Post-Rendering Image Warping for Latency Compensation
UNC-CH Computer Science Technical Report #96-020

William R. Mark, Gary Bishop, Leonard McMillan

January 10, 1996[*]

Abstract

Systems that provide remote viewing of three-dimensional data with interactive viewpoint control must confront two key problems: latency and bandwidth. The straightforward approach of transmitting and displaying rendered images results in a delay of one round-trip between viewpoint change and the corresponding change in the displayed image. We avoid this delay by transmitting a representation of the scene to the user's machine, which then locally closes the viewpoint-to-display loop. If the scene representation is geometry-based, the bandwidth, user-side storage, and user-side graphics rendering capability required for updates to the scene are unbounded. We show that an image-based representation can allow for arbitrary scene changes, while requiring only fixed bandwidth, storage, and rendering power. We demonstrate a system that renders images on a "rendering server", and then transmits them to the user's machine where image warping using per-pixel disparity values compensates for system latency and synthesizes stereo images for display. We also develop enhancements to the warping technique that improve its quality and speed.

Introduction

In this paper we address the problem of rendering imagery at one location for a user who is controlling the view from a second location. If the user and rendering engine are separated by a significant distance–they could be on opposite sides of a continent or of the world–then we are presented with two major difficulties: latency and limited bandwidth.

Latency is unavoidable because of the distances involved. When we send messages over a communications link which is several thousand miles long, even the delay caused by the speed of light becomes significant. Any actual system will also add latency caused by rendering and other computations. Limited bandwidth is a secondary difficulty since it restricts the ways in which we can attack the latency problem. Although high bandwidth is becoming cheaper and more widely available, we do not believe that it will become either "unlimited" or "free" in the near future.

The real-time remote image display system we have built and describe in this paper tackles the dual problems of latency and bandwidth in a very general way, without constraints on the scene or its dynamics. Our system generates and transmits an image-based representation from a rendering server to the user's system. The required transmission bandwidth is independent of scene complexity and depends only on the user's display resolution. The user's system then warps the image-based representation to produce the final displayed image.

Image-based representations and image-based rendering are recent developments in computer graphics. Instead of using geometric models as a scene description, image-based methods use a series of reference images to model the scene. Image-warping techniques like those discussed in [Wolberg90] are used to reproject the reference images according to the desired viewing parameters. In our system we have used the perturbed-projective image warps described by [McMillan95a] to reproject and render the desired image. This method augments the typical image representation with a disparity value at each pixel. From this point on we will refer to this class of perturbed-projective image warps simply as an image warp.

This image warp has the important property that its cost is proportional to display resolution, not to scene complexity. As scenes rapidly become more complex while display resolutions change slowly, image warping becomes more and more attractive as an alternative to rendering.

There are a wide variety of uses for a system in which images are rendered remotely, but viewed locally with local viewpoint control. The application which initially drove our research was the desire to make use of graphics supercomputers from other locations without moving the actual machine or the user. But since the image warp is relatively simple and its cost is bounded by the display resolution, we believe that the user side machine can eventually be made cheaply, opening up other uses. In particular, our technique may be attractive for applications consisting of many user-side machines which must be inexpensive and a smaller number of central rendering servers. In the future these user-side machines could be PC's or set-top boxes. Finally, we believe that there is a class of applications where, because of the bandwidth limitations for remote display systems, an image-based technique is the only technically viable one. The key observation in favor of this argument is that image-based representations of a scene provide the optimal level-of-detail for representing an arbitrarily complex scene.

In the rest of the paper we will discuss in more detail the issue of latency in remote display, categorize this latency into two types, and discuss our system's approach. We discuss the implications of limited bandwidth for remote display and the advantages and disadvantages of our system with respect to other techniques. We then describe our system and its performance. We discuss some of the improvements we have made to the image warping technique as part of our system. We conclude with a discussion of ways in which our system can be

