The Development of a Hybrid Virtual Reality/Video View-Morphing Display System for Teleoperation and Teleconferencing

by

William E. Hutchison

Master of Science in Aerospace Engineering
The University of Texas at Austin, 1998.

Submitted to the System Design and Management Program in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

June 2000

Signature of Author: _____
System Design and Management Program
May 5, 2000

Certified by: _____
Dr. Charles M. Oman
Director, Man-Vehicle Lab,
Senior Lecturer, Dept. of Aeronautics and Astronautics
Thesis Supervisor

Accepted by: _____
Thomas A. Kochan
LFM/SDM Co-Director
George M. Bunker Professor of Management

Accepted by: _____
Paul A. Lagace
LFM/SDM Co-Director
Professor of Aeronautics & Astronautics and Engineering Systems

The Development of a Hybrid Virtual Reality/Video View-Morphing Display System for Teleoperation and Teleconferencing

by

William E. Hutchison

Submitted to the System Design and Management Program
on May 5, 2000 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Engineering and Management

Abstract

The goal of this study is to extend the desktop panoramic static image viewer concept (e.g., Apple QuickTime VR; IPIX) to support immersive real time viewing, so that an observer wearing a head-mounted display can make free head movements while viewing dynamic scenes rendered in real time stereo using video data obtained from a set of fixed cameras. Computational experiments by Seitz and others have demonstrated the feasibility of morphing image pairs to render stereo scenes from novel, virtual viewpoints. The user can interact both with morphed real world video images, and supplementary artificial virtual objects ("Augmented Reality"). The inherent congruence of the real and artificial coordinate frames of this system reduces registration errors commonly found in Augmented Reality applications. In addition, the user's eyepoint is computed locally so that any scene lag resulting from head movement will be less than those from alternative technologies using remotely controlled ground cameras. For space applications, this can significantly reduce the apparent lag due to satellite communication delay. This hybrid VR/view-morphing display ("Virtual Video") has many important NASA applications including remote teleoperation, crew onboard training, private family and medical teleconferencing, and telemedicine. The technical objective of this study developed a proof-of-concept system using a 3D graphics PC workstation of one of the component technologies, Immersive Omnidirectional Video, of Virtual Video. The management goal identified a system process for planning, managing, and tracking the integration, test and validation of this phased, 3-year multi-university research and development program.

Thesis Supervisor: Charles M. Oman
Title: Director of Man-Vehicle Lab, Senior Lecturer, Dept. of Aeronautics and Astronautics

Supported in part by NASA JSC Space Human Factors Engineering Program Grant NAG9-1004

If one advances confidently in the direction of his dreams,

and endeavors to live the life which he has imagined,
he will meet with a success unexpected in common hours.
He will put some things behind, will pass an invisible
boundary; new, universal, and more liberal laws will begin to
establish themselves around and within him; or the old laws be
expanded, and interpreted in his favor in a more liberal sense,
and he will live with the license of a higher order of beings. In
proportion as he simplifies his life, the laws of the universe will
appear less complex, and solitude will not be solitude, nor
poverty poverty, nor weakness weakness. If you have built
castles in the air, your work need not be lost; that is where they
should be. Now put the foundations under them.

-Henry D. Thoreau, Walden

Acknowledgements

I would like to thank the following individuals and groups for their assistance during the making of this thesis:
My family for their enthusiastic support and unyielding faith,
Dr. Charles M. Oman for giving me the opportunity to work in the Man-Vehicle Lab,
Dr. Andy Beall for his invaluable advice and friendship,
Ben Sachtler for his insights and suggestions regarding the programming implementation,
Dr. Andy Lui for his review of my thesis and his assistance with the programming implementation,
My friends at MIT, Brian Ippolito, John Johnson, Tom Pelland, and Dave Tonaszuck for offering their friendship, their humor and, their support,
and the Members of the Man-Vehicle Lab.

Partial funding for this work was provided by NASA grant NAG9-1004.

Table of Contents

Table of Figures

List of Tables

Introduction

In its fully developed form, the Virtual Video system combines three distinct technologies that together will significantly improve traditional tele-application displays. The first technology, Immersive Omnidirectional Video, extends the desktop-PC static panoramic image-viewer concept (e.g., Apple QuickTime VR, IPIX) to an immersive, real-time viewing environment. With this capability, an observer wearing a head-mounted display can make free head movements while viewing dynamic scenes rendered in real time using

spherically captured video data obtained from a set of fixed cameras. When the second technology, Video View Morphing, is integrated within the Immersive Omnidirectional Video platform, the user will not only be able to view real-time stereo video images of an entire spherical scene from a particular camera vantagepoint but may also observe virtual stereo images between and perhaps slightly outside the arc connecting two or more of these physical camera viewpoints. The final technology, Augmented Reality, enhances the Virtual Video display further by blending computer-generated images into the video scene. The user can now interact with scenes that contain “HUD-like” information, for instance, which affords enormous advantages over a simple immersive video display.

Capabilities

Since its inception, the Virtual Video system has been focused on improving or mitigating the problems associated with human performance and presence in space. To gain better insight into the capabilities of the Virtual Video system in its application to human spaceflight, a number of far-reaching and in some instances near term NASA-

specific scenarios are presented below. The first few scenarios address NASA's most immediate needs and do not rely on a fully developed view-morphing capability:

Teleoperation Scenario

Operating the robotic arm (RMS) during shuttle operations, and now during ISS assembly, is a perceptually demanding task. When single views are available of the assembly area, even highly trained and experienced operators will inevitably make mistakes in judging distances to and clearances between target objects. Using "Virtual Video", the input from one or more stationary cameras can be processed in real time allowing the operator to have stereoscopic views from arbitrary moving vantage points. In some cases, these may include vantage points that are physically impossible to place a real camera. The stereoscopic capability will be used naturally by the operator to convey a much more veridical impression of the physical layout that will significantly reduce planning time and increase performance accuracy.

Virtual Cockpit Scenario

In advanced hypersonic vehicles and in some military applications, windowless cockpits may be required. Synthetic vision systems are already under consideration for these applications. "Virtual Video" techniques could also be applied to synthesize a view using sensors and data streams from both inside and outside the cockpit.

Crew On-board Training Scenario

Presumably pieces of equipment will have to be delivered to the ISS that have not been previously examined by the astronauts aboard. An astronaut could virtually "stand" next to a ground expert while he gave a training demonstration of the equipment's operation. While they talked with one another, the astronaut could alternate between looking directly at the expert and examining closer certain features of the equipment or virtually "follow" him as he walks around the equipment. In 1-G training applications, the "Virtual Video" system could be used to simulate the viewpoints achievable in a weightless environment.

After the three Virtual Video technologies are completely realized, another possible application is as follows:

Video-Teleconferencing Scenario

Using the "Virtual Video" system, astronauts aboard the ISS will be able to experience a home-like situation such as "dinner time" that is more compelling than a video-teleconferencing session for a minimal incremental equipment cost. An astronaut will be able to look around his family's dining room and interact with his family as if he were sitting at the dinner table. The family will be able to maintain eye contact with the astronaut while they communicate. The astronaut could also virtually "walk" around the family's dining-room table if he so desired.

Synopsis

This thesis technically defines the Virtual Video concept and describes a phased, 3-year multi-university research and development program to develop and demonstrate the component technologies. Initial work has already begun under NASA grant NAG9-1004. The work for this thesis will comprise a small part but will nonetheless lay the foundation for the entire Virtual Video project. The outline of this thesis is described below:

Technical Content: The technical objectives of this thesis are two-fold: first, to provide both a brief engineering description of the entire, fully mature Virtual Video system and a more detailed account of one of its component technologies, Immersive Omnidirectional Video. The second objective is to demonstrate a proof-of-concept system on a PC platform of the Omnidirectional Video technology. Specifically, this work will incorporate the use of a wide-angle reflector to support immersive (180° by 180°) real-time viewing from a fixed camera. Building the proof-of-concept demonstrator will serve as an "experimental" endeavor whose aim is to thoroughly explore the major practical limitations of the omnidirectional video technology.

Management Content: Another goal of this thesis is to identify a system process for planning, managing, and tracking the integration, test and validation of the Virtual Video's program of research and development. This system plan will broadly outline the required development steps for the entire Virtual Video system and will more specifically detail the steps necessary to develop the Immersive Omnidirectional Video technology.

This paper will describe and discuss the development of the entire Virtual Video system in sections 2 and 3, respectively. Then the focus will center on the background and development of the Immersive Omnidirectional Video technology in sections 4 and 5, respectively. Finally, conclusions will be presented in section 6.

Virtual Video System Description

What is innovative about the Virtual Video system is its integration of the three technologies: Video View Morphing, Immersive Omnidirectional Video, and Augmented Reality. In combination, these technological advances will greatly enhance traditional tele-application displays. The development of the Virtual Video system will be complete when the following technological hurdles are solved:

- Video View Morphing — develop view-morphing technology to allow rendering of continuous virtual viewpoints from a limited number of physical camera vantagepoints (see Fig. 1 below (Seitz, 1998)).

