Managing Latency in Complex Augmented Reality Systems

Marco C. Jacobs* Mark A. Livingston Andrei State

University of North Carolina at Chapel Hill
*Delft University of Technology, the Netherlands
{jacobs|livingst|state}@cs.unc.edu

Abstract

Registration (or alignment) of the synthetic imagery with the real world is crucial in augmented reality (AR) systems. The data from user-input devices, tracking devices, and imaging devices need to be registered spatially and temporally with the user's view of the surroundings. Each device has an associated delay between its observations of the world and the moment when the AR display presented to the user appears to be affected by a change in the data. We call the differences in delay the relative latencies. Relative latency is a source of misregistration and should be reduced. We give general methods for handling multiple data streams with different latency values associated with them in a working AR system. We measure the latency differences (part of the system dependent set of calibrations), time-stamp on-host, adjust the moment of sampling, and interpolate or extrapolate data streams. By using these schemes, a more accurate and consistent view is computed and presented to the user.

CR Categories and Subject Descriptors: I.3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality; I.3.1 [Hardware Architecture]: Three-dimensional displays; I.3.6 [Methodology and Techniques]: Interaction Techniques.

Additional Keywords: Augmented Reality, Latency Management, Ultrasound Echography.

1 INTRODUCTION

Augmented reality (AR) is the term used to describe systems in which the user is presented with an enhanced view of the surroundings. This view is created by compositing computer graphics with a view of the real world. The graphics must be generated in such a way that the user believes that the synthetic objects exist in the environment. Azuma [2] gives an introduction to the field of AR, the technology that can be used to achieve it, current existing applications, and the potential of the paradigm.

1.1 Motivation

As has been recognized in previous AR systems, *registration* (or alignment) of the synthetic imagery with the real is crucial. AR systems have a variety of input streams (tracking devices, real-time imaging devices, user input devices and others), which differ in accuracy, bandwidth, dynamics and frequency. The devices need to be both spatially and temporally registered.

Consider the example of a movie in which sound effects are overdubbed. The illusion suffers if a crash is not heard at the precise time an object is seen to hit the ground. Although both the video and audio signal may have considerable delay, it is the *difference* in delay that is noticed[1]. The differences in latency between data streams or external signals cause misregistration. In the rest of this paper, we call the difference in latency between two streams the *relative* latency. We concentrate on minimizing relative latencies since they are sources for misregistration.

1.2 Contribution

We develop methods for measuring relative latency and a variety of techniques for managing latency to reduce the misregistration it causes. We introduce these methods while keeping in mind that we are building a complete system, and thus must focus on our goal–providing a convincing and accurate illusion. We apply these methods to a previously described AR system for real-time ultrasound visualization [13] and demonstrate improved registration and visualization resulting from our latency management techniques.

1.3 Latency sources

The following sources of delay have been classified [8][16][14]:

Off-host delay: Duration between the occurrence of a physical event and its arrival on the host. ($T_{offhost}$)

Computational delay: Time elapsed while the data is in the host system and while the system is doing computations. (T_{comp})

Rendering delay: Time elapsed while the graphics engine is generating the resulting picture. (T_{render})

Display delay: Time elapsed between sending images to the display and the display actually showing them. ($T_{display}$)

Synchronization delay: The time in which data is waiting between stages without being processed. (T_{sync})

Frame-rate-induced delay: Between two frames the display is not updated, causing the user to see an outdated image stream. This delay can also be considered a special case of synchronization delay between the display system and the human eye.

Relative latency has its source in off-host delay, computational delay and synchronization delay. The data from separate external devices follow different paths in the system and each path has its own latency. The relative latency between the different data paths causes misregistration. Rendering

[1] Although this example considers a non-interactive system, it shows the importance of temporal synchronization

delay and display delay do not contribute to misregistration because all the data follows one path[2]. Delay in this path will result in a lower frame rate and in higher latency between real-world events and the displayed image, but will not cause any misregistration since the relative latency between streams here is constant. Figure 1 explains the symbols and terms used.