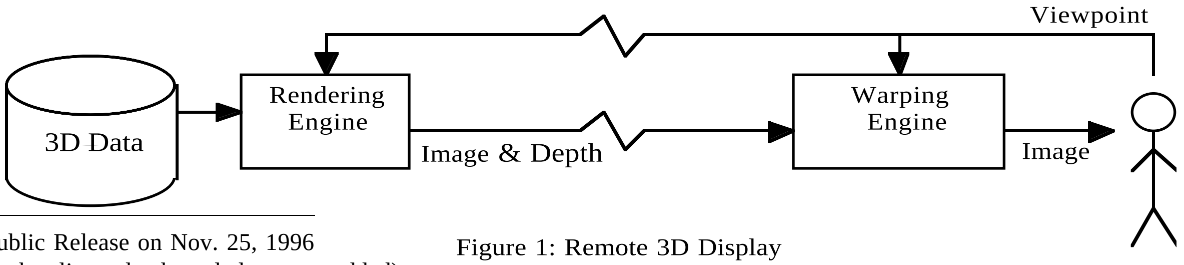

Figure 1: Remote 3D Display

[*] Public Release on Nov. 25, 1996 (with author list and acknowledgements added).

enhanced and integrated with other latency compensation techniques.

Latency and Remote Display

We consider the general type of system in which visual information is generated at one location, and viewed by a user at a second location. The user controls the viewpoint and gaze direction. This control can be in the form of head or gaze tracking, joystick, or mouse control. In this type of system, we can consider two kinds of latency: The first is the delay between the user's specification of a new viewpoint/view direction and the display of the appropriate image (view transform latency). The second is the delay between a change in the underlying visual information and the display of this changed information (scene change latency). In the first case, we are changing how the data is viewed, but in the second case, the data itself is changing. The key difference between these two types of latency is the location where the change is initiated -- at the user's side for view transform latency, and at the information-generating side for scene change latency.

Unless we can accurately predict future scene changes, there is nothing we can do about scene change latency. We are constrained by the speed of light -- the change is happening at one location, and we simply can't immediately find out about it at a different location. In practice of course, most remote display systems add significant additional latency above that imposed by the speed of light, but even if this latency could be eliminated the transmission link delay would remain. Fortunately, scene change latency can not be perceived unless we are interacting with the scene, thus creating a servo loop from us to the scene and back. Without interaction, we are simply viewing a delayed "movie" and have no way of knowing that we seeing it with a time delay. Even with interaction, scene change latencies of a significant fraction of a second may be quite tolerable, depending on the application.The situation for view transform latency is much more promising. Because the change in the view transform is generated at the user's side, where the images of the scene are displayed, we can potentially create an entirely local "view transform to display" servo-loop. However, in order to close this loop locally, we must maintain a representation of the scene at the user's side which allows for changes to viewpoint and gaze direction [Fig 1]. A geometry-based representation would seem to be ideal, except that it requires potentially unbounded storage and rendering performance on the user side because scene complexity is potentially unbounded. It also requires potentially unbounded transmission bandwidth. We propose that an image-based representation is an alternative for many applications. An image-based representation's storage requirements, required rendering performance, and required transmission bandwidth are all bounded by the display resolution, not by the scene complexity.

A conventional image-based representation of a scene is accurate for only one viewpoint, and while it can accommodate arbitrary rotations, it can not be used to produce correct displayed images after a translation. When compensating for relatively short delays, such as those found in a non-distributed system, the errors due to translation may be small enough that they can reasonable be ignored. This approach is the one taken by earlier image-based latency compensation schemes, such as image shifting [Burbidge89, So92], and Regan & Pose's address recalculation [Regan94]. However, for the delays on the order of 100's of milliseconds encountered in remote display systems, the errors due to translation are too large to ignore. We overcome this rotation-only restriction by augmenting our images with per-pixel disparity values, and using the more general perturbed-projective image warp. This image warp correctly handles translation.

A shortcoming of this image warp is that only pixels visible in the source image are mapped to the output image. This behavior introduces "blank" or "empty" areas in regions of the output image that were occluded in the original image. We will discuss these artifacts in more detail later, and propose techniques for reducing or eliminating them.