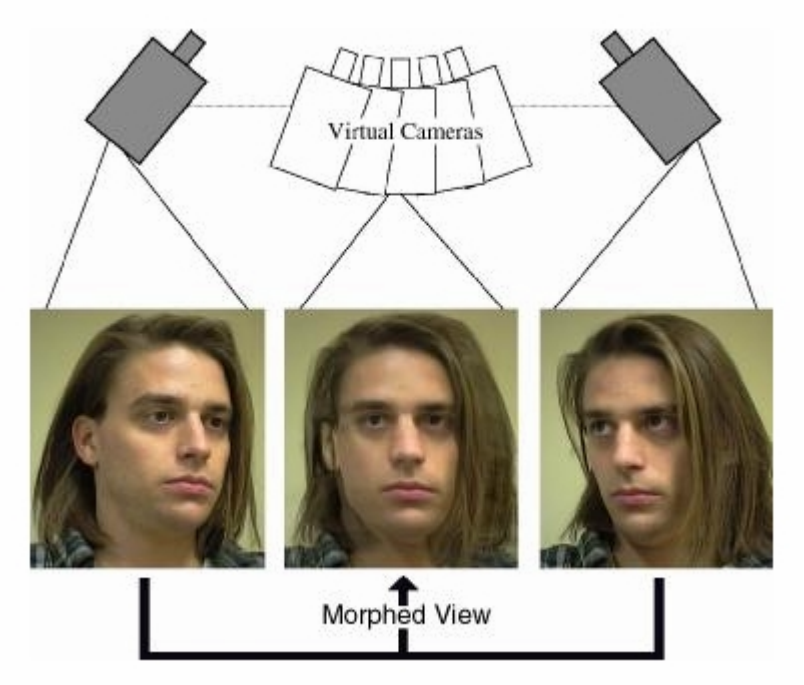


Figure 1. View morphing. *Two images of an object taken from two different viewpoints produces the illusion of physically moving a virtual camera.*

- Immersive Omnidirectional Video — digitally simulate a pan/tilt/roll camera by processing a wide field-of-view input image (e.g. fish-eye lens) in real time much like

Apple's QuickTime VR desktop panoramic image viewers or IPIX omnidirectional image viewers but extended to operate on dynamic video scenes. Figure 2 demonstrates this technology by taking a wide field-of-view image that initially appears as a donut-shaped scene and is undistorted using software into a 360° window that can be navigated around using a head-mounted display (HMD) and the pan and tilt capabilities of a head tracker.

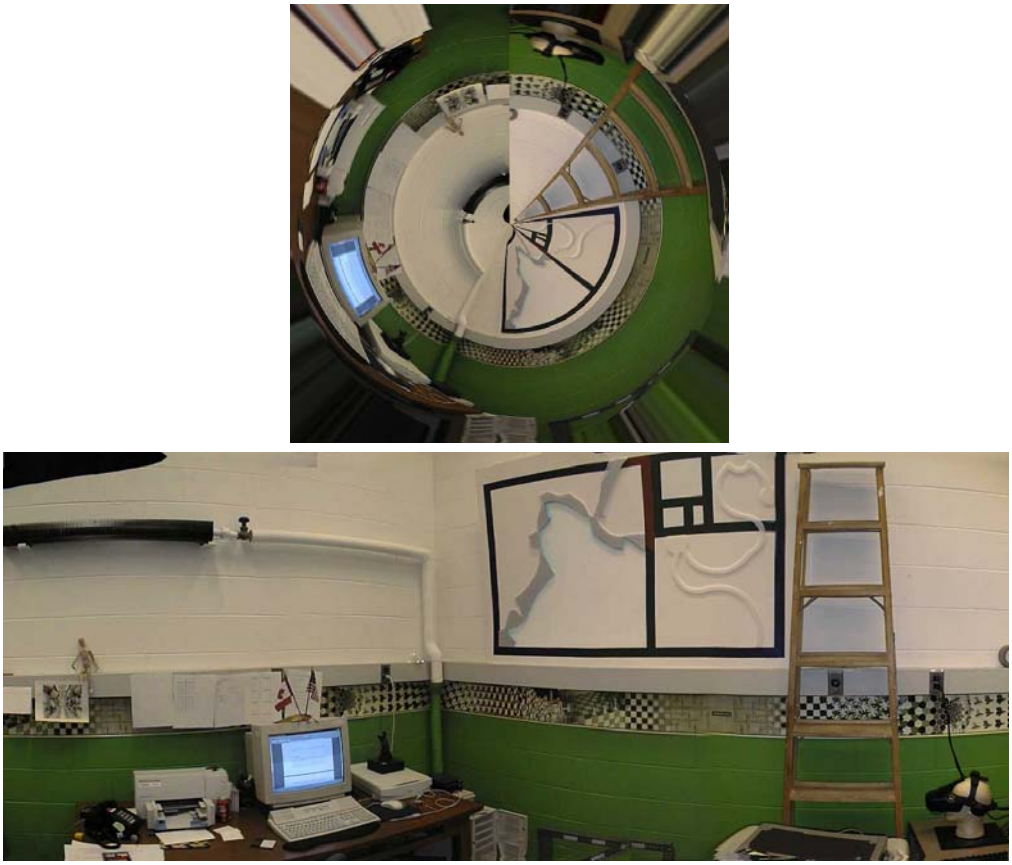


Figure 2. Omnidirectional Video. *The top image is the scene as it appears through the fish-eye lens. The bottom image is the same image, however, it has been undistorted by software. The particular portion of this undistorted 360° image is displayed to the HMD based on the head direction sensed by the head tracker.*

- Augmented Reality — combine video and computer generated imagery superimposed in three dimensions. For the video-teleconferencing application, a practical problem is that a portion of the HMD-wearer's head is obscured. This prevents eye contact

and impairs two-way social interaction. However, it is possible to remedy this by using an Augmented Reality technique to make the HMD effectively disappear (see Fig. 3). The solution is to create a virtual cylindrical visor which follows the HMD wearer's head with a head tracker and an eye movement monitor in the HMD to determine the wearer's angle of gaze. The virtual visor is then mapped over the HMD portion of the video with a pre-recorded texture showing the wearer's upper head and moving eyes.

Figure 3. Teleconferencing Augmented Reality Application. *Rendering an overlay of a previously photographed image of the user's eyes in 3D that will effectively mask the image of the HMD.*

Virtual Video System Advantages

By extending QuickTime VR-like omnidirectional viewers to operate upon real-time video, Virtual Video effectively creates a remotely servoed pan/tilt camera with no moving parts and no lags other than those inherent in the computational algorithms. For space applications, this can greatly reduce the apparent lag due to satellite communication delay. By incorporating view-morphing technology, Virtual Video will further simulate limited ranges of camera translation again without physically moving the camera. Finally, Virtual Video's method for capturing, undistorting, and morphing real video images will provide a robust means for superimposing computer graphic images

creating an “Augmented Reality” (AR) display. Conventional Augmented Reality systems use see-through head-mounted displays (HMD) and therefore must accurately align computer graphic images (CGI) upon the external scene. This method is prone to dynamic misalignments due to delays and to static inaccuracies due to head-tracking errors. By capturing the external scene using physically stationary video cameras and rendering the superimposed images in the same coordinate system, the registration of the images no longer depends on the accuracy or speed of the head-tracking system.

In summary, the “Virtual Video” system and its component technologies have numerous advantages and overcome a number of limitations:

- Video View Morphing provides the capability to“virtually” fly around a scene.
- Immersive Omnidirectional Video enables a 360° panoramic view of a scene without moving the camera—thereby significantly reducing any remote-control camera lags.
- Augmented Reality provides the ability to insert computer graphic images for scene enhancement and can be accomplished with greater accuracy than current systems.

Virtual Video System Development

Virtual Video will be a novel, photo-realistic, immersive augmented-reality display system that can be used in teleoperation, teleconferencing, and telemedicine for a variety of space, military, and commercial applications. The ideas for this system began with an effort to extend the capabilities of the existing virtual reality (VR) equipment at the Man Vehicle Lab (MVL) at MIT. The formal work for this system started with the preparation and submittal of a proposal by MVL to NASA for a NRA-98-HEDS-04 research grant in February 1999 (see Appendix A). Before the response to the HEDS-04 proposal was received, the process of patenting the entire Virtual Video concept was initiated. The unique combination of the three technologies for the Virtual Video system meets the novel, useful, and nonobvious criteria for a patent (for a comparison and description of like systems see Table 1 in the next section). A literature review was conducted along with a preliminary investigation of systems similar to Virtual Video either currently in development or already in the marketplace. Afterward, an application was submitted to the Technology Licensing Office (TLO) at MIT, however, further processing of the patent is pending once the Virtual Video system has been reasonably reduced to practice.

In August, 1999, the proposal received a high scientific merit score from the HEDS-04 review board, however, it was not funded. Nonetheless, the system is currently being considered by NASA to be used as a supplementary situational awareness/synthetic vision capability for the Emergency Crew Return Vehicle (ECRV, also called the X-38)

during vehicle separation from the ISS and during its parachute unreefing and landing site evaluation phases of operation.

Virtual Video System Development Plan

The high-level technical goals of the entire Virtual Video project are:

- To develop the three component technologies of "Virtual Video": real-time Video View Morphing, Immersive Omnidirectional Video, and Augmented Reality techniques
- To demonstrate a proof-of-concept system on a PC platform similar to the NASA Human Research Facility computer workstation on the International Space Station

In order to accomplish these goals, a 3-year development plan for the Virtual Video system was formulated and is shown in Fig. 4 below. The proposed work for the Virtual Video project has been divided between laboratories at MIT and Carnegie Mellon University (CMU) reflecting the technical strengths of each group. Specifically, the Video View-Morphing technology will be developed by Seitz at CMU while both the Immersive Omnidirectional Video and Augmented Reality technologies will be developed at MIT.

The developments for each technology require several steps, some of which can proceed concurrently. The individual steps taken or yet to be accomplished for the Immersive Omnidirectional Video technology are detailed later in this thesis. Whereas, the other two technologies, Video View Morphing and Augmented Reality, are described in the HEDS-04 proposal in the Appendix. As portions from both the Video View Morphing

and Augmented Reality technologies become developed they will be integrated with the Immersive Omnidirectional Video platform at several milestones. These integration steps and their associated system evaluation tests are also described later in this thesis.