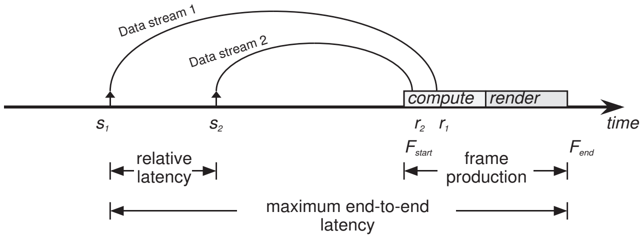

Figure 1: Acquiring data and producing a frame. s_i is the time a real-world sample was taken by the external device and r_i is the time it arrived on host. $r_i - s_i$ is the off-host latency ($T_{offhost}$) for stream i. F_{start} and F_{end} denote the start and end times for the generation of the frame.

2 PREVIOUS WORK

Many authors have examined latency and have tried to reduce its effects. Previous efforts can be categorized as bounding latency, reducing latency, compensating for latency, and achieving registration despite latency.

Time-critical computing [5][4][15] is a technique in which quality is traded for speed. The application is constantly aware of time. Reducing latency here means reducing computation accuracy, which might not be always advisable.

Most real-time graphics system are aimed at high throughput instead of low latency. High throughput is achieved through pipelining which results in latency. Olano et al. [9] specifically discuss a low-latency rendering system and a technique to reduce errors caused by delays in the display system. Parallelization reduces latency for T_{comp}, T_{render} and T_{sync} by increasing throughput. Wloka [16] dedicates a processor to sample the external data streams at a high frequency.

Prediction of future head position and orientation can be used in order to reduce perceived delay [1]. This position and orientation is estimated by extrapolation of current and older values. This predicted value can then be used to generate an image for the time when the image will be displayed. Since T_{render} is not a constant, it may be difficult to determine the extrapolation interval.

Image generation delay can be reduced by using a post-rendering warp. Rendering starts with an initial guess for the head position. This process results in a data structure from which an image can quickly be computed based on a newer head position. This newer head position can be acquired after rendering or be approximated through prediction. Mark et al. [7] use a warping mechanism to reduce latency caused by bandwidth limitations. Regan et al. [10] render the scene on the faces of a cube. A head rotation causes an address offset for the final image to be taken from this cube.

State et al. [12] and Bajura et al. [3] synchronize a video stream with head-tracking data by reducing the head-tracking error with the use of videometrically tracked landmarks. Bajura et al. [3] temporally synchronize the output of a rendering system and a video stream by buffering the lower latency video.

3 METHODS

The naive way of writing an AR application is to sample all the data streams at the start of a frame after which the application starts computing and rendering (Figure 2). This implies that those sources whose data is not required immediately will have extra computational delay associated with them, and leads to large relative latency values ($s_i - s_j$) causing misregistration and a large maximum end-to-end latency ($F_{end} - \min(s_i)$).

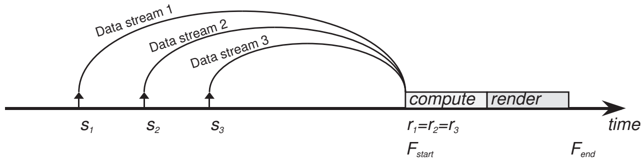

Figure 2: Naive order of acquiring data. The application samples all data streams at the beginning of a frame, then computes and renders the output. This leads to high maximum end-to-end latency and high relative latency.

If we know the moment in time that the data was sampled (s_i), or the difference in time between the samples of two streams, we can adapt the program to reduce the effects of the known relative latencies. Unfortunately, most data streams are not timestamped at the source. However, we can measure the relative latency between data streams with experiments (see section 4.1). Timestamping upon arrival on host enables us to measure the dynamic computational delay (T_{comp}).