Bandwidth and Remote Display

Previous approaches to the remote display problem have relied upon maintaining a geometric representation of all or part of the scene on the viewer's computer. Changes to the scene sent from the "database" end are in the form of new geometry or changes in modeling transforms for existing geometry, or in some cases "higher-level" application-specific information. The best known examples of this type of system are based on the Distributed Interactive Simulation (DIS) standard. [Zyda93, Katz94] DIS systems transmit new vehicle locations (model transforms) to the user's computer, as well as simulation-specific events such as explosions. DIS also transmits "predictive" information about future vehicle locations. The DIS system is an extremely efficient user of bandwidth, but it does so at the cost of maintaining a model on the user's machine, and by allowing only certain types of easily described scene changes.

Consider a system which needs to remotely display models of arbitrary complexity, and allow any number of arbitrary changes to the model in a frame. Any geometry-based remote display scheme will require potentially infinite bandwidth to the user's machine to transmit these changes. In contrast, our image-based scheme requires a fixed bandwidth to transmit the new image representation each frame. What we are really doing is using the image-based representation as a means to represent the geometry at an appropriate level of detail.

The crossover point where the image-based representation becomes more efficient than a geometry based representation is reached approximately when the number of polygons in the geometry based representation is greater than the number of pixels in the display. (In fact, it occurs somewhat earlier, since a polygon is more complicated than a pixel). Today's most complex models have already passed this crossover point, and as models continue to become more complex this situation will become more common

For models which are large but below this crossover point, we can still reap the computational benefits of the image warp. The simple and regular structure of the warp computations makes warping very competitive with traditional rendering even for relatively simple models.

For many applications, it is not necessary to update the image-based representation at 60Hz. Because the viewer's display updates are decoupled from updates to the image-based representation, a system can present the user with a 60Hz update rate for view transform changes, while presenting a lower update rate for scene changes. The effect is that dynamic objects in the scene appear to move in a discontinuous and jerky manner, but the sense of presence the user gets from low latency head rotation/translation is preserved. This idea is not new -- see for example Gossweiler et al [Gossweiler93] -- but it has important implications for our system. If there are few or no moving objects in the scene, then we may be able to tolerate scene update rates on the order of 1 Hz, thus reducing our transmission link bandwidth requirements and making our rendering server cheaper. However, if the scene update rate falls below 1 Hz then the user has the opportunity to move too far from the viewpoint for which the scene was rendered. When such movements occur, the viewpoint coherency assumption which is implicit in the use of the image-based representation begins to break down. The result is undesirably large occlusion artifacts which appear as the user's viewpoint changes.

Additional Advantages of Image Representation

An image-based local-representation provides several additional advantages beyond that of bounded bandwidth requirements. If the scene being viewed is a model of a design for a new ship, car, or aircraft, the geometric representation of the model may be considered proprietary. By transmitting only rendered images of the geometry, rather than the model itself, the model owner retains control over the model. The generality of our technique also makes it very simple to integrate with any existing application. The technique does not depend on any sort of application-specific information -- it only requires the capability to generate images and their corresponding z-values from appropriate viewpoints. The current implementation of our system uses the Silicon-Graphics Performer rendering library to generate our image-based representation, and we can view any model which Performer is capable of loading.

The Remote Rendering System

We have constructed a proof-of-concept system to illustrate the major ideas presented above. The system consists of two Silicon Graphics Onyx computers connected via a 155 Mbit/sec ATM link. One machine is on the "rendering" side of the link, and the other is on the "user" (warping) side. The machines are physically adjacent to each other, so we simulate latency over the ATM link by buffering data on the user-side computer.

The rendering computer is a 2-processor SGI Onyx/Reality-Engine 2. For each viewpoint transmitted from the user-side computer, this rendering computer renders four sides of a cube (we skip the top and bottom sides for performance reasons). We use the Silicon Graphics Performer library to generate the images, with a callback to read the RGB, and Z values from the framebuffer. The disparity values required by the image warping algorithm are calculated from the Z values. The X and Y image space derivatives of the disparity are also calculated. The RGB and disparity values are then transmitted to the user-side computer over the ATM link. An ideal system would reduce bandwidth requirements by compressing the RGB and disparity values, but we have not implemented this improvement.