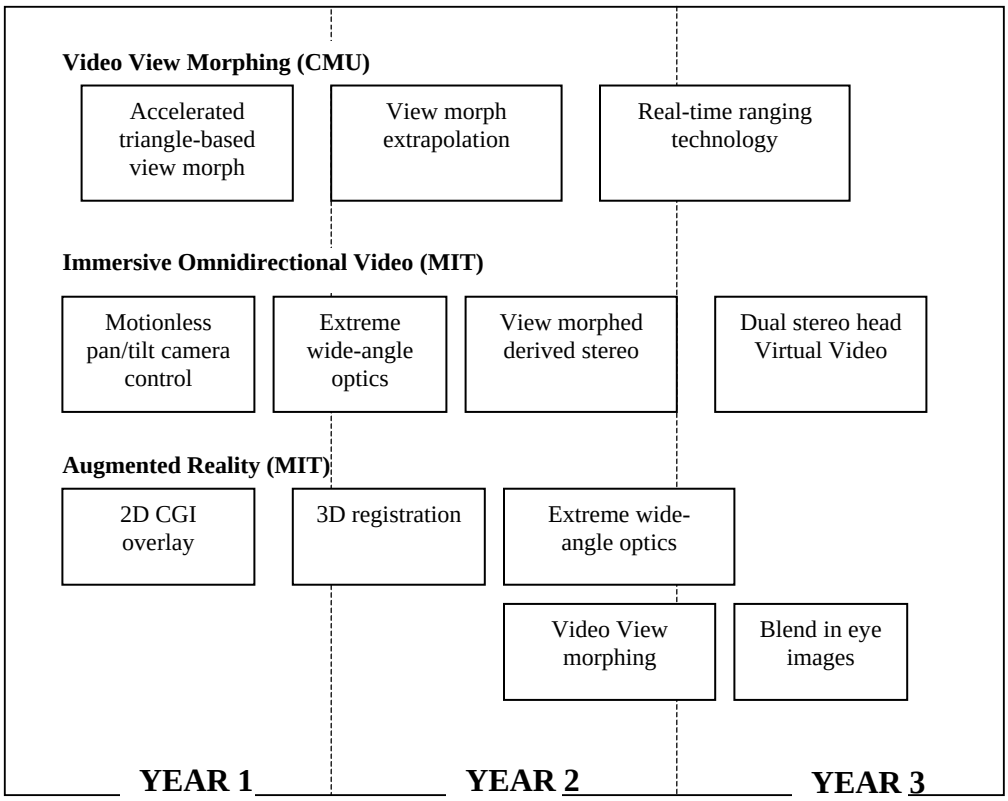


Figure 4. "Virtual Video" Plan of Work

Immersive Omnidirectional Video Background

Problem Statement

Historically, traditional tele-application displays have had numerous drawbacks. Typically, the user views a video monitor displaying the output of a video camera at the other end of the application. The camera's point-of-view is either stationary or can be pan/tilt controlled via a remotely operated servomotor. When available, the servomotor is commonly controlled by the user with a joystick or some other similar pointing device. Mechanical impedance and communication delays make quick camera motions difficult to control (Nayer, 1998; Zimmerman, 1993). Under the best of conditions, it remains difficult for the user to spatially integrate the changing camera perspective into a contiguous percept of the surrounding space captured from the opposite end of the tele-application system (Rock, 1981). Other difficulties are specific to the particular tele-application. As an example, users of video teleconferencing often complain of a lack of proper eye-gaze information that results from having the view screen and input camera at different angular positions; some even believe that the resulting negative interpretations may undo the attempted gains of adding the video channel (Noro, Kawai, and Takao, 1996).

Donning a head-mounted display (HMD) is a potential method of overcoming the difficulties of integrating multiple viewpoints by allowing the user to control the camera's pan/tilt with natural head movements (Sutherland, 1968). Not only does this simplify the camera control issues, but studies have shown that if a user indeed feels immersed in the visual environment this may consequently have a positive impact on

performance. NASA has contributed to many of the early pioneering efforts to "develop a new kind of interface that would be very closely matched to human sensory and cognitive capabilities" (Fisher, 1990). With regards to tele-applications, however, the deleterious effect of control lags between the sensed head motions and the concomitant scene changes (via a remotely servoed camera) are exacerbated in part because humans are accustomed to an extremely stable visual world provided by the vestibular-occular reflex (Young & Stark, 1963). Concerns over these control lags have driven the development of immersive but physically large, back-projected panoramic displays such as the CAVE system (Cruz-Neira, Sandin, & DeFanti, 1993) or the Virtualized Reality system (Kanade, 1995 & 1997). While this technology overcomes several of the concerns raised above, the physical dimensions and lack of portability of CAVE-type systems make them impractical for many applications such as NASA flight operations.

Time delays between user movements and visual feedback can have deleterious effects on performance. Delays between a user's movement and the visual feedback in a tele-application display can arise from a number of sources, including transmission time for communication with a remote site, sensor delays used in head or hand tracking, and rendering delays of the graphics processor. The effect of delay on manual tracking performance has been thoroughly investigated (Poulton, 1974; Wickens, 1986); as has its effect on remote manipulation in teleoperator systems (Ferrell, 1965). Held, Efstathious, & Greene (1966) conducted an adaptation experiment examining the effects of prism displacements and visual feedback delays; in their study, they found that delays as short as 300 ms eliminated adaptation to the displacements.

There have been several recent advances in optical, omnidirectional imaging devices that have been combined with the QuickTime VR-like technology to provide the ability to digitally synthesize pan/tilt effects of a remotely mounted camera (Nayar, 1998; Zimmerman, 1993). Furthermore, these techniques are ideally suited for integration with an HMD immersive tele-application display and this immersive application, in essence, describes the Immersive Omnidirectional Video technology.

In order to test whether this particular application is a viable and practical concept, the technical scope of this study and report are restricted to the demonstration of a proof-of-concept system of the Immersive Omnidirectional Video technology. Restated, this omnidirectional video capability will address the desire for an immersed tele-application environment controlled by head movements without an excessive control lag. Furthermore, the Immersive Omnidirectional Video system will render this environment with real-time video scenes that are both stereoscopic and photorealistic.

Review of Relevant Prior Art

Traditional cameras provide photographs with a limited field of view of a particular scene being photographed. Take for example the famous Zapruder film of the assassination of JFK. The sequence of frames of the movie shows only the car and its occupants. If the movie had recorded a visual image of the entire scene, many issues, such as whether or not all of the shots originated from the book depository building,

could possibly be resolved. It is thus desirable to provide a camera with a wider field of view.

A number of multimedia development packages, such as Apple Corporation's QuickTime VR (Chen, 1995), are designed to combine multiple still photographs and digital still images of a particular scene into a single seamless wide-angle panoramic photograph or digital image. While these types of systems work only with static, pre-recorded images and, hence, are inappropriate for real-time tele-operation applications, they are, nonetheless, the first desktop attempts to create panoramas that replicate the virtual reality immersive experience. As an example, the QuickTime VR system's process (Chen, 1995) begins with a camera used to shoot a panorama of multiple images taken as the camera is rotated about its nodal point (i.e., the optical center) of the lens and in which the frames of the photos overlap slightly. Software called a "stitcher" automatically blends the individual photos together to make a seamless 360-degree view. User-selected portions of this coherent panoramic image are warped by a software based, two-pass algorithm and then reprojected onto a cylindrical shape to form an environment map (Chen & Miller, 1995). Rendering the environment map creates a novel, perspective-correct planar view.

The earliest group of prior art systems to attempt to develop real-time, dynamic equivalents of a mechanical pan, tilt, zoom, and rotation camera viewing system with no moving mechanisms was developed by TeleRobotics International Inc.. The first of these systems captures a 180-degree hemispherical image using a fish-eye or hemispheric

lens (Zimmermann, 1993). A CCD camera captures image data from the lens and the image data is processed to "flatten it out." After this image data has been transformed, it can be accessed electronically. A user can zoom into a particular portion of the image and view only that portion in an expanded format. While this system is extremely capable of providing perspective-corrected views of selected portions of a hemispherical view utilizing no moving parts, the camera requires complex circuitry for implementing a mathematical transformation.

Two other systems from TeleRobotics International Inc. provide perspective and distortion corrected views of a selected portion of a field of view using a motionless camera. The first system incorporates an endoscopic arrangement in which a video camera is attached to a medical device such as a laparoscope or a cytoscope (Kuban et al., 1994a). This system is typically utilized in internal or external imaging for industrial inspection. The second system's motionless camera concept is similar to the endoscopic adaptation, however, its design and intent is to enhance surveillance systems (Kuban et al., 1994b).

Perhaps the two most relevant prior art systems to the Immersive Omnidirectional Video technology is the Cyclovision system developed by Nayar (1998) and the Panospheric Imaging (PI) system developed by Bogner (1995) with the Defence Research Establishment Suffield (DRES). Both the Cyclovision system and the PI system are architecturally quite similar to the omnidirectional video technology. However, in place of the projection lens, the Cyclovision system employs a particular type of parabolic

reflector in which the focus point is coincident with a single viewpoint positioned to orthographically reflect an image of a substantially hemispherical scene. An orthographic image is a projection of a single view of an object (as a view of the front) onto a drawing surface in which the lines of projection are perpendicular to the drawing surface. The use of orthographic reflection from a single viewpoint is an advantageous feature because reflection from any curved surface that provides perspective projection typically results in multiple viewpoints. Multiple viewpoints cause significant distortion and require complex manipulation and translation of the image to reconstruct the scene as viewed from a single viewpoint. Therefore, as a result of the orthographic reflection and the one-to-one correspondence between the reflector and the image sensor, no image reconstruction or complex frame transformation is required.

The projection lens for the Panospheric Imaging system is a sophisticated optical device consisting of reflective elements (convex spherical mirrors, annular conic mirrors) and refractive elements (fish-eye lens, wide-angle optics) that captures 50% or more of the total spherical field available. The PI system also utilizes an electronic mechanism, called an Image Transform Engine (ITE), which references lookup tables to recover the distorted input image. It seems, however, based on the most recent publications, that the PI system does not have video stereo capability nor does it include either the Video View Morphing or Augmented Reality technologies.

For reference purposes, a comparison of the features incorporated in the Virtual Video system and other prior art systems is summarized in Table 1 below.