Once we know the relative latencies, we can reduce the effects by adjusting the *moment of sampling*. Sampling a higher latency stream later than a lower latency stream will reduce the relative latency between the two streams. Another method is to use *interpolation* and *extrapolation* to compute a value for a data stream for any moment in time based on previously sampled data values. These techniques reduce relative latency and therefore improve registration.

3.1 Adjusting the moment of sampling

Perhaps the simplest technique to reduce relative latency is to adjust the moment of sampling the incoming data stream. This requires no computation or special hardware, and proves to be a useful method for reducing relative latency.

One method to do this would be to schedule the polling of input devices at relative intervals that correspond to the

[2] Our system captures the live video on host in order to mix it with graphics. In some other AR systems, the live video never enters the host. The graphics and video are mixed by chroma or luminance keying. This can introduce misregistration caused by relative latencies. An optical-see-through setup also suffers from the relative latency between the latent rendered images and the zero-latency real world.

relative latency between the various devices (Figure 3), and wait between readings. Processing begins after all input has been received. However, this would increase the end-to-end latency of the frame produced from this data and reduce the frame rate to $\frac{1}{\max(r_i)-\min(r_i)+Tcomp+T_{render}}$.

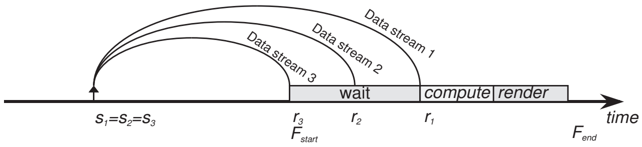

Figure 3: Deferred acquisition of data. The data streams are sampled in order of increasing latency. The wait times are equal to the respective relative latencies. This results in zero relative latencies, but very high maximum end-to-end latency.

A compromise that does not increase maximum latency in order to decrease relative latency is just-in-time acquisition of the data while interleaving computation with the data acquisition (Figure 4). This reduces both relative latency and maximum end-to-end latency, since wait time is filled with useful work.

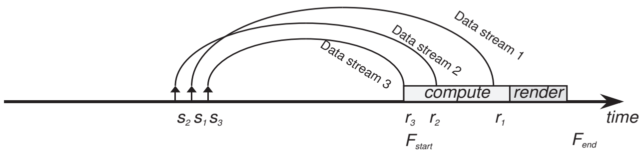

Figure 4: Just-in-time acquisition of data. Polling of data streams is delayed until the data is required for computation to continue or when the relative latency is zero. This reduces relative latencies and maximum end-to-end latency.

3.2 Temporal interpolation and extrapolation

The next method is to store multiple readings and either interpolate or extrapolate these readings to simulate new readings. The value of a data sample for each moment in time can now be computed.

For example, if a tracking system's acquisition path has lower latency than the video camera's acquisition path, we can buffer readings and interpolate the position and orientation reported.

On the other hand, if a tracking system's acquisition path has higher latency than the video camera's acquisition path, we can use predictive tracking methods [1] to compensate for the difference.

4 CASE STUDY: AR ULTRASOUND SYSTEM

Our testbed AR system is the ultrasound visualization system currently being developed at the University of North Carolina at Chapel Hill [13]. This system is a video-see-through AR system designed for the medical procedure known as ultrasound-guided needle biopsy. The physician wears a head-mounted display fitted with two video cameras. A Flock of Birds (FOB) magnetic tracker from Ascension Technology Corporation is used to track the head-mounted cameras. Ultrasound image data is acquired from a Pie Medical Scanner 200. A Metrecom IND-01 (Faro) mechanical arm from Faro Technologies, Inc. tracks the ultrasound probe. The system runs on a Silicon Graphics Onyx Reality Engine^2. It is equipped with Sirius Video^{TM} unit for video capture and Multi-Channel Option^{TM} for stereo display. The data from the four input streams is captured asynchronously on a per-frame basis from the four input devices under CPU control. The ultrasound images are rendered and merged with a view of the patient.

