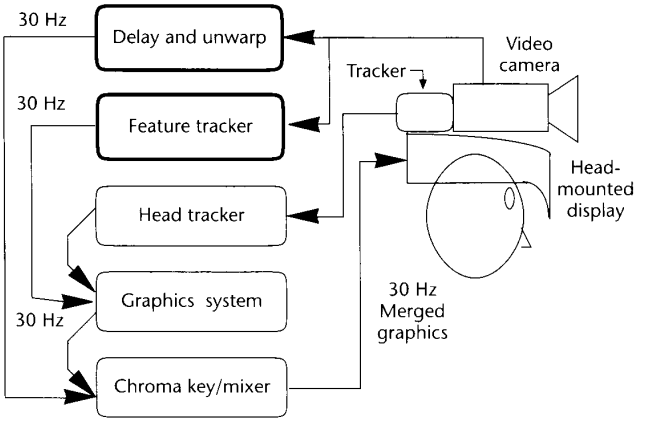
Correction based on a single point is underconstrained. Even so, misregistration can still be reduced. In many cases, parameters that cannot be estimated can be assumed to be correct, and you can reduce misregistration by adjusting the remaining parameters. In other cases, there is no way to separate the error contributions from different parameters, but you can adjust one or more depending on their relative uncertainty. The important point is that image registration error can be reduced even when some approximations are made. The application determines how appropriate and useful the corrected results are.

The selection of which transformation parameters to adjust depends on both their uncertainty and the sensitivity of image-space errors to that uncertainty. For example, if the positions of objects are well known but the camera pose is relatively uncertain, the camera pose should be adjusted instead of the object positions. The orientation component of camera pose is better determined by objects relatively far from the camera, while the positional component of camera pose is better determined by objects nearer to the camera.

A closed-loop AR system can also be used to correct temporal errors. In video-based AR systems, the real video image stream can be delayed to match the virtual image path latency, which is the time it takes to measure misregistration and generate a corrected virtual image. With video delay, it is possible to obtain exact temporal and spatial image registration at every recognized point in every image the AR user sees. If regis-

tration error occurs, it is only because the error compensation algorithm failed. For applications in which users can tolerate the miscue introduced by minimal video delays, you can achieve potentially perfect registration. This trade-off is not possible with optically based AR systems, which let users see their surroundings directly.

Some success at improving registration error has been achieved with autocalibration approaches[7] and predictive tracking techniques[8,9] that use a state estimate to help predict current measurements. However, these approaches still suffer from the open-loop requirement for perfect tracking and calibration.



6 Experimental video-based augmented reality system with registration correction.

The experimental system

Our experimental closed-loop AR system corrects image registration error on a frame-by-frame basis. The following sections describe its functional components and their hardware implementation, system calibration, the method of correcting registration error, and the results of operating the system both with and without dynamic registration correction.

System components

Figure 6 is a schematic for the experimental AR system. It is similar to the system described in Figure 3 except for the addition of a real-time video delay and unwarp pipeline and a real-time image-feature tracker. The delay and unwarp pipeline delays video by a constant number of frames and optionally applies an inverse distortion function that converts the incoming signal into an equivalent pinhole camera image. With our hardware, it is more practical to undistort the real camera video images to match the undistorted virtual images than to distort the virtual images to match the real camera images. The pipeline delay is adjustable but constant during operation, and set to match the delay in generating the correct virtual image to mix with the corresponding real camera video frame.

The image-feature tracker recognizes features in the real video stream and passes their image coordinates to the graphics system. The detected features are red LEDs placed in the environment and driven by a 9V power supply. The LEDs are significantly brighter than other objects in the environment, and the system detects them by applying a brightness and image-area threshold to each image.

Correspondence between LEDs and the particular features they represent is established by matching detected LED positions with the nearest estimated feature positions in each corresponding uncorrected virtual image. Establishing correspondences does not require rendering uncorrected virtual images. Only the feature positions must be computed. Once feature correspondence is established, the difference between each feature's position in each real image and its estimated position in each corresponding virtual image can be used to render virtual images that are better registered with the real video images. (The methods are explained below under "Correcting registration error.")