Table 1. Features Comparison between Virtual Video and Other Prior Art Systems

	Virtual Video	QuickTime VR	Cyclovision	Panospheric Imaging
Dynamic, Real-time	✓		✓	✓
Immersive	✓			✓
360° Spherical View	✓			✓
Stereo Video	✓			
Video View Morphing	✓			
Augmented Reality	✓			
Small size & portable	✓		✓	✓

Immersive Omnidirectional Video Development

Hardware Operation and Development Description (Work Accomplished Thus Far)

At MIT, a simple, proof-of-concept development platform consisting of a 3D graphics PC workstation, a wide-angle projection lens, a video camera, and a low-cost frame capture board has been constructed for testing the Immersive Omnidirectional Video technology. A system diagram and associated photographs are shown in Figs. 5 and 6, respectively, while system technical specifications are listed in Table 2. The video camera, preferably of high-resolution, is directed toward a real-world scene. A suitable camera is the commercially available Cannon Hi8 Camcorder with its standard NTSC video analog output. For the wide field-of-view projection, a convex, spherical/parabolic shaped reflector is mounted to the camera to reflect an image of a substantially hemispherical scene. A "fish-eye" lens was initially considered since it has a short focal length, which enables the field of view to be as large as a hemisphere. The use of such lenses proved to be difficult, however, because they are significantly more complex and difficult to construct (requiring as many as seven to ten lens elements) than the spherical reflector. The output of the video camera is an analog signal which is provided to a real-time digitizing board, commonly referred to as a "frame grabber," which is a commercially available NTSC video analog-to-digital signal converter. In the present instantiation of this proof-of-concept system, an acceptable frame grabber is the Hauppauge WinTV. Other digitizing board alternatives were considered such as the high-end Matrox Meteor

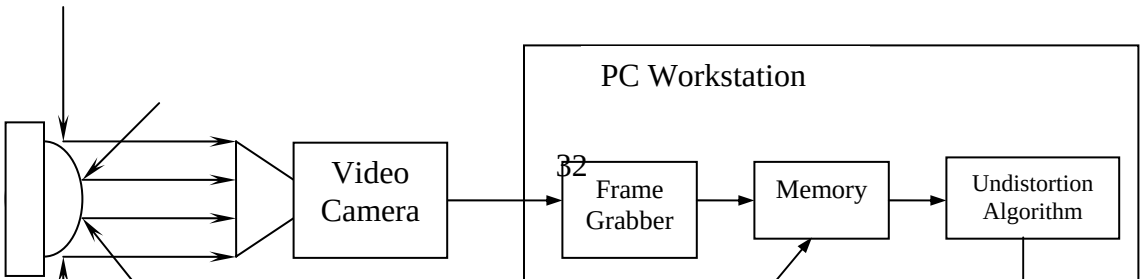


Figure 5. Immersive Omnidirectional Video Architectural Platform

II but the WinTV was chosen for practical and economic reasons. The digitized video signal is thus transferred through the digitizing board (typically but not necessarily at 30 frames per second) into memory of the computer so that portions of the video picture can be randomly accessed by the computer's microprocessor at any time. The computer should be capable of processing video information at an acceptable rate (a Pentium Pro 200 MHz microprocessor or greater is sufficient).

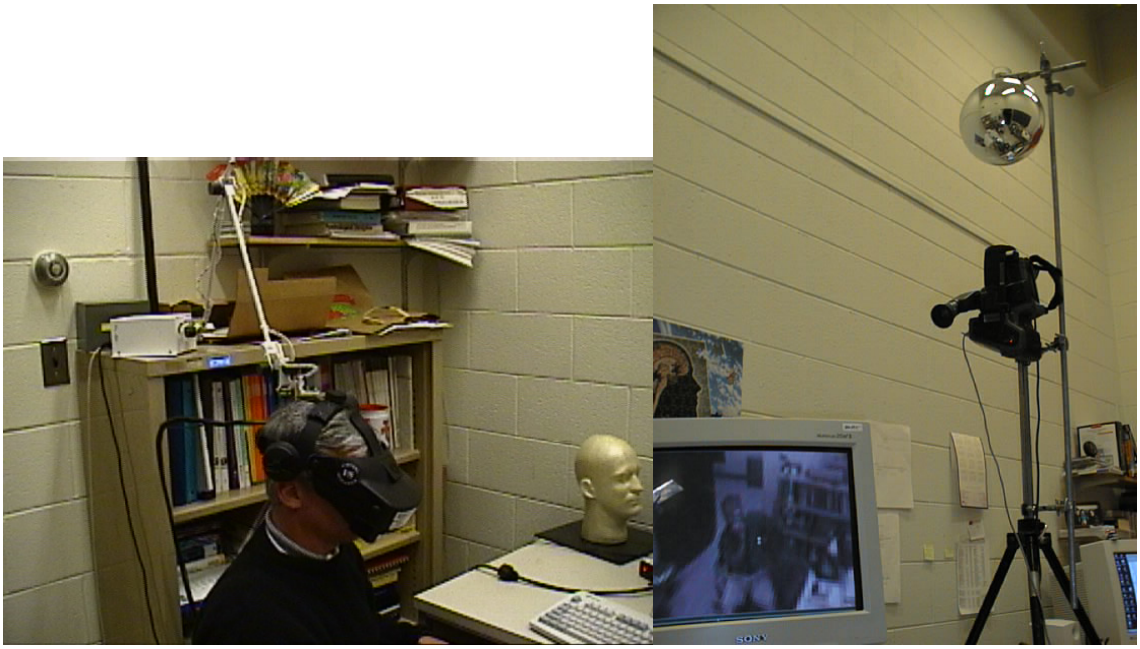


Figure 6. Immersive Omnidirectional Video System Photos. *The photograph on the left depicts the HMD on the user's head and head tracker attached to the HMD, while the photo on the right displays the arrangement of the video camera below the spherical mirror.*

The spherically shaped reflector causes the video output signal from the camera to be optically warped in a non-linear manner (see Fig. 7). Before a user can comfortably view the image, perspective-correcting measures must be taken. A number of approaches for developing the corresponding undistortion algorithms to support immersive (180° by 180°) real-time viewing from a fixed camera were considered. Initially, a spherical lens polygon was created using the commercial 3D Studio Max whose geometrical

Figure 7. Optically Distorted View seen from Camera

properties matched those of the reflective mirror. The spatial coordinates of the simulated lens would be used to produce a spherical polygon. However, the process of obtaining the coordinates and mapping the video texture onto this spherical object proved to be problematic. A straightforward yet more complicated approach would mathematically transform the distorted image. Nevertheless, the simplest and most effective method is to map the video image textures onto a substantially hemispherical object (in the current system, this is a cup-shaped polygon with a hole at the bottom) simulating an eyepoint at the reflective mirror's center and, in turn, indirectly undistorting the video image. The natural, spherical perspective images also offer a more compelling immersive experience.

The dewarping software is stored in memory and is applied to the video signal by the microprocessor. The stored video signal is then transmitted from memory to a special purpose "graphics accelerator" board capable of higher-order polynomial transformations for texture warping and interpolation—an excellent example, and one used in this system, is the Nvidia RIVA TNT graphics accelerator board. The perspective-corrected video signal is next transmitted to a video output stage, such as a standard VGA card, and from there displayed through a suitable head-mounted display (the current system uses a Virtual Research V8 with built-in internal eye-tracking capability, see Fig 8). An input device, such as a headtracker (which senses the head movements of a user wearing a HMD) transmits position information through a suitable input port, such as a serial or parallel port, to the microprocessor to control the portion of stored video that is selected, dewarped, and displayed. The head-tracking device (a mechanical Shooting Star tracker, see Fig. 8) for this system also transmits pitch/roll/yaw information to the microprocessor so that the user may control the orientation of the dewarped video signal. Theoretically, the headtracker may also allow the user to magnify a selected portion of the video input, constrained only by the resolution of the camera.

Figure 8. Virtual Research V8 HMD and Shooting Star Head-Tracker Close-up

Table 2. Immersive Omnidirectional Video System Component Specifications

COMPONENT	Specification
Canon 8mm Camcorder	
Resolution:	NTSC analog video (250-400 lines)
PC Workstation	
Motherboard/Processor:	200 MHz Pentium Pro
Memory:	128 Mb RAM
Hauppauge WinTV Frame Grabber	
Operating system/Host bus/API type:	Windows 98/PCI/Video-For-Windows (VFW) compatible
VGA resolutions supported:	Up to 1280 x 1024
Capture rate:	30 frames/sec
Nvidia RIVA TNT graphics accelerator board	
Fill rate:	180 Mpixels/sec
Virtual Research V8 HMD	
Variable resolution	640 x 480 per eye
Refresh rate	60 Hz
Shooting Star Head Tracker	
Type:	Mechanical
Degrees of Freedom:	6 DOF

Software Operation and Development Description (Work Accomplished Thus Far)

The WinTV frame grabber required the development of a software interface (commonly referred to as an API) to control the processing of the video frames once they arrived in the workstation's memory. The driver for the WinTV board is Video-For-Windows (VFW) compatible, so the structure of the API, albeit primarily empirically determined, follows a form (shown in Fig. 9) vaguely suggested by VFW documentation. In terms of the graphics languages utilized, generally speaking, the API controlled its frame-capture processing with VFW procedure calls and employed OpenGL commands for texture rendering and the control of the head tracker (see Appendix B).

In order to give a better understanding of the processing required for one frame of video data, the following steps in their temporal and logical sequence are listed below:

1) Create main OpenGL window (parent window),
2) Create main VFW window (child of main OpenGL window),
3) Create capture window (child of main VFW window),
4) Set frame-event-callback flag so upon arrival of each video frame, it is associated with the capture window and is stored logically into a program-controlled "frame" buffer,
5) Connect the frame grabber driver to the capture window,
6) Capture a frame of video data (640 x 480 x 3) and store it in the frame buffer. In the center of the frame data is a circular region of the distorted reflection of the real-world scene from the spherical mirror,

7) Associate the frame buffer within a larger texture area (1024 x 1024 x 3),

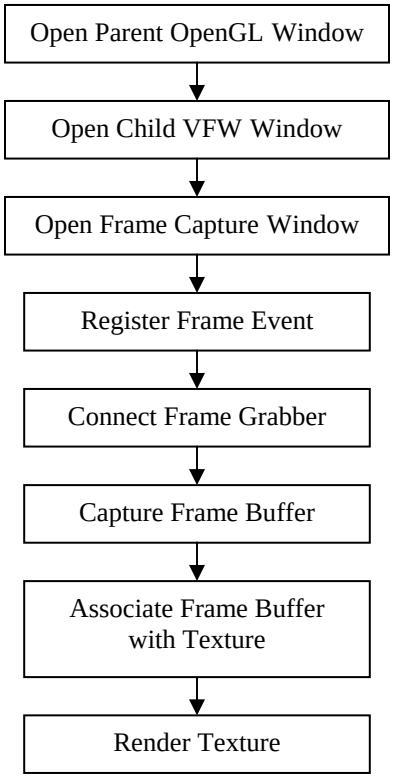
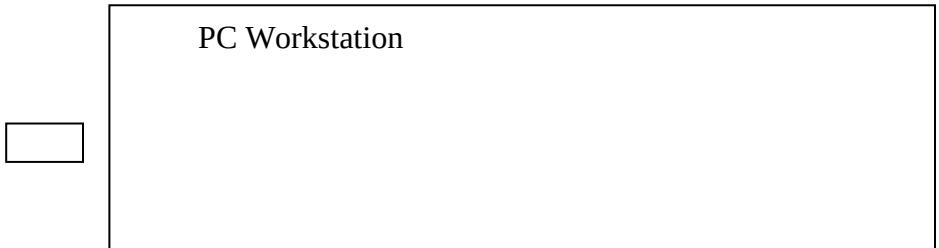


Figure 9. Flow Diagram for the VFW/OpenGL API

8) Sequentially render small planar quadrilateral portions of the distorted, circular texture (A-B-C-D) to similarly planar quadrilateral elements at appropriately mapped locations on a truncated hemispherical polygon until a complete correspondence between the distorted texture and the hemispherical polygon is achieved (see Fig. 10).