4.1 Relative latency measurements

We first determined by visual tests that the real-world camera video is the lowest latency stream. For the external device trackers, we determined relative latency by rendering a model in the coordinate system of the tracker that should be aligned with the real-world object. It was then easy to visually determine whether the tracker lagged behind the camera video by moving the tracked object and checking whether the virtual or real copy appeared to move first in the AR view. Table 1 summarizes the results.

Rel. Lat. (ms)	Cameras	Faro	FOB	Ultrasound
Cameras	-	30	30	250
Faro		-	0	220
FOB			-	
Ultrasound				-

Table 1: Measured Relative latency between the data streams. Relative latency between the different data streams. These measurements were made at a frame rate of 10Hz.

The tests showed that both the Faro and the FOB have higher latency than the real-world video. We applied linear predictors to compensate for this latency. By adjusting the prediction interval until the rendered coordinate system matches the tracker image in the video, we measured the latency. Plate 1, center, and Plate 4, center, show the result for the Faro tracker without prediction and with partially improved registration via prediction. We could buffer the video over an interval, but video is high bandwidth and expensive to store and move in memory. Also, the end-to-end latency of the video would slow the response to the user's head movements, thereby diminishing the AR illusion.

In order to measure the relative latency between the camera video and the ultrasound video, we used a single live video signal sent into both video paths. By positioning the ultrasound image data in the AR view, we visually aligned the "ultrasound data" with the real world. By imaging a rotating drum with a regular pattern of vertical bars, we determined the latency between the two streams by aligning the vertical bars in one image to be exactly one pattern width behind the other image (Plate 2). We computed the relative latency from the rotational velocity of the drum.

We next measured the latency between the FOB and the Faro. We did this by rigidly fixing the two to each other via an intermediate plastic rod. The rod reduces magnetic interference of the FOB by the Faro's stainless steel mount plate. Using wooden blocks, we created a track for moving this assembly back and forth. By moving the assembly in an oscillating pattern, we determined the latency between the two trackers by examining a graph of their respective positions versus a global timer. The relative latency between the two streams proved to be small and dependent on T_{sync}. More precisely, the difference in frequencies at which the devices operate caused a variable delay in the signal. Neither the Faro nor the FOB was consistently ahead of the other.

Finally, we measured the relative latency between the ultrasound image data and the mechanical tracker by holding the tracker stationary until the image data catches up, then setting a marker at the location of a bolt being imaged. We then swept the probe in an oscillating pattern, and progressively increased the delay added to the mechanical tracker readings until the ultrasound image data did not appear to waver from the known location of the bolt.

It appeared that the ultrasound image has higher latency than the tracking data. We decided to buffer the tracking data over an adjustable interval. We again made a sweeping motion over the bolt and adjusted the interval until the imaged object did not lag behind and appeared in the same place as the real object. The relative latency between the ultrasound image data and the Faro was 220ms at a frame rate of 10fps. The reason for such a high value is the long video format conversion path from the ultrasound machine to the host. This path could be shortened with hardware solutions currently not available to us.

4.2 End-to-end latencies

In addition to knowing and managing the relative latency, we wanted to know the end-to-end latencies of the streams in the system–that is, the latencies between the data streams and the real world. We measured the latency of the camera to the real world. To do this, we used an LED blinking at a rate of 5Hz, set with a pulse generator. We used the LED as one trigger for an oscilloscope. We pointed our video-see-through camera at the LED. We taped a photoelectric sensor to the monitor where the image of the LED appeared, and connected it to the oscilloscope as a second trigger. (Figure 5) This allowed us to see the two signals on the oscilloscope screen simultaneously: one with negligible latency that came directly from the pulse generator and one that came from the photoelectric sensor. We measured the latency between the real world event (LED blinks) and the time the photosensor "saw" that event by measuring the distance of the two signals on the oscilloscope. For our system, this value was 40ms, and ranged between 30ms and 60ms. We took these measurements with computation turned off, so that the latency was $T_{offhost} + T_{display} + T_{sync}$.