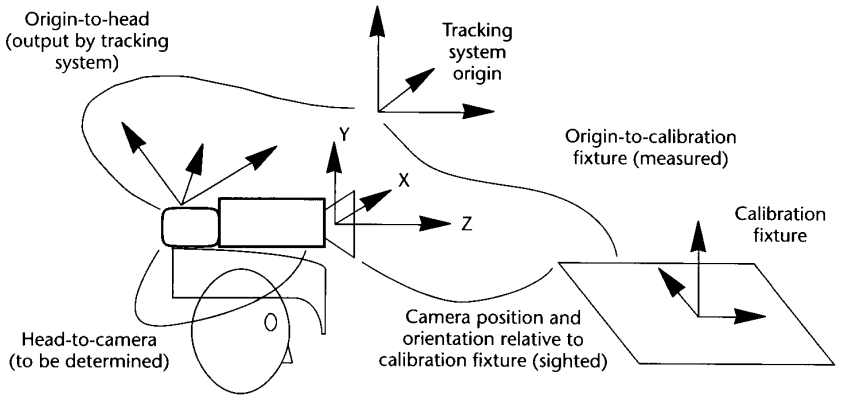
The camera used this experiment is a Panasonic GP-KS102 color charge-coupled device (CCD) camera with a highly distorting 110 degree wide-angle lens. The head tracking system is a "A Flock of Birds" by Ascension. The video delay and unwarp, image-feature tracker, and graphics system are different software modules that use separate portions of the Pixel-Planes 5 graphics multicomputer at the University of North Carolina at Chapel Hill.[10] Video is input to the Pixel-Planes 5 system via a real-time video digitizer and output via a standard double-buffered frame buffer. The real and virtual images are merged with an analog Sony CRK-2000 Universal Chroma Keyer video mixer.

The AR world of the experimental system consists of a virtual TV antenna positioned atop a real model house (where an LED is located) and a virtual arrow that indicates an adjustment screw on a real disk drive (where a second LED is located).

Calibration

Before the system can be operated, the head-to-camera transformation and the camera-to-image mapping must be estimated. These are static transformations, which, in theory, can be measured exactly. However, practical limits of measurement accuracy often preclude accurate calculation of the true transformations. Our method achieves approximate results by operating the AR system and using manual feedback to converge on a solution.

Figure 7 (next page) shows how we determine the head-to-camera transformation. A calibration fixture is used to represent a fixed pose. It is measured relative to the tracking system origin. When the camera is placed in a specific pose relative to the calibration fixture, the system records the pose of the head-tracking element. The head-to-camera transformation is the difference between the head tracking element's pose and the camera's pose. The experimental system requires a



7 Calibration transformations.

calibration fixture because the tracking system reports positions relative to a fixed but not precisely known origin. If the tracking system reported coordinates in a known coordinate system relative to itself, a calibration fixture would not be necessary.

We described the calibration procedure in more detail in the VRAIS version of this article.[11] (Other methods are described in Janin, Mizell, and Caudell.[12]) Rather than repeat the procedure here, we instead detail the compensation for nonlinear lens distortion in the camera-to-image mapping.

We model the camera-to-image mapping by an ideal projection and a nonlinear lens distortion function. The nonprojective distortion in our wide field-of-view lens (110 degrees) is severe and therefore requires correction. Such correction may be omitted when using narrow field-of-view lenses, which typically have less nonprojective distortion.

Lens distortion is measured by examining a distorted camera image and finding a 2D warp function that converts the image into projective one. The method appeals to a basic rule of (linear) projective geometry: Straight lines remain straight under projection. Scales may change and parallel lines may intersect, but the image of a straight line is always straight. If there is a mapping that converts images from a distorting camera into images where all straight lines appear to be straight, then the distorting camera can be modeled by a composition of this mapping and a pinhole camera model.