The user-side computer is a 4-processor SGI Onyx with an attached head position/orientation tracker and head-mounted display. Our image warping is done entirely in software. One of the four Onyx processors is devoted to communications with the rendering computer – sending viewpoints and receiving rendered images. The other three processors perform the image warping, using up-to-date position and orientation reports from the tracker to compute final displayed images from the intermediate images. By adding the appropriate position offsets for each eye to the head position, our system can generate stereo imagery for the head-mounted display without requiring any additional information from the rendering computer.

Our system currently has a typical frame rate of 7 frames per second for monocular output at 320 x 240 resolution.[*] This output is for a 60 degree vertical field of view, so the four sides of the cube which form the image-based intermediate representation are each 416 by 421 resolution. The user-side computer receives a complete new intermediate image-based representation from the rendering machine approximately every four seconds. Transmitting this representation uses less than 25% of the 155 Mbit/sec ATM bandwidth. Despite the excessively long interval between updates to the intermediate image-based representation, a user in a headmounted display feels a sense of presence.

[*] Our resolution is actually a hybrid between 320 x 240 and 640 x 480. The final output image resolution is 640 x 480, but the corresponding portion of the intermediate image representation has approximately a 320x240 resolution.

Speedup Techniques: Clipping and Parallelism

We use the [McMillan95a] notation for the warp equations. We do not have the space to explain the equations in detail here, and we refer you to the original paper for a more complete discussion.

$$x' = \frac{ax + by + c\delta + k}{gx + hy + i\delta + m} \qquad y' = \frac{dx + ey + f\delta + l}{gx + hy + i\delta + m}$$

where x , y represent the pixel location in the source image, x' , y' represent the pixel location in the destination image, δ represents the generalized disparity for the source pixel, and a - m represent constants for a given pair of viewpoints and view directions.

An important part of our system's warping implementation is its clipping of regions of the intermediate images which can be guaranteed to fall off-screen. This clipping avoids the warp algorithm's expensive per-pixel calculations for pixels which fall in these regions, thus providing a several-fold performance increase over previous implementations. Our clipping algorithm works as follows: The rendering computer determines the maximum and minimum disparity values for each row of the intermediate image, and passes these values to the user-side computer along with the intermediate image. For a given row in the intermediate image, the maximum and minimum disparity values can be used in conjunction with the location of the row to construct a quadrilateral in the projective 2-space used by the warp algorithm. This quadrilateral (actually a parallelogram) is then clipped against the view frustum in the conventional manner. If the quadrilateral lies entirely outside the view frustum, then we skip the entire row. This case is likely for rows in the one or two sides of the cube which fall "behind" the viewer. If the quadrilateral is partially clipped, then we determine from the clipped polygon's new vertices which portion of the row is potentially visible, and pass only that portion to the per-pixel warper. Finally, if the polygon is entirely "visible", then we treat the entire row of the source image as potentially visible.

A second important component of our warping implementation is its parallelization across three processors. Our serial implementation of the planar-to-planar warp relied on the occlusion-compatible rendering order described in [McMillan95a], but this rendering order presents challenges when parallelizing. Our first attempt at parallelization assigned the different occlusion-compatible regions to different processors, but we got poor load balancing with this strategy because the regions can differ greatly in size. Our current parallel implementation instead uses a z-buffer to resolve visibility, thereby avoiding the constraints imposed by the occlusion compatible-rendering order. We do however pay a price for the Z-buffering – an extra multiply to compute the Z value, and the test/compare/modify cycle for each pixel which is written. We hope to once again return to the occlusion-compatible rendering order, but achieve better load balancing by appropriately subdividing the occlusion-compatible regions.

Image Reconstruction Kernel

McMillan and Bishop's paper does not discuss the implementation of their forward-mapping image reconstruction technique. Our implementation writes pixels to the final image in a splat-like manner. With this technique careful attention must be paid to the size of the splatted pixels if the results are to look acceptable. The further the new viewpoint is from the viewpoint used to render the original image, the more opportunity there is for significant change in pixel size from the original to new images. Deviations from the correct new size show up as holes or blotches in the final image. To avoid these problems as much as possible, we carefully compute the reconstruction kernel size. These computations are actually more expensive than the ones used to determine the location of the splat.