We measured the end-to-end latency of the camera (40ms). Adding the relative latency between these two streams (30ms), we conclude that the results of the experiments are consistent with the Faro specification. The off-host latency of the Faro is specified as 67ms [11].

4.3 Results

First, we arranged the order of polling the devices to reflect the latencies measured in the previous section.

1. Read Faro and FOB trackers for prediction.
2. Capture camera video.
3. Read FOB tracker again.
4. Invoke vision-based hybrid tracker [12].
5. Read Faro tracker again.
6. Capture ultrasound video.
7. Render world.

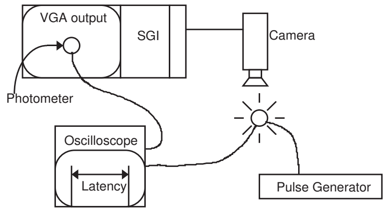

Figure 5: SYSTEM DIAGRAM OF CAMERA LATENCY EXPERIMENT. We measured the end-to-end latency of the camera video stream. The pulse generator triggers the LED, which in turn triggers the oscilloscope. The LED blinks and is seen in the camera image, which is subsequently seen by the photosensor. The photosensor also triggers the oscilloscope. This causes two traces to appear on the screen of the oscilloscope. By reading the scale of the oscilloscope and measuring the distance between the traces, we measured the latency.

We placed the reading of the Faro tracker (5) directly after the (expensive) vision-based tracking computation (4), and the acquisition of ultrasound video data (6) directly after the reading of the Faro tracker. We would have liked to have placed the reading of the FOB tracker (3) later as well, but it is necessary for the initial guess for the hybrid tracker. This arrangement significantly decreased the relative latency in our system.

By predicting every stream to match the environment the user is in, the system would behave as if it had no latency. Only errors in the prediction can spoil the illusion. However, video streams are hard (or impossible) to predict since they have high bandwidth and are dynamic in behavior.

Our second option is to buffer and interpolate lower latency streams to match the higher latency ultrasound stream. Buffering the real-world video is expensive. We can only store the video directly in the frame-buffer. Also, the perceived delay of the system would grow since the real-world video gives the most visual cues to the user. We therefore decided to define two synchronization points, represented by the ultrasound and real-world video streams. We had to synchronize the tracker information with these two points (s_{US}, s_{Camera}). (Figure 6)

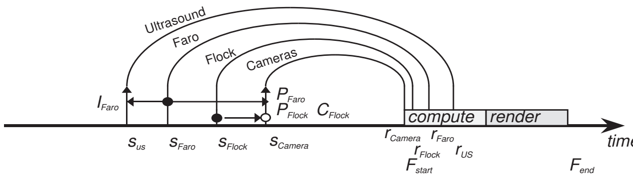

Figure 6: EXTRAPOLATION, INTERPOLATION AND JUST-IN-TIME SAMPLING. Just-in-time sampling is used to reduce relative latency. The FOB is predicted to match the video after which it is corrected with a vision-based algorithm. The Faro is buffered and interpolated to match the ultrasound video stream, and predicted to match the cameras.

We used linear extrapolation based on the last two values for the ultrasound probe's position and orientation. (P_{Faro}) The prediction interval is based on the relative latency between the Faro and the cameras. Both streams are timestamped as they arrive on the host (r_{Faro}, r_{Camera}). Timestamping is done by reading a system clock before and after sampling and averaging these values in order to compensate for the time it takes to sample the stream. The extrapolation interval for the probe relative to the last Faro reading is:

$$r_{Faro} - r_{Camera} + RelativeLatency$$

The rendered model of the probe is now registered with the video image of the probe. The model's depth values are used in order to achieve correct occlusion.

We have eliminated a relative latency of 30ms in the tracking of the ultrasound probe. In analyzing registration error, Holloway [6] found that 1ms of latency can cause 1mm of registration error, so we have eliminated a potential source of up to 30mm of registration error. We of course introduce error in the prediction, but this is a small price.