The mapping process, described in Step 8 above, geometrically simulates an eyepoint at the reflective mirror's center. The dimensions of the truncated hemispherical polygon are



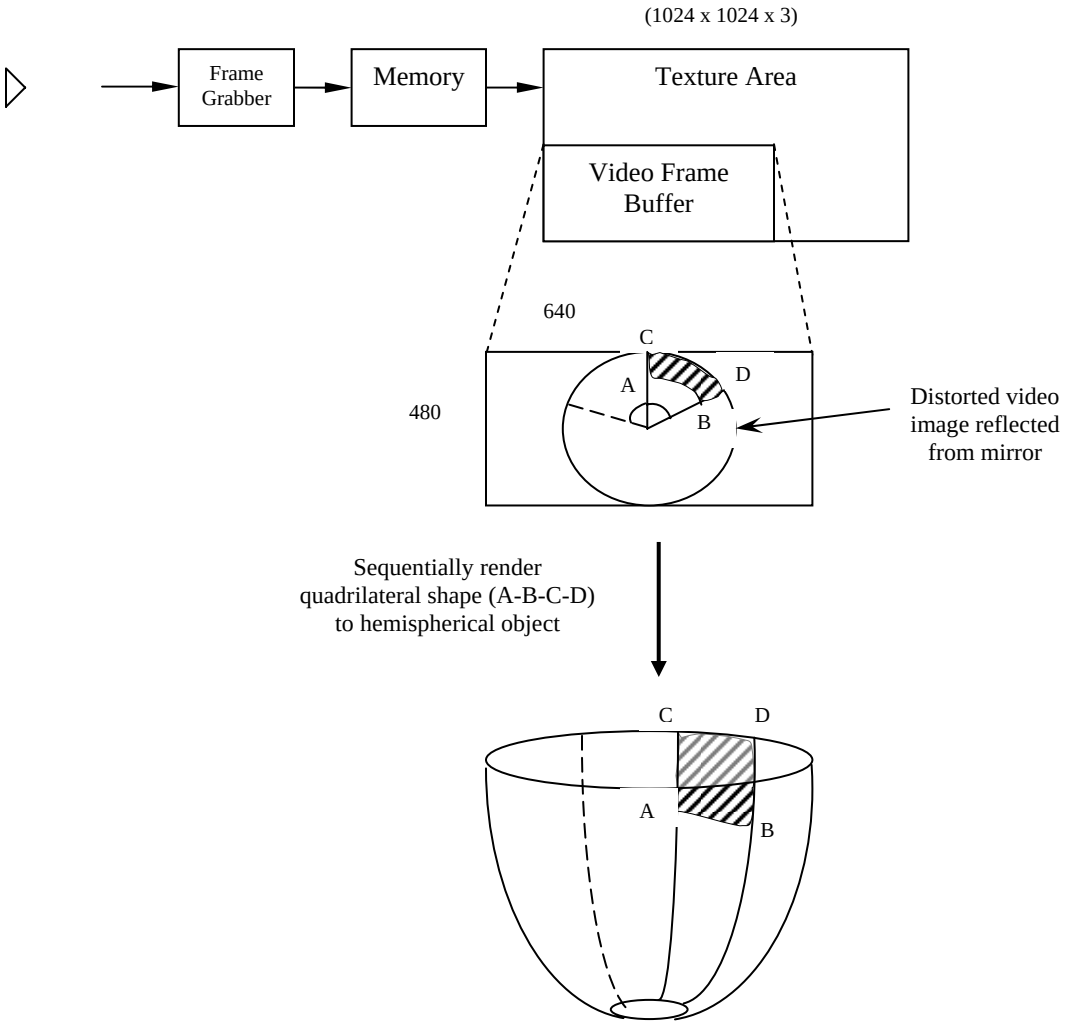


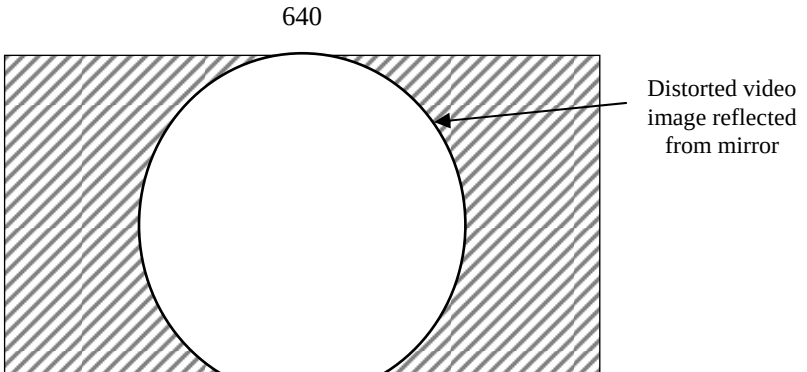
Figure 10. Graphical Depiction of the Immersive Omnidirectional Video Rendering Process. *The transformation consists of "remapping" the video image textures onto a substantially hemispherical object simulating an eyepoint at the reflective mirror's center and, in turn, indirectly undistorting the video image.*

spatially identical to the half of the convex, spherical mirror seen by the video camera; therefore, the transformation from the distorted texture to the polygon is merely a remapping of the image to that as seen from the mirror's optical center. In turn, this remapping mechanism indirectly undistorts the video image. As the number of small

quadrilateral segments of the distorted, circular texture increases, the amount of nonlinear artifacts of the overall hemispheric mesh are reduced because adjacent hemispheric quadrilateral elements become more laterally and vertically blended and the mapping more smoothly conforms to a hemispheric shape. On the order of 700 quadrilateral elements are required to achieve a qualitatively acceptable video output image. Despite this large number of elements, the display rate of the image was unaffected. The remapping transformation into spatial coordinates has been incorporated into a MATLAB script (see Appendix C).

Immersive Omnidirectional Video System Limitations

One of the goals of this study was to explore the major practical limitations of the Immersive Omnidirectional Video system. There are a number of systemic limitations that exist as a result of the system's architecture. One of the principal limitations imposed on this system is the resolution of the final output image to the HMD. Perhaps the most straightforward method of describing the inherent loss in resolution of the system is to calculate the change in a metric called the solid angle resolution utilizing the units of pixels/steradian during the remapping transformation from the distorted circular texture to the truncated hemispheric polygon. Referring to Fig. 11 (a graphical representation of the photo in Fig. 7), there are 180,956 pixels in the circular texture based on a 640 x 480 upper resolution limit of the Hauppauge WinTV frame grabber. An



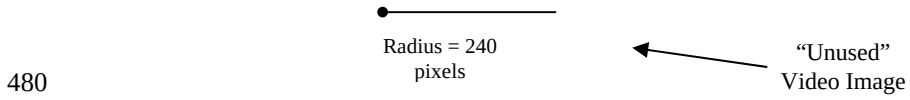


Figure 11. Graphical Representation of the Video Frame Buffer

assumed 65° horizontal x 48° vertical FOV for the video camera was estimated, but not actually measured, from the position of the camera and the dimensions of the input image plane. Therefore, using 48° as the total angular FOV, the solid angle resolution for the distorted reflection recorded from the spherical mirror is computed to be 333,123 pixels/steradian. When the same number of circular texture pixels is mapped to the truncated hemispheric polygon, they are spread over a much larger solid angle, in fact, a hemispheric solid angle. This expansion in surface area, in turn, reduces the solid angle resolution to 28,800 pixels/steradian, a factor of 10 decrease. For comparative purposes, the solid angle resolution of a 640 x 480 VGA output with a 65° horizontal x 48° vertical FOV is 323,230 pixels/steradian. Table 3 below depicts the correlation between various 2-dimensional resolutions and the corresponding hemispheric solid angle resolution of the final output image. From Table 3, the hemispheric solid angle resolution is directly proportional to the 2-dimensional resolution. Therefore, the output hemispheric solid angle resolution does not achieve a VGA solid angle resolution until the total 2-dimensional resolution is increased by a factor of over 10.

Furthermore, referring to Fig. 11, the mismatch between the smaller circular mirror reflection and the larger rectangular shape of the camera's image screen, in a sense, "disposes of" valuable captured input data and its corresponding resolution.

Table 3. Resolution Limitation of the Immersive Omnidirectional Video System

2-Dimensional Resolution	Number of pixels in distorted circular texture	Circular solid angle resolution (Pixels/steradian)*	Hemispheric solid angle resolution (Pixels/steradian)
2000 x 1600	2,010,619	3,284,925	**320,000**
1600 x 1200	1,130,973	2,094,394	180,000
1280 x 1024	823,550	1,516,080	131,072
800 x 600	282,743	520,505	45,000
640 x 480	**180,956**	**333,123**	**28,800**
320 x 320	45,239	83,281	7,200

* assumes a 65° horizontal x 48° vertical FOV for the video camera

Another limitation of the system is the processing time of the video frames. It has been currently measured at approximately 11 frames/sec. This delay is tolerable but not ideal. The likely cause of this delay is in the VFW/OpenGL API and, therefore, can be eliminated with a different implementation.