Figure 8 presents images of a test pattern imaged with the 110 degree wide-angle camera lens. Figure 8a is the distorted image output from the camera. Figure 8b is a corrected version of the same image. The correction is a radial distortion at the image center, which accounts for most of the image distortion.[13] The steps of the algorithm are

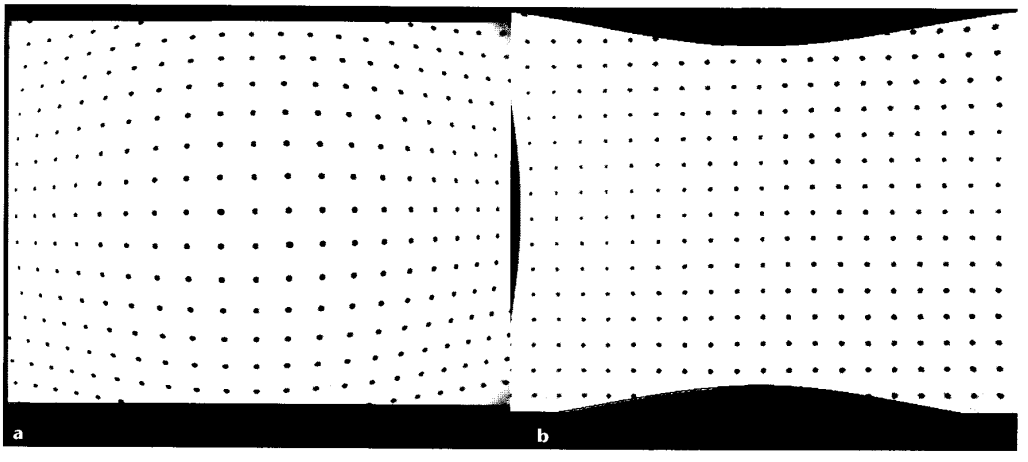
1. Obtain an image of a test grid (a regular 2D grid of dots) with the camera perpendicular to the grid plane.
2. Locate the centers of the dots in the distorted image. A fixed threshold segments the dots from the background. Connected pixels that segment as dots are identified and their weighted centroid is computed:

$$\text{Center} = \frac{\sum_{i=1}^{N} P(i) \cdot V(i)}{\sum_{i=1}^{N} V(i)} \quad \textbf{(1)}$$

where N is the number of pixels in the dot, $P(i)$ is the coordinates of the ith pixel, $V(i)$ is the value of the ith pixel.

The perspective projection and lens distortion both introduce error into the centroid calculation. However, in our case, these errors are small because the test grid plane is almost perpendicular to the camera view direction and the distortion over the area of any single dot is small.

3. Starting from the centermost column of dots in the



8 Test pattern with wide-angle camera lens: (a) distorted image and (b) corrected image.

image, search outward horizontally for the nearest neighboring dots to form rows of dots that lie in a straight line in the test grid. We implemented an automated procedure, but you could identify the dots forming a line manually as well. Regardless of the method, the process is repeated for vertical columns so that all dots in the image are identified as part of horizontal or vertical lines in the test grid.

4. An iterative method computes the coefficients of the distortion function by minimizing an image error function. The distortion function is

$$R = r + ar^2 + br^3 + cr^4 + dr^5 \quad \textbf{(2)}$$

where r is distorted radial distance; R is undistorted radial distance; and a, b, c, and d are coefficients to be computed.

The image error (Ie) is the total error of all the lines after the distortion function is applied:

$$Ie = \sum_{i=1}^{L} Le(i) \quad \textbf{(3)}$$

where L is the number of lines and $Le(i)$ is the error of ith line.

A line's error is the sum of the squares of each dot's distance to the least-squares line that it lies on:

$$Le(i) = \sum_{j=1}^{U} (D(j,i))^2 \quad \textbf{(4)}$$

where U is the number of dots on ith line and $D(j, i)$ is the distance from dot j to least-squares line i.

The iteration begins with $a = b = c = d = 0$. We assume that $|a| > |b| > |c| > |d|$. First, only a is varied and a minimum is found. Next, a and b are varied to find another minimum solution. Then a, b, and c are varied, and so on. The distortion function is computed for a set of candidate image center pixels. The lowest residual image error (Ie) identifies the pixel closest to the optical axis. The final corrected image is scaled to fit the desired image size.