We also use a linear extrapolation scheme for the camera position and orientation (P_{Flock}, Figure 6) in order to synchronize with the real-world video stream. However, due to the noise and inaccuracy of the magnetic tracker signal the linear predictor is not precise enough. We therefore apply a vision-based corrector (C_{Flock}, Figure 6) that relies on color-coded landmarks visible in the image [12]. This corrector typically eliminates relative latency (and inaccuracies in the tracking report), since its tracking information comes from the video image, to which we are synchronizing (s_{Camera}) (Plate 1, left; original, Plate 4, left; method applied). We have thus eliminated a relative latency of 30ms in head tracking. Again, this was a potential source of up to 30mm of registration error.

Finally, we need a solution for the relative latency between the ultrasound video stream and the probe tracking data. The latency would otherwise cause misregistration (Plate 1, right and Plate 3). Since the Faro has lower latency than the ultrasound video, (and we cannot predict the video) we store and interpolate (I_{Faro}, Figure 6) the tracking information. Again, upon arrival on host we timestamp the video image and the tracking data. This tracking data is then stored in a buffer. The interpolation interval for the probe relative to the last Faro reading is :

$$r_{Faro} - r_{Ultrasound} + RelativeLatency$$

A linearly interpolated value is computed from the previously stored data for the probe's position at the time the ultrasound image was taken. This computed position and orientation is then used for rendering the ultrasound slices. By applying this interpolation technique, we align ultrasound slices spatially with (the video of) the patient. This detaches the slice from the visible probe (in particular when the probe is moving), but eliminates relative latency between the ultrasound image data and the probe tracking data. This eliminates the relative latency as a source of up to 220mm of registration error between the location of the acquired ultrasound slices and the patient.

5 CONCLUSIONS

We have given general methods for handling multiple data streams. These methods are applicable to AR systems that have multiple input streams with different latencies. By eliminating relative latency, we remove a potential source of registration error. While latency-based misregistration often goes unnoticed or is considered unimportant, performance can be improved with simple synchronization schemes.

Relative latency can be managed by measuring the latency differences (part of the system-dependent set of calibrations), on-host timestamping, adjusting the moment of sampling, and interpolation or extrapolation of these data streams.

In our ultrasound system, synchronizing the probe tracking data with the ultrasound video data and prediction of the probe model's position and orientation reduced errors significantly. The vision-based tracking scheme reduces latency-based misregistration by reducing the error.

These schemes show the importance of thinking about latency early in the design phase. Sampling a stream once per frame might not be sufficient if prediction is used. Adjusting the moment of sampling requires rearranging code which might be difficult to adjust later in the design phase.

6 FUTURE WORK

The sampling frequency of the data streams in our system is currently proportional to the frame rate. We would like to have autonomous sampling processes for each stream [16]. Every stream could then have its own (higher) sampling frequency making it possible to have more accurate interpolation and extrapolation functions. For video images this is hard, since video has high bandwidth and moving video around in memory costs time, but for tracking information this is easily feasible. Having more knowledge about the frequency and phase of the external device also holds the potential for further adjusting the moment of sampling.

Our assumption of off-host relative latency being static would not be necessary if manufacturers of the data-gathering devices would timestamp the data with a real-world clock value. We think and hope it will be more common in the future for data gathering devices to have real-world clock timestamping mechanisms. For the video signals, we could have a hardware device inserting a bit pattern at the source, representing the real-world time. This bitpattern could then be read on host. If all devices had timestamping mechanisms, latency measurement experiments and on-host timestamping would not be necessary. This would allow dynamic measurement of relative latency and thus more accurate compensation using the techniques we described.

Our hardware platform uses the Unix operating system. In Unix there are no accurate timing guarantees. The program is written in the C language. A better suited operating system and programming language to real-time actions would be useful. Even on such a machine one could not guarantee constant end-to-end latency due to lack of synchronization between input and output video streams. This could be remedied by applying a generator lock (genlock) to both the input and the output video streams. A similar level of control could be achieved by timestamping the vertical retrace events of input and output video streams.