One restriction of the system that may arise is the effect that distortion may have on the ability to derive accurate stereoscopy. At the present time, however, it appears that the technique for dewarping the video images is effective and that any artifacts that may be introduced will have to be empirically minimized.

An unavoidable obstruction within the final output image is the reflected image of the video camera itself. The size of this obstruction is related to the distance between the camera and reflector. If, for example, the camera is set very close to the mirror, the resultant output image would consist primarily of the camera and, furthermore, the available reflected scene would be greatly reduced. In contrast, if the camera is placed far from the mirror, the reflected camera image would be quite small, but the reflected scene would also be further reduced. Consequently, the output image resolution would be severely compromised. The optimal camera placement should be no closer than that distance which enables the full hemispheric capture of the scene where the rectangular camera image limits just graze the circumference of reflected mirror (see Fig. 7). Regardless of the placement of the camera, however, the current system is limited to approximately 95% of the total hemispherical field available unless a lens similar to the Panospheric Imaging system is incorporated.

Immersive Omnidirectional Video System Recommendations

There are a number of deficiencies of the present Immersive Omnidirectional Video system that are strictly related to the particular implementation of its various components. A majority of these anticipated shortcomings are inherent in the nature of a short-term proof-of-concept system. Recommendations to improve the Immersive Omnidirectional Video system's performance are described below. They were formulated as a result of the work conducted thus far.

- The resolution components of the system transmit their information serially (camera, frame grabber, and then the HMD); therefore, the system resolution is only as good as

the poorest element in the sequence. The Virtual Research V8 HMD's performance is commercially state-of-the-art; however, the resolution of the entire system can be improved by introducing a digital video camera and frame grabber of greater resolution (for commercially available performance specifications see Table 4). Based on this study's resolution analysis, in order to achieve a VGA-like final output image, the required resolution of the digital video camera may exceed what is commercially available. However, professional digital cameras or those that are intermediates between commercial and professional are available that would meet or exceed this system's requirements.

- The processing rate of the system can be improved with a digital camera, a faster frame grabber, and a workstation with a higher processor speed (see Table 4). The digital camera would eliminate the frame grabber's operation of digitizing the analog frame buffer. There is no need to replace the Nvidia RIVA TNT graphics accelerator card because it is state-of-the-art and only incremental performance improvement would be gained by a commercial alternative.

- The Hauppauge WinTV frame grabber should be replaced with a more capable, yet more expensive, digitizing board, such as the commercially available Matrox Genesis-LC (see Table 4). The costs associated with the limited performance, poor implementation features and difficulty associated with developing the API of the Hauppauge system, in the long term, will outweigh the extra expense of a Matrox Genesis-LC, for instance.

- The mechanical Shooting Star head tracker should be replaced with an improved version or a higher quality device. A proper headtracking system should allow for calibration at the initiation of the tracking process, proper rotation responses, and does not occasionally introduce uncommanded jerks of the entire output display.

- An improved projection lens is required to ensure a complete, unobstructed hemispherical field of view. The Panospheric Imaging system's optic is an excellent example of one that is not obstructed by the camera itself.

- A recommendation associated with the tracker would be to incorporate the ability of the user to translate about the scene, if so desired, which would enable a magnification/zoom capability limited only by the resolution of the system.

Table 4. Suggested Immersive Omnidirectional Video System Commercial Component Replacements

COMPONENT	Specification
Digital Video Camera	
Resolution:	(500 lines) 380,000 pixels

PC Workstation	
Motherboard/Processor:	600 MHz Pentium III
Memory:	512 Mb SDRAM
Matrox Genesis-LC Frame Grabber	
Acquisition:	Analog or digital
Maximum acquisition rate:	140 MHz analog; 40 MHz digital
Display:	On-board display up to 1600 x 1200 with true-color capabilities
Host bus:	PCI
Memory:	6 MB WRAM image display buffer; 2 MB WRAM overlay display buffer
Additional feature:	Acquires video directly into system and display memory in real-time

- If so desired, develop the additional transformation algorithm or remapping process to undistort the scene that is greater than or "above" the full hemisphere. In other words, there may exist the need or desire to see "above the horizon," if the normal to the spherical reflector's zenith is pointed toward the ground.

The undistortion or remapping technique utilized by the current proof-of-concept system is very straightforward and easy to implement relative to prior art alternatives. It takes advantage of the improved capabilities in 3-D graphics software and hardware currently available in the marketplace. As a result, it is the preferred and recommended method for follow-on systems.

Finally, and most importantly, it is determined that the overall Immersive Omnidirectional Video system's architectural concept is sound and practical. Thus far, it performs successfully as a real-time, immersive equivalent of a mechanical pan, tilt, and rotation camera viewing system with no moving mechanisms. Furthermore, it is not

anticipated that subsequent development of the system will present any major shortcomings in either its implementation or its performance.

Future Work - Immersive Omnidirectional Video

Planned Development Steps

For the first year of this project, a proof-of-concept version of an immersive omnidirectional (180° by 180°) video system has been developed and will become the research platform for integrating stereoscopic displays and view-morphing algorithms. The planned steps to accomplish this integration are described below.

The first step in the further development of the Immersive Omnidirectional Video technology is to synchronize the video and audio channels. Currently, only the video data will be manipulated once it arrives at the PC workstation (later consideration may be given to incorporating a 3-dimensional audio capability). As a result, the audio data will arrive at the user well before the processed video output. Therefore, what follows is an example of an empirical process to determine the temporal offset or delay required to ensure that the two channels arrive at the user simultaneously.

The two data streams, video and audio (S_V and S_A, respectively, in Fig. 12), diverge either before or after the frame grabber digitizes the video stream. The video images arrive in a sequence of frames through a digital bus of the PC workstation into a two-frame deep RAM buffer (F_T and F_{T-1}). Initially the frames are optically compressed due

to the distortion from the wide-angle lens of the video camera. A first component, A_U, undistorts the frames and transfers them to another RAM buffer, U. A second

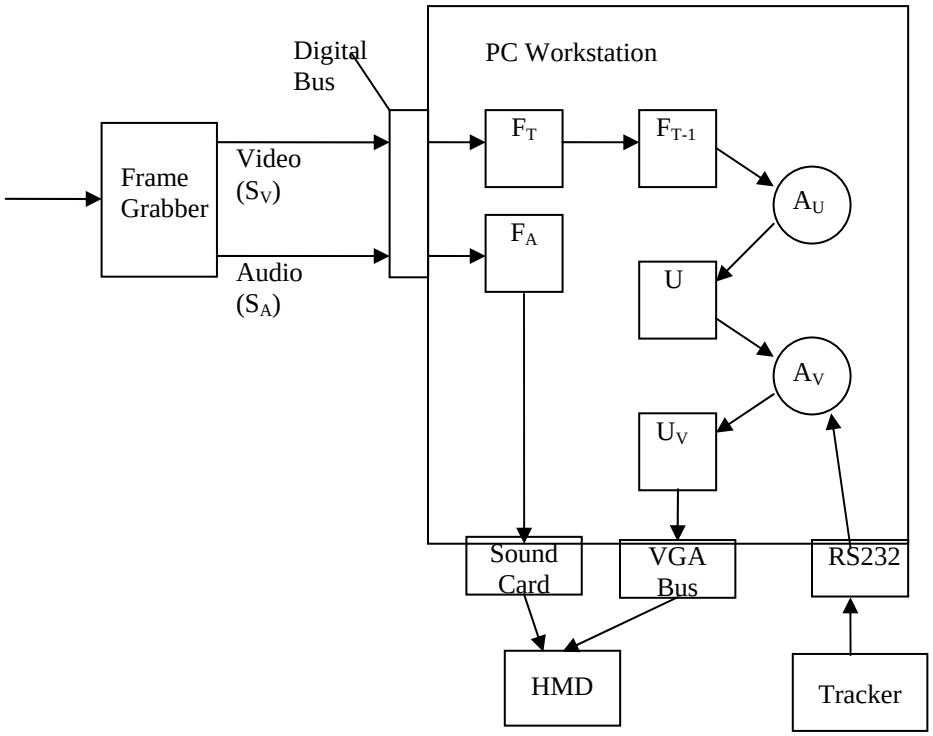


Figure 12. Pictorial Representation of the Audio Synchronization Process for the Immersive Omnidirectional Video System

component, A_V, uses signals from the head tracker to determine which portion or "window" of the U frame to be displayed in the HMD. This "window" frame, U_V, is sent through a VGA bus to the HMD. The time delay for the processing of one full frame to the next (F_T to F_{T-1}) is on the order of 16 msec, the image processing delay from the A_V routine to the U_V buffer is on the order of 16 msec, the delay from the U_V buffer to the HMD is approximately 16 msec—for a total delay of approximately 48 msec. The audio

channel, S_A, therefore, is stored theoretically in buffer, F_A, for roughly 48 msec so that it arrives synchronously with the appropriate video images at the HMD.

The next step in Immersive Omnidirectional Video development is the integration of extreme wide-angle optics into the development platform. There are several alternatives that will be considered: a) cylindrical projections, b) spherical or "fish-eye" projections, and c) parabolic projections. Others have considered each of these projection methods and cite their distinct advantages and disadvantages (Nayar, 1998; Zimmerman, 1993). Extreme wide-angle systems enable motionless, 360-degree camera panoramas which while desirable for some uses may detract from others both for reasons of low local resolution and possible distortion artifacts. The need for a full 360° x 360° spherical panorama capability depends on the particular application. The number of cameras used to achieve this capability depends on both the resolution requirements and the practical limits in terms of size. In this project, only the use of pairs of cameras, and not large numbers of them, will be evaluated. Furthermore, in many cases for simplicity in testing, the full spherical panorama will not be computed. Nonetheless, the techniques developed for this system will eventually be extended.

Various projections will be tested to see how they interfere with the ability to derive accurate stereoscopy and how it impacts the ability to render view-morphed images. Although it is expected that some special purpose extreme wide-angle lenses will be required, it is possible to test these projection systems in simulation by mathematically modelling the distortions and testing their respective effects on our real-time algorithms.