This method provides a stable solution for any image of the test grid. However, the results differ for different images taken at varying distances or positions. The method is sensitive to the distribution of dots in the image, since it finds a best-fit solution for given dot positions—not the entire screen. In spite of this, the result is a good approximation and the closed-loop system approach can correct for the residual error from this source as well as the others.

Correcting registration error

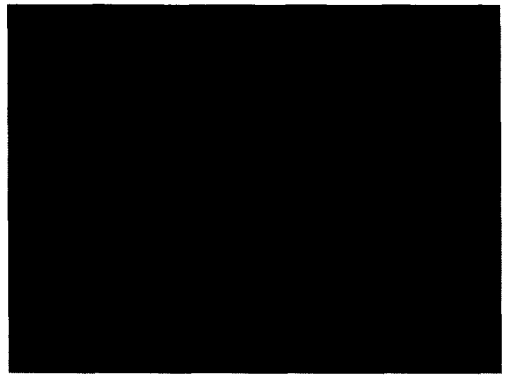
The image registration model of matching a point on each object makes it difficult to determine which particular errors are causing misregistration. One way to think about this is to consider the misregistration as a function of the camera pose error (a composition of errors in the origin-to-head and head-to-camera transformations), camera-to-image mapping error, and origin-to-object transformation error.

Misregistration can be reduced by modifying one or more of the parameters that might be causing it. We studied two approaches to reducing registration error in this experiment. One approach assumes that the camera pose is absolutely correct and that misregistration is due to errors in the camera-to-image mapping and origin-to-object transformation. The second approach assumes that the camera-to-image mapping and origin-to-object transformations are correct and that the camera pose is in error. Neither of these approaches is optimal in the sense of minimizing error by smoothly adjusting all possible parameters according to parameter certainty and registration sensitivity, for example, optimal filtering. Such an analysis is difficult to make and may not be any better than making a few reasonable assumptions. Both approaches tried here are relatively easy to implement and are sensible in certain situations.

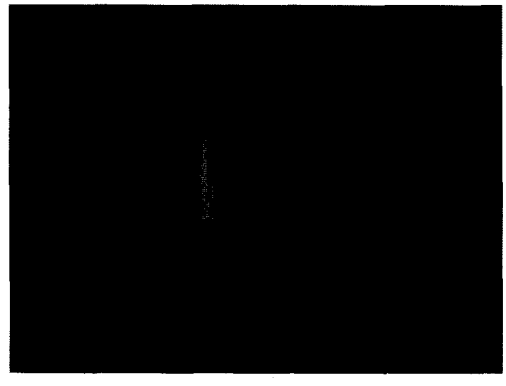
In the first correction approach, if we assume the reported pose of the virtual camera to be correct, there is no way to tell whether registration errors were caused by incorrect camera-to-image mapping, incorrect origin-to-object transformations, or both. By making a further assumption that the camera-to-image mapping is also correct, object positions alone can be adjusted to account for any registration error. To render a corrected image, each misregistered object is temporarily displaced to a position where it will appear to be registered correctly. This correction produces combined images with no measured registration error. Since the registration metric gives no estimate of distance between each object and the camera, virtual objects are displaced on a constant radius (rotated) from the virtual camera viewpoint. This maintains the best estimate of distance between the camera and each object so that objects don't grow or shrink unnaturally.

In the second correction approach, we correct the virtual camera viewpoint to reduce registration error while assuming the object position and camera distortion to be correct. If enough features are visible, it is theoretically possible to compute the camera pose from the 3D (X, Y, Z) feature positions and their corresponding (U, V) image locations. If the feature positions aren't degenerate, we can recover the camera pose by nonlinear methods with a minimum of four points and by linear methods with a minimum of six points.[14,15] Trying to correct the camera position this way isn't practical for at least three reasons. First, there is no way to guarantee enough features will be visible in every image. Second, these solution methods are highly sensitive to noise and spatial feature distribution. Third, a good estimate of the virtual camera position is already available.