The expression we use for our axis-aligned reconstruction kernel size is:

$$xsize = \left|\frac{\Delta x'}{\Delta x}\right| + \left|\frac{\Delta x'}{\Delta y}\right| \approx \left|\frac{\partial x'}{\partial x}\right| + \left|\frac{\partial x'}{\partial y}\right|,$$

and similarly for ysize. For the partial derivatives we have:

$$\frac{\partial x'}{\partial x} = \frac{\frac{\partial \delta}{\partial x}(c - ix') + a - gx'}{gx + hy + i\delta + m}$$

and likewise for the others.

It is tempting to simplify the computation by assuming $\frac{\partial \delta}{\partial x} = \frac{\partial \delta}{\partial y} = 0$, but this simplification leads to small holes in the destination image when the angle between the viewpoint and the surface changes significantly between the source and destination images. We instead approximate the disparity derivatives by taking the difference of disparity values at adjacent pixels, and use this result for the computation.

Discussion

Improving Quality of Final Images

The most obvious drawback of all *single* source-image perturbed projective warping systems is that cracks appear in the final image when the occlusion in the scene changes. There is no information about surfaces which are occluded in the original image, and when these surfaces become "exposed" some erroneous data must be inserted. In the long term, this problem will probably be the key one that needs to be resolved if image-based rendering techniques are to find widespread application. We currently see two reasonable approaches to this problem -- multiple source images, and intelligent "guesses".

When multiple source images are available, the extra images can be used to fill in the gaps left after warping the first image. An interesting variation on this idea is the multi-valued Z-buffer demonstrated by Nelson Max [Max95]. The extra source images will not necessarily consume the same amount of transmission bandwidth as the first source image, because for most scenes they will be similar enough to the first image that a warp-based compression should be very effective on them The biggest challenge in using multiple source images is the choice of rendering viewpoints for the extra images. These viewpoints might be chosen to bracket the estimated future position of the user, or to be as different as possible (within some constraints) from the first source image.

Another variation on this theme would be to retain "old" source images on the user-side, and attempt to use these images to fill in gaps left by the current image.

A second approach to the occlusion problem is to attempt to use intelligent "guesses" to fill in the gaps. There will usually be two surfaces at the sides of a gap – one is the "front" surface and the other is the "rear" surface. We will typically be able to classify the surfaces at the gap by examining the disparity (and possibly surface orientation) information carried with the pixels. If we fill in the gap with the color of the nearest pixel from the rear surface, we are making the assumption that the rear surface continued behind the front object. In most cases, this assumption will be correct if the gap to be filled is small, and the error in the final image will be much less noticeable.

A second major quality issue is that of aliasing. Because we rely on correct Z-values, we render our intermediate images without super-sampled anti-aliasing. In this context, a simple solution to the aliasing problem is to transmit some representation of the n-samples to the user-side computer (the extra n-1 should compress very well), warp the n-samples, and then perform the super-sampled averaging. This technique should produce a high-quality anti-aliased final image, but it would be of course be computationally expensive.

Compression of Transmitted Data

Although the bandwidth required for our technique is bounded, it is still quite large. To provide the viewer with a 60 degree FOV looking in any direction requires that the transmitted image-based representation contain approximately twelve times the number of pixels required for the final 60 degree FOV. For 640x480 (NTSC) final resolution, the image-based representation thus contains 2.8 million pixels. This large size is mitigated by the fact that an image-based representation is extremely compressible. Conventional image compression techniques should be effective on the color portion of the image, and work by Guenter [Guenter93] implies that the disparity values should be highly compressible as well.

We can further compress our image-based representation by taking advantage of its temporal coherence. Both ends of the transmission link can generate "expected" next images, and only the difference between the expected and actual images needs to be transmitted. Typically the expected image is generated from a block approximation of the flow field which is transmitted across the link -- MPEG works this way for example. Normally this flow field is estimated using correlation techniques, but since our type of system is working from rendered images, it can take advantage of per-pixel depth information in conjunction with the view transform to *compute* the exact flow field. Agrawala et al. [Agrawala95] have described such a technique which uses the resulting flow field to compute accurate 2D transformations for blocks of pixels.