More specifically, virtual 3D scenes using 3D Studio Max can be created, a virtual camera with various lens properties can be set up (for instance, 3D Studio has built in fish-eye lenses), and then the scene for a particular type of lens can be rendered. In general, the rendered output from 3D Studio would serve as a static scene input to the virtual video algorithms. A dynamic scene could be further simulated by rendering an .AVI movie file.

The final step in the second year of work will be to test the ability to extract stereoscopic views based on the view-morph algorithms being developed by CMU. CMU has already developed the preliminary real-time algorithms for triangle-based view-morphing, so the derived-stereo objectives can be incorporated and tested immediately. Because these morphing algorithms process clusters of pixels, they are expected to introduce some artifacts which will have to be experimentally minimized.

In theory, the processing of the video signals for the view-morph system will be very similar to the non view-morph operation (the audio and video synchronization process described above). However, in this case, there would be multiple sources for the frame sequences (F1, F2, etc, see Fig. 13) from each of the video cameras. Consequently, each of these sequences of frames would undergo their own separate undistortion processes ($A1_U$, $A2_U$, etc.) and then transfer to RAM buffers, U1, U2, etc, respectively. All of the undistorted frames will be "morphed" together in a piece of software, A_M, that will be developed by CMU. Beyond this point, the computational processing of the video frames proceeds in a manner similar to the non view-morph system.

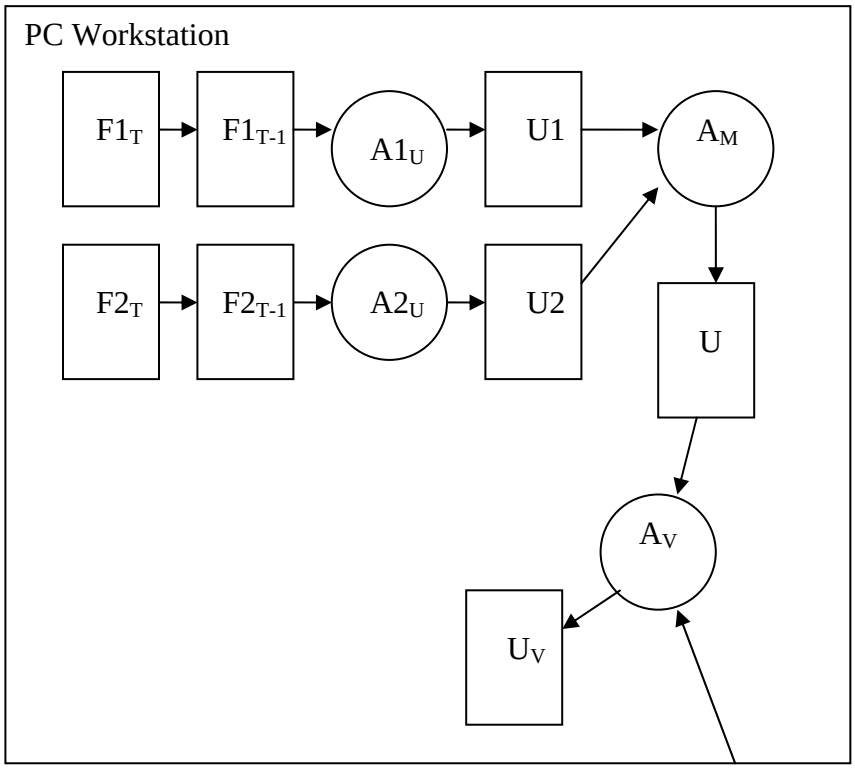


Figure 13. Pictorial Representation of the View-Morph Video Processing for the Immersive Omnidirectional Video System

Immersive Omnidirectional Video System Evaluation Tests

In the development process shown in Fig. 4 sequential milestone points have been defined where MIT and CMU integrate their software and evaluate the combined system. These are described below.

Immersive omnidirectional video evaluation

The goal is to demonstrate that the basic Immersive Omnidirectional Video implementation works successfully. For this milestone, the following system properties will be tested and measured: 1) end-to-end system lag for changing the viewpoint (expected: 50-60 ms), and 2) ability of an observer to tell the difference between the Immersive Omnidirectional Video system versus a conventional HMD with helmet mounted video cameras.

To measure the end-to-end system lag, a head sensor will be placed on a turntable equipped with a high-speed phototransistor switch. By calibrating the sensor with the light switch under static conditions, the onset of a predetermined sensor position can be accurately measured. Then, by placing another phototransistor inside the HMD, the onset of a display field set to be illuminated at the predetermined sensor position can be measured. Using an oscilloscope, the time delay between the phototransistors will be measured with high accuracy.

Further, naive users will be asked if they notice important differences when they are looking through an HMD with head-mounted cameras, as opposed to an Immersive Omnidirectional Video system. In the first case, the input video images to the HMD will be buffered in computer memory so that the end-to-end system lag exactly matches that of the Immersive Omnidirectional Video system. The contents of the visible scene as well as the field-of-view will be matched and the field of regard will be limited so the two cases correspond.

Interpolated view-morph stereoscopy integration and evaluation

The goal of this milestone is to show that stereoscopic views of a scene can be rendered by view-morph *interpolation* and that these derived images can be fused by human viewers to perceive the 3D depth of a scene. (View extrapolation will be tested later). Using human depth perception to test the fast view-morphing methods will provide an immediate indication of the success of the algorithms. The preliminary MIT and CMU software components will be integrated and 1) measure both overall system lag and the processing lag of the CMU view morphing software, and 2) assess the accuracy of the perceived stereo depth.

The end-to-end system lag will be measured again in order to determine the incremental cost of adding the algorithms to undistort the video images. To determine the accuracy of the derived stereoscopic view, a simple distance judgement experiment will be performed. In this experiment, luminous spheres will be displayed at varying distances from an observer in an otherwise darkened room. The spheres will vary in size such that the angular subtense of each sphere is constant at the observer's viewpoint. The dependent measure will be a magnitude estimate of the egocentric distance to each target sphere. Users will be asked with normal stereo vision to compare 3 conditions: 1) normal stereo direct viewing (i.e., not using any display), 2) stereo HMD viewing using helmet mounted cameras, and 3) view-morph derived stereo HMD viewing.

Omnidirectional viewing demonstration

The objective of this milestone is to display a portion of a hemispheric video image captured with wide-angle optics on an HMD, updating the stereo morphed image pairs based on the user's head movement. The following system properties will be tested and measured: 1) residual static image distortion due to errors in the lens model, 2) processing lag of the algorithms to undistort the video images, and 3) compare results with "fish-eye", cylindrical, and parabolic projections.

The end-to-end system lag will again be measured in order to determine the incremental cost of adding the algorithms to undistort the video images. Again both the end-to-end system lags and the field-of-regard of the video camera HMD will be matched to the omnidirectional video system and test whether users notice striking differences between the two display methods. However, the resolution of the video camera HMD must be decreased to mimic that of the omnidirectional system; this will be accomplished by capturing the video at an appropriately lower resolution and using a fast enlarging technique (e.g., pixel-doubling to fill the display area). A test of stereoscopic depth similar to that above will also be performed. Performance should be worse (show greater variance) with a wide angle lens for two reasons: 1) optical compression by the lens will have reduced the effective angular resolution, and 2) errors in the lens model will introduce distortions in the stereo disparity which in turn will result in depth errors.

Conclusions

In its fully developed form, Virtual Video requires the integration of three techniques: Immersive Omnidirectional Video, Video View Morphing, and Augmented Reality. As a first step, this thesis defined the system concept and focussed on development of a proof-of-concept immersive omnidirectional video system. The work for this thesis will

comprise a small part but will nonetheless lay the foundation for the entire Virtual Video project. Real-time Video View Morphing and Augmented Reality remain to be developed and integrated in future work.

The two technical objectives of this thesis were met: first, by demonstrating a proof-of-concept system on a PC platform of the omnidirectional video technology, and second, by providing both a brief engineering description of the entire, fully mature Virtual Video system and a more detailed account of its component technology, Immersive Omnidirectional Video. Specifically, the proof-of-concept development incorporated the use of a spherical reflector to support immersive (180° by 180°) real time viewing from a fixed camera. The proof-of-concept demonstrator served as an "experimental" platform whose aim was to thoroughly explore the major practical limitations of the omnidirectional video technology.

As a result, recommendations to improve the performance of the current Immersive Omnidirectional Video system were suggested. A majority of the proposed improvements were a result of the particular implementation of this system's various components. However, two principal limitations, the systemic resolution loss and the obstruction of the video camera itself within the final output image, were inherently related to the system's architecture. To counter the resolution loss, this report's analysis suggested a system imaging device, such as a digital video camera, that may be only professionally available or an intermediate between commercial and professional products. To ensure a complete, unobstructed view, this study recommends further consideration of the Panospheric Imaging system's optic.

Nonetheless, the state-of-the-art undistortion or remapping technique developed for this system was recommended for follow-on systems. More importantly, it is concluded that the overall system's architectural concept is sound and practical. Thus far, the proof-of-concept platform has performed successfully and it is not anticipated that subsequent development of the system will present any major shortcomings in either its implementation or its performance.

The management goal of this thesis was met by identifying a system process for planning, managing, and tracking the integration, test and validation of this program of research and development. This system plan broadly outlined the required development steps for the entire Virtual Video system and more specifically detailed the steps necessary to develop the Immersive Omnidirectional Video technology.

Bibliography

Bogner, S. (1995). An Introduction to Panospheric Imaging. Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics. pp. 3099-3106.

Chen, S.E. & Williams, L. (1993). View interpolation for image synthesis. Proceedings of SIGGRAPH '93. pp. 279-288.

Chen, S.E. (1995). QuickTimeVR --- An image-based approach to virtual environment navigation. Proceedings of SIGGRAPH '95. pp. 29-38.

Chen, S.E., Miller, G.S.P. (1995). Cylindrical to planar image mapping using scanline coherence. United States Patent 5,396,598.

Cruz-Neira, C., Sandin, T. A., & DeFantini, R. V. (1993). Surrounding screen projection-based virtual reality: the design and implementation of the cave. Proceedings of SIGGRAPH '93, 135-142.