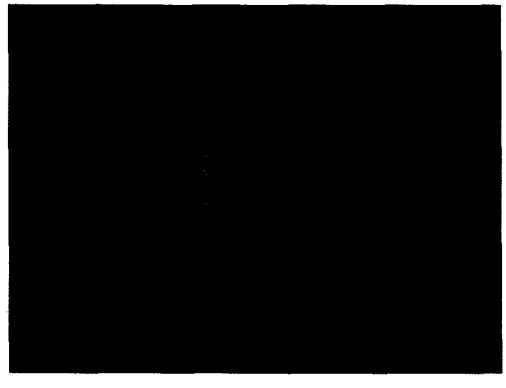
The easiest simplification is to make the registration error entirely due to camera orientation error. We assume the virtual camera position to be correct as reported by the origin-to-head and head-to-camera transformations. This is a good assumption for three reasons. First, orientation corrections can be made when only one feature is visible. If more than one feature is visible, a best-fit solution can be found. Second, under the assumption that objects are relatively far from the camera, which is true in most AR



9 "Open loop" mode without dynamic registration correction or distortion correction. Virtual objects "swim" around and are poorly registered.



10 "Open loop" mode with video delay and optical distortion correction. Virtual objects do not swim as much and registration is somewhat improved.



11 "Closed loop" mode with optical distortion correction, video delay, and dynamic correction of object position and lens distortion. Virtual objects appear to be "nailed" to their reference points.

12 "Closed loop" mode with optical distortion correction, video delay, and camera orientation correction to align base of antenna. Adjustment arrow is steady but slightly misregistered.

applications, registration errors are much more sensitive to errors in camera orientation than camera position. This means that solving for camera position is unstable (sensitive to errors) and that solving for camera orientation when camera position is fixed is well-behaved and relatively insensitive to errors. Third, tracking system data has more error in rotation than in translation. This is because the alignment error perceived when HMD wearers rotate their heads is greater than when they translate them, and the head tracking system used incurs significant delays in reporting measurements (temporal error).[16] In the experimental system, camera orientation error is adjusted by considering only one "reference" feature position and rotating the virtual camera to align with that position. This is only an approximation; it can correct the alignment of a particular point but not an orientation about that point.

Registration results

The experimental system in Figure 6 can be operated in nine different modes by different selections of the two parameters, real-video delay, and registration-correction method. The real video delay is either

1. no delay or distortion correction,
2. delay without distortion correction, or
3. delay with distortion correction.

The registration-correction method is either

A. none,
B. correction by adjusting camera-to-image mapping and/or origin-to-object transformations (move the object), or
C. correction by adjusting camera orientation (rotate the camera).

The results of different combinations of these parameters are described below:

(1,A): This open-loop mode is equivalent to the "current model" shown in Figure 3. Figure 9 shows the result. The virtual objects are not aligned with their proper positions and lag noticeably behind during

user movement in spite of careful calibration and system tuning.

(1,B): This option has good registration at the object feature positions except during user motion when the registration still lags noticeably. It appears possible to shake the virtual objects from their proper positions, but they always return. This case shows the simple power of the closed-loop system model over the open-loop system model in Figures 3 and 9. Despite the lack of lens distortion correction, noticeable lag, and various other errors, the virtual objects still appear to belong in specific spatial positions, a result not easily achieved without dynamic registration correction.

(2,A), (2,B): These combinations have the same static results as (1,A) and (1,B). However, the registration error during motion (temporal registration error) is extremely small because the real video delay is the same as the tracking and image generation delay; the dynamic registration appears to be the same as the static registration. The reduced "swimming" of the virtual objects during motion makes them appear much more stationary and solid, even in the case of (2,A) where the registration is poor.

(3,A): The addition of lens-distortion correction without registration correction produces the best open-loop operation possible with the experimental system (Figure 10). The lens distortion correction improves registration considerably but the virtual objects still wander slightly during movement and appear in different positions as the tracking system exhibits errors within its working volume.

(3,B): This combination of distortion correction, delay, and registration correction by displacing objects produces the best registration in the experimental system (Figure 11). In all cases, during both static and dynamic viewing, the virtual objects appear to be registered correctly with respect to their reference positions. They appear to be "nailed" in place.