It may be possible to improve on Agrawala's strategy by using an image warp to directly produce the "expected" images (and depth values), bypassing the explicit computation and transmission of a flow field. The resulting technique would be similar in spirit to Guenter's work compressing animation sequences [Guenter93] but would not use object information. This use an image warp is in addition to, but orthogonal to, its use for latency compensation. The two uses complement each other well, because they both require that depth or disparity values be transmitted to the user-side computer.

Warping Speed

The most obvious drawback to our system's current implementation is the speed of the image warp. To get full NTSC resolution with a 20 Hz frame rate would require approximately a 12-fold increase in performance. Our next step towards improved performance is to develop a fixed-point formulation of the algorithm to replace our current floating-point formulation. We have begun work on this task. A fixed-point formulation of the algorithm can be implemented using the new multimedia/graphics instruction set extensions which have begun to appear recently, thus yielding faster software-based implementations of the warp. An integer formulation of the warp will also be amenable to hardware implementation. A hardware assisted implementation of the algorithm will almost certainly be necessary if our technique is to be used in low-end machines such as PC's and set-top boxes.

Prediction Techniques

If we could somehow accurately predict future viewpoints, then any sort of local representation would be unnecessary -- we could instead just render images using the predicted future viewpoints and transmit them to the user side for unmodified display. This technique would eliminate apparent view transform latency completely, although scene change latency would remain. Unfortunately, for the case of a viewpoint controlled by head position, no known predictive tracking algorithm is sufficiently accurate over the necessary prediction interval of hundreds of milliseconds. Furthermore, frequency domain studies of user head motion [Azuma95] argue that no such algorithm will ever exist.

Although predictive tracking is not sufficiently accurate to eliminate the need for a user-side scene representation, it is useful for *estimating* future head positions. So and Griffin [So92] successfully combined predictive head tracking with simple image shifting. We hope to combine it with the more general image warping we use to increase the quality of the warped images. This increase in quality will result from reducing the errors in the warped image which grow as a function of translation distance. Interestingly, the use of image warping with predictive tracking provides us with the opportunity to circumvent one of the "paradoxes" of predictive tracking: As the gains on the predictor are turned up, the mean-squared error goes down, but much of the remaining error is at high-frequencies. These high frequency errors cause "jitter" in the displayed image which is extremely bothersome to users [Azuma95]. The final image-warping step which we propose can eliminate this jitter, while talking advantage of the greater mean-squared accuracy of the high gain.

In our earlier discussion of scene-change latency, we argued that if changes in the *scene* could not be predicted, then there was nothing that could be done about scene-change latency. Let us now consider the case where some predictive information about scene changes is available. We believe that most predictive techniques which have in the past been used at the object level can be extended to the pixel level for use with image-based representations. For example, each pixel in the image-based representation can be assigned a velocity (and possibly acceleration), based on the motion of the underlying object in the scene. The warp algorithm then calculates final pixel positions as a function not just of translation and pixel-disparity, but also as a function of time, pixel-velocity, and pixel-acceleration. Costella has proposed this technique outside the context of perturbed-projective image warps [Costella93], but we believe that it can be effectively integrated with this type of image warp. There also the potential to assign more complex time-dependent behaviors to pixels. In addition to reducing apparent scene-latency, these techniques could also greatly reduce the required rate at which local-scene representations are transmitted from the rendering engine to the user-side computer. Finally, these techniques could be integrated with the the use of image warping for compression.

Non-Remote Systems

While we have concentrated on the use of image warping to compensate for remote display latency, we also believe that these techniques also have significant promise for delay compensation in entirely local systems. Local systems have less latency, and so the argument that an image-based technique needs to properly handle translations as well as rotations is not as strong. But, if our techniques are used to reduce the rate at which the rendering engine needs to generate frames, then the latency again becomes large. For local display, our technique thus presents an alternative to the hybrid address recalculation + priority rendering scheme proposed by Regan and Pose [Regan94]. The approaches differ in the types of artifacts they would produce -- priority rendering has the potential for artifacts at "priority boundaries", while our approach produces occlusion artifacts. It might make sense to combine an image warp with priority rendering in an effort to get the best of both approaches.

Acknowledgements

This research was supported primarily by DARPA contract #DABT 63-93-C-0048. This report has been approved for public release—distribution unlimited. Leonard McMillan was partially supported by a Division, Inc. fellowship. Additional support for this work was provided by the NSF/ARPA Science and Technology Center for Computer Graphics and Scientific Visualization (Contract #ASC-8920219).