7 ACKNOWLEDGMENTS

We would like to express our gratitude to Henry Fuchs, Bill Garrett, Todd Gaul, David Harrison, Gentaro Hirota, Erik Jansen, Kevin Jeffay, Bill Mark, Mark Mine, Etta D. Pisano, Stephen M. Pizer, Frits Post, Paul Rademacher, Allen Sajedi, Mary Whitton, and the anonymous reviewers,

This work was supported in part by the ARPA DABT63-93-C-0048 ("Enabling Technologies and Application Demonstrations for Synthetic Environments"), the NSF Science and

Technology Center for Computer Graphics and Scientific Visualization, and PIE Medical Corporation. Approved by ARPA for Public Release–Distribution Unlimited.

References

[1] Azuma, R., and Bishop, G. A Frequency-Domain analysis of head-motion prediction. In *SIGGRAPH 95 Conference Proceedings* (Aug. 1995), R. Cook, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 401–408. held in Los Angeles, California, 06-11 August 1995.

[2] Azuma, R. T. A survey of augmented reality. In *Computer Graphics (SIGGRAPH '95 Proceedings, Course Notes #9: Developing Advanced Virtual Reality Applications)* (Aug. 1995), pp. 1–38.

[3] Bajura, M., and Neumann, U. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications 15*, 5 (Sep 1995), 52–60.

[4] Funkhouser, T. A., and Séquin, C. H. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Computer Graphics (SIGGRAPH '93 Proceedings)* (Aug. 1993), J. T. Kajiya, Ed., vol. 27, pp. 247–254.

[5] Holloway, R. L. Viper: A quasi real-time virtual-environment application. Tech. Rep. TR92-004, Department of Computer Science, The University of North Carolina, 1992.

[6] Holloway, R. L. Registration errors in augmented reality systems. Ph. D. Dissertation TR95-016, Department of Computer Science, The University of North Carolina, 1995.

[7] Mark, W. R., Bishop, G., and McMillan, L. Post-rendering 3-d warping for latency compensation. In *1997 Symposium on Interactive 3D Graphics* (Apr. 1997), ACM SIGGRAPH.

[8] Mine, M. Characterization of end-to-end delays in head-mounted displays. Tech. Rep. TR93-001, Department of Computer Science, The University of North Carolina, 1993.

[9] Olano, M., Cohen, J., Mine, M., and Bishop, G. Combatting rendering latency. In *1995 Symposium on Interactive 3D Graphics* (Apr. 1995), ACM SIGGRAPH, pp. 19–24. ISBN 0-89791-736-7.

[10] Regan, M., and Pose, R. Priority rendering with a virtual reality address recalculation pipeline. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (July 1994), A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 155–162. ISBN 0-89791-667-0.

[11] Sajedi, A. Private communication, October 1996.

[12] State, A., Hirota, G., Chen, D. T., Garrett, W. F., and Livingston, M. A. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 429–438. held in New Orleans, Louisiana, 04-09 August 1996.

[13] State, A., Livingston, M. A., Hirota, G., Garrett, W. F., Whitton, M. C., and Fuchs, H. Technologies for augmented-reality systems: Realizing ultrasound-guided needle biopsies. In *SIGGRAPH 96 Conference Proceedings* (Aug. 1996), H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 439–446. held in New Orleans, Louisiana, 04-09 August 1996.

[14] Taylor, V. E., Stevens, R., and Canfield, T. Performance models of interactive, immersive visualization for scientific applications. In *Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualization* (July 1995), pp. 238–252.

[15] Wloka, M. M. Dissertation proposal: Time-critical graphics. Tech. Rep. TR-CS-93-50, Computer Science Department, Brown University, 1993.

[16] Wloka, M. M. Lag in multiprocessor virtual reality. *Presence: Teleoperators and Virtual Environments 4*, 1 (Winter 1995), 50–63.