(3,C): Here only the reference position for the TV antenna is used to adjust the virtual camera orientation, while the real video is corrected for distortion and delayed (Figure 12). No registration correction is made for lens distortion or object position errors. This combination produces the second-best registration after combination (3,B). The base of the antenna appears to be registered correctly on the house, but the arrow on the disk-drive adjustment screw consistently appears to be just a bit low. This misregistration could be caused by errors in the origin-to-object transformations for the TV antenna and disk-drive screw or by errors in the initial camera orientation, which aren't completely corrected with this method.

(1,C), (2,C): These combinations did not make sense. Without lens distortion correction, you cannot modify the camera position to improve registration for more than one object.

Conclusions and future directions

Building augmented reality systems with accurate registration is difficult. The visual registration requirement between real and virtual objects exposes any measurement or calibration error in the system. The strongest argument in favor of dynamic compensation is that no matter how carefully measurement and calibration are performed, there almost certainly will be errors in the composite images. What the real camera sees must be taken as the ground truth and the registration error between the real and virtual images must be corrected by compensating for errors in calibration, tracking, and/or distortion correction. It is more practical to measure and correct errors using a closed-loop design than to avoid making them in the first place with an open-loop design.

The experiment described here demonstrates the importance and feasibility of dynamically measuring and correcting image-space registration error. The experimental system is more stable and better aligned than systems without registration correction.

The idea of measuring and correcting image registration error has implications for the design of future AR systems. Since feedback can compensate for tracking errors—in essence becoming part of the tracking system itself—less accurate and less expensive tracking systems might be feasible. Optical tracking systems[8] could be designed to use stationary cameras to track a user's position while cameras on the user's head could look outward to determine the user's orientation. Feedback also reduces the accuracy requirements for lens-distortion correction and system calibration.

The trade-off of delaying a user's real-world view to achieve correct registration of virtual objects within that view is a subject for further research. Some applications require, above all else, that users see their surroundings as immediately and as directly as possible, while, in others, the advantages of correct registration outweigh the disadvantages of a small video delay. It should be noted that the video delay introduced merely matches current tracking and virtual-image-generation system latencies. From a user's perspective, the addition of video delay makes interactions within an AR system exactly like interactions within an immersive VR system—the system latency is perceived between any real and observed action. As future image generator and tracking system latencies are reduced, the degree of perceptual miscue will decrease as well.

Haptic interactions in an AR system also encounter latency-induced perceptual miscues. When driven by the graphics system, haptic output can be tightly synchronized with virtual object interactions. In applications where haptic interfaces are observed as virtual effectors such as cursors or probes, a delayed video AR system offers synchronization between all visual and haptic events. Of course, the latency-induced miscue between all real user actions and their perceived effects still exists as noted above.

Further research and experimentation must weigh the perceptual strengths and weaknesses of closed-loop delayed-video AR systems. The two main strengths are

- the capability for consistent synchronization of all perceived cues from the AR system and
- temporal alignment compensation.

The weakness is the latency introduced between phys-

ical actions and their direct observation.

The success of registration correction depends on the ability to accurately measure registration in the first place. This is not a simple task in general. The experiment described here uses an oversimplified method for measuring registration, which may not be practical in many environments. The computer vision community has already done a large amount of work in this area, and some of their results may apply to AR systems.

Correct occlusion cues are still needed for augmented reality systems to be truly believable. This method of registration only works for virtual objects that appear completely in front of real ones. What is really needed is a way to sense positions and depths in the environment from the real camera. With such information, the reference positions could be used to position virtual objects and properly hide them if they were obscured. ■

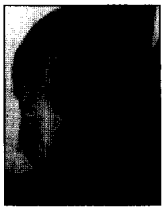
Acknowledgments

Support for this research was provided by The Link Foundation and the NSF/ARPA Science and Technology Center for Computer Graphics and Scientific Visualization (NSF cooperative agreement #ASC8920219). Thanks to Henry Fuchs, Adam Janin, Ron Azuma, and the reviewers for their insightful suggestions.