We thank David Ellsworth for his advice during our first effort to implement this system. We also thank Fred Brooks, Henry Fuchs, Steve Molnar, Turner Whitted, Nick England, Andrei State, David Harrison, Peggy Wetzel, Marc Olano, Carl Mueller, Bill Dally, and Dan Aliaga, who all provided useful advice and/or assistance.

References

[Agrawala95] M. Agrawala, A. C. Beers, and N. Chaddha, "*Model-Based Motion Estimation for Synthetic Animations*," **ACM Multimedia 1995**, (San Francisco, CA) November 5-9, 1995.

[Azuma95] R. Azuma and G. Bishop, "A Frequency-Domain Analysis of Head-Motion Prediction," Proceedings of SIGGRAPH '95, In Computer Graphics, Annual Conference Series 1995, pp.401--408.

[Burbidge89] D. Burbidge and P. M. Murray, "*Hardware Improvements to the Helmet Mounted Projector on the Visual Display Research Tool (VDRT) at the Naval Training Systems Center*," **Helmet-Mounted Displays**, Jerome T. Carollo, Editor, Proc. SPIE 1116, pp. 52-60 (1989).

[Costella93] J. P. Costella, "*Motion Extrapolation at the Pixel Level*", unpublished paper available from http://www.ph.unimelb.edu.au/~jpc., Jan 14, 1993.

[Gossweiler93] R. Gossweiler, C. Long, S. Koga, and R. Pausch, "*DIVER: a Distributed Virtual Environment Research*

Platform," **1993 IEEE Symposium on Research Frontiers in Virtual Reality.**

[Guenter93] B. K. Guenter, H. C. Yun and R. M. Mersereau, "*Motion Compensated Compression of Computer Animation Frames*," **Proceedings of SIGGRAPH '93**. In Computer Graphics, Annual Conference Series 1993, pp. 297-304.

[Katz94] W. Katz, "*Military Networking Technology Applied to Location-Based, Theme Park and Home Entertainment Systems*," **Computer Graphics**, Vol. 28, No. 2, May 1994.

[Max95] N.Max and K. Ohsaki, "*Rendering Trees from Precomputed Z-Buffer Views,*" **6th Eurographics Workshop on Rendering**, (Dublin, Ireland) June 1995, pp. 45-54.

[McMillan95a] L. McMillan and G. Bishop, "*Head-Tracked Stereo Display Using Image Warping*," **1995 IS&T/SPIE Symposium on Electronic Imaging Science and Technology**, SPIE Proceedings #2409A, (San Jose, CA) February 5-10, 1995, pp.21-30.

[McMillan95b] L. McMillan, "*A List-Priority Rendering Algorithm for Redisplaying Projected Surfaces*," UNC-CH Computer Science Technical Report TR95-005, University of North Carolina, 1995.

[Regan94] M. Regan and R. Pose, "*Priority Rendering with a Virtual Reality Address Recalculation Pipeline*," **Proceedings of SIGGRAPH '94**. In Computer Graphics, Annual Conference Series 1994, pp.155-162.

[So92] R. H. Y. So and M. J. Griffin, "*Compensating Lags in Head-Coupled Displays Using Head Position Prediction and Image Deflection*," **Journal of Aircraft**, Vol. 29, No. 6, Nov.-Dec. 1992.

[Tharp92] G. Tharp, A. Liu, L. French, S. Lai, and L. Stark, "*Timing Considerations of Helmet Mounted Display Performance*," **Human Vision, Visual Processing, and Digital Display III**, B. Rogowitz, Editor, Proc. SPIE 1666, pp. 570-576 (1992).

[Wolberg90] G. Wolberg, **Digital Image Warping**, IEEE Computer Society Press, Los Alamitos, California, 1990.

[Zyda93] M. Zyda, D. Pratt, J. Falby, P. Barham, and K. Kelleher, "*NPSNET and the Naval Postgraduate School Graphics and Video Laboratory*," **PRESENCE**, Vol. 2, No. 3, Summer 1993, pp. 244-258.