References

1. S. Feiner, B. MacIntyre, and D. Seligmann, "Annotating the Real World with Knowledge-Based Graphics on a See-Through Head-Mounted Display," *Proc. Graphics Interface 1992*, Canadian Information Proc. Soc.,1992, pp. 78-85.
2. S. Feiner, B. MacIntyre, and D. Seligmann, "Knowledge-Based Augmented Reality," *Comm. ACM*, Vol. 30, No. 7, July 1993, pp. 53-62.
3. T.P. Caudell and D.W. Mizell, "Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes," *Proc. Hawaii Int'l Conf. System Sciences*, Vol. 2, 1992, pp. 659-669.
4. M. Bajura, H. Fuchs, and R. Ohbuchi, "Merging Virtual Reality with the Real World: Seeing Ultrasound Imagery within the Patient," *Computer Graphics* (Proc. Siggraph), ACM Press, 1992, pp. 203-210.
5. E.K. Edwards, J.P. Rolland, and K.P. Keller, "Video See-Through Design for Merging of Real and Virtual Environments," *Proc. IEEE VRAIS*, IEEE Computer Society, Los Alamitos, Calif., 1993, pp. 222-233.
6. B. Adelstein, E. Johnston, and S. Ellis, "A Testbed for Characterizing Dynamic Response of Virtual Environment Spatial Sensors," *Proc. Fifth ACM Symp. User Interface Software and Technology*, ACM Press, New York, 1992, pp. 15-22.
7. S. Gottschalk and J. Hughes, "Autocalibration for Virtual Environments Tracking Hardware," *Computer Graphics* (Proc. Siggraph), ACM Press, New York, 1993, pp. 65-72.
8. R. Azuma and G. Bishop, "Improving Static and Dynamic Registration in an Optical See-Through HMD," *Computer Graphics* (Proc. Siggraph), ACM Press, 1994, pp. 197-204.
9. U.H. List, "Nonlinear Prediction of Head Movements for Helmet-Mounted Displays," Tech. Report AFHRL-TP-83-45, Williams AFB, Ariz.
10. H. Fuchs et al., "Pixel Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories," *Computer Graphics* (Proc. Siggraph), Vol. 23, No. 3, July 1989, pp. 79-88.10.
11. M. Bajura and U. Neumann, "Dynamic Registration Correction in Augmented Reality Systems," *IEEE VRAIS 1995 Proc.*, IEEE Computer Society, Los Alamitos, Calif., 1995, pp. 189-196.
12. A.L. Janin, D.W. Mizell, and T.P. Caudell, "Calibration of Head-Mounted Displays for Augmented Reality Applications," *Proc. IEEE VRAIS*, IEEE Computer Society, Los Alamitos, Calif., 1993, pp. 246-255.
13. J. Weng, P. Cohen, and M. Herniou, "Camera Calibration with Distortion Models and Accuracy Evaluation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 14, No. 10, Oct. 1992, pp. 965-980.
14. R. Horaud, B. Conio, and O. Leboulleux, "An Analytic Solution for the Perspective 4-Point Problem," *Computer Vision, Graphics, and Image Processing*, Vol. 47, No. 33-34, 1989, pp.33-43.
15. S. Ganapathy, "Real-Time Motion Tracking Using a Single Camera," AT&T Bell Labs, Tech Report 11358841105-21-TM, 1984.
16. J. Liang, C. Shaw, and M. Green, "On Temporal-Spatial Realism in the Virtual Reality Environment," *Proc. Fourth Ann. ACM Symp. User Interface Software and Technology*, ACM Press, New York, 1991, pp. 19-25.

Michael Bajura *is a doctoral student in the Department of Computer Science at the University of North Carolina at Chapel Hill. His primary research areas are computer vision and real-time computer graphics for augmented reality applications. He hold a BS in engineering from California Institute of Technology (1988) and an MS in computer science from University of North Carolina (1991).*



Ulrich Neumann *is an assistant professor of computer science at the University of Southern California. His research relates to interactive computer graphics for visualization and virtual environments. He completed his MSEE from State University of New York at Buffalo in 1980 and completed his computer science PhD at the university of North Carolina at Chapel Hill in 1993.*

Readers may contact Bajura at Computer Science Dept., Sitterson Hall, UNC at Chapel Hill, Chapel Hill, NC 27599-3175, e-mail bajura@cs.unc.edu or Neumann at Computer Science Dept., USC, Los Angeles, CA 90089-0781, e-mail uneumann@usc.edu.