Ferrell, W. R. (1965). Remote manipulation with transmission delay, M.I.T., NASA TN D-2665.

Fisher, S. (1990). Virtual interface environments. In B. Laurel (Ed.), The Art of Human Computer Interface (pp. 423-438). Reading, MA: Addison-Wesley.

Held, R., Efstathiou, A., & Greene. (1966). Adaptation to displaced and delayed visual feedback from the hand. Journal of Experimental Psychology, 72, 887-891.

IPIX, version 1.0. (1997). Interactive Pictures Corporation, Inc.

Kanade, T., Narayanan, P.J. & Rander, P.W. (1995). Virtualized reality: Concepts and early results. Proceedings of IEEE Workshop on Representation of Visual Scenes. pp. 69-76.

Kanade, T., Rander, P. & Narayanan, P.J. (1997). Virtualized reality: Constructing virtual worlds from real scenes. IEEE Multimedia, vol.4, no.1. pp. 34-46.

Kuban, D., Martin, H., & Zimmermann, S. (1994a). Omniview Motionless Camera Endoscopy System. United States Patent 5,313,306.

Kuban, D., Martin, H., Zimmermann, S., Busko, N. (1994b). Omniview Motionless Camera Surveillance System. United States Patent 5,359,363.

Nayar, S. K. (1998). Omnidirectional imaging apparatus. United States Patent 5,760,826.

Noro K, Kawai T., Takao, H. (1996). The development of a dummy head for 3-D audio visual recording for transmitting telepresence. Ergonomics, 39, 1381-1389.

Poulton, E. C. (1974). Tracking Skill and Manual Control. New York: Academic Press.

QuickTime VR, version 1.0. (1995). Apple Computer, Inc.

Rock, I. (1981). Anorthoscopic perception. Scientific American, 244, 145-53.

Seitz, S. M., & Dyer, C. R. (1996a). View morphing. Proceedings of SIGGRAPH '96, (pp. 21-30).

Seitz, S. M., & Dyer, C. R. (1996b). Toward image-based scene representation using view morphing. Proceedings of International Conference on Pattern Recognition, (pp. 84-89).

Seitz, S. M., & Dyer, C. R. (1997). View morphing: Uniquely predicting scene appearance from basis images. Proceedings of Image Understanding Workshop. pp. 881-887.

Seitz, S. M., & Kutulakos, K. N. (1998). Plenoptic image editing. Proceedings of Sixth International Conference on Computer Vision, (pp. 17-24).

Sutherland, I. (1968). A head-mounted three dimensional display. Proceedings of the Fall Joint Computer Conference, 757-764.

Wickens, C. D. (1986). The effects of control dynamics on performance. In Boff, K. R., Kaufman, L., Thomas, J. P. (Eds), Handbook of Perception and Human Performance, Vol. II, Ch. 39. New York: John Wiley & Sons.

Young, L. R., & Stark, L. (1963). Variable feedback experiments testing a sampled data model for eye tracking movements. IEEE Transactions on Human Factors in Electronics, HFE-4, 38-51.

Zimmermann, S. D. (1993). Omniview motionles camera orientation system. United States Patent 5,185,667.

Appendix A - NRA-98-HEDS-04 Proposal

The NRA-98-HEDS-04 Proposal was submitted to NASA in Feb, 1999. The authors were Charles M. Oman, PhD, Principal Investigator, and Co-Investigators, Andrew C. Beall, Ph.D., Steven M. Seitz, Ph.D., and William E. Hutchison, M.S. **NOTICE: The information contained in this proposal is furnished in confidence with the understanding that it will not, without permission of the offerors.**

PROJECT DESCRIPTION

1. Objectives and Significance

The goal of this project is to develop practical methods for creating a novel, photo-realistic, immersive augmented-reality display system that can be used in teleoperation, crew onboard training, private medical and family teleconferencing, and telemedicine. Computational experiments by Seitz and others have demonstrated the feasibility of interpolating or morphing photographs to render real scenes from novel viewpoints by processing digitized photographs. One of our objectives is to extend Apple Computer's Quicktime VR omni-directional desktop viewer concept to an immersive situation, so that an observer wearing a head-mounted display can make free head movements while viewing photorealistic images rendered in stereo from an arbitrary viewpoint using these morphing techniques. We refer to the resulting hybrid VR/view-morphing display as "Virtual Video". The user will interact with remapped real world video images, supplemented if necessary with artificially rendered objects. The inherent congruence of the real and artificial coordinate frames of this system reduces registration errors commonly found in augmented reality (AR) applications. In addition, the user's eyepoint is computed locally so that any scene lag resulting from head movement will be far less than those from alternative technologies using remotely controlled ground cameras. For space applications, this can greatly reduce the apparent lag due to satellite communication delay.

This proposal presents a plan for a three-year program of research and development of several enabling software technologies and ground-based validation experiments that will provide preliminary tests of the efficacy these technologies. The technological hurdles that we propose to solve and extend include the following:

- Video View Morphing — develop view-morphing technology to allow rendering of continuous virtual viewpoints from a limited number of physical camera vantage points. This work will primarily be accomplished at Carnegie Mellon University (CMU).
- Immersive Omnidirectional Video — digitally simulate a pan/tilt camera[1] by processing a wide field-of-view input image (e.g., fish-eye lens) in real time much like Apple's Quicktime VR desktop panoramic image viewers or IPIX omnidirectional image viewers but extended to operate on dynamic video scenes. This work will be done at MIT.
- Augmented Reality — combine video and computer generated imagery superimposed in three dimensions. This work will also be completed at MIT.

What is innovative about this research is its integration of the three technologies and, in combination, these technological advances will greatly enhance traditional tele-application displays. By extending Quicktime VR-like omnidirectional viewers to

[1] We mean a pan-, tilt-, and roll-capable camera mount

operate upon real-time video, we effectively create a remotely servoed pan/tilt camera with no moving parts and no lags other than those inherent in the computational algorithms. By incorporating view-morphing technology, we will further simulate limited ranges of camera translation again without physically moving the camera. The computational demands for performing this technique are far greater, but tests already conducted by our team have demonstrated that effective algorithms can be implemented in real time on COTS hardware (see section 4.1 for details). Finally, our method for capturing, undistorting, and morphing real video images will provide a robust method for superimposing computer graphic images creating an "augmented reality" display. Conventional augmented reality systems use see-through head-mounted displays (HMD) and therefore must accurately align computer graphic images (CGI) upon the external scene. This method is prone to dynamic misalignments due to delays and to static inaccuracies due to head-tracking errors. By capturing the external scene using physically stationary video cameras and rendering the superimposed images in the same coordinate system, the registration of the images no longer depends on the accuracy or speed of the head-tracking system.

We will demonstrate that this technology can be fully implemented and easily deployed using the HRF VEG workstation, head-mounted display, and head-tracker that will be deployed onboard the ISS. We further plan to show the advantages of using this available technology for several ISS relevant tele-applications such as remote training and teleconferencing. Using this hardware, we propose to conduct several validation experiments that will pit our R&D technology against several traditional displays variously testing the effects of lag time, field-of-regard, and sense of presence (see section 4.5).

The far-reaching and in some instances near term NASA-specific applications of this research will lead to the following:

Teleoperation Scenario

Operating the RMS during shuttle operations, and now during ISS assembly, is a perceptually demanding task. When single views are available of the assembly area, even highly trained and experienced operators will inevitably make mistakes in judging distances to target objects. Using "Virtual Video", the input from one or more stationary cameras can be processed in real time allowing the operator to have stereoscopic views from arbitrary moving vantage points. In some cases, these may include vantage points that are physically impossible to place a real camera. Both the stereoscopic disparity and motion parallax will be used naturally by the operator to convey a much more veridical impression of the physical layout that will significantly reduce planning time and increase performance accuracy.

Crew On-board Training Scenario

Presumably pieces of equipment will have to be delivered to the ISS that have not been previously examined by the astronauts aboard. An astronaut could "virtually" stand next to a ground expert while he gave a training demonstration of the equipment's operation. While they talked with one another, the astronaut could alternate between looking directly at the expert and examining closer certain features of the equipment or "virtually" follow him as he walks around the equipment. In 1-G training applications, the "Virtual Video" system could be used to simulate the viewpoints achievable in a weightless environment.

Video-Teleconferencing Scenario

Using the "Virtual Video" system astronauts aboard the ISS will be able to experience a home-like situation such as "dinner time" that is more compelling than a video-teleconferencing session for a minimal incremental equipment cost. The astronaut will be able to look around the family's dining room and interact with his family as if he were sitting at the dinner table. The family will be able to maintain eye contact with the astronaut while they communicate. The astronaut could also virtually "walk" around the family's dining-room table if he so desired.

Virtual Cockpit Scenario

In advanced hypersonic vehicles and in some military applications, windowless cockpits may be required. Synthetic vision systems are already under consideration for these applications. "Virtual Video" techniques could also be applied to synthesize a view using sensors and data streams from both inside and outside the cockpit.

This proposal is organized in the following manner: Section 2 describes the present state of knowledge and previous work in view morphing, immersive omnidirectional video and augmented reality; Section 3 presents our scheduled approach for this proposal; methods and procedures are described in Section 4.

2. Relationship to Present Knowledge and Prior Work

2.1. Video View Morphing

Humans perceive the three-dimensional world by means of its projection onto our retinal fields. By mimicking this projection process, the eye can be fooled into perceiving depth and three-dimensionality in computer-generated imagery. Interactive computer graphics systems exploit this trick to enable a user to visualize and transform artificial objects and scenes in three dimensions via 2D display and input devices. A central tenet of this proposal is that the same principle also applies to *real* imagery: by modeling the effects

of camera motion in the image domain, it is possible to render real scenes from novel viewpoints by processing digitized photographs.

Cast as an image transformation, a change in camera viewpoint can be modeled as a mapping, or *warp*, between pixels in one or more input views to pixels in a new image, representing a virtual view of the same scene. By categorizing the set of all such mappings, we can devise mechanisms for synthesizing changes in viewpoint by warping images. This capability is critical for interactive applications like visualization, virtual reality, telepresence, flight simulation, and other tasks in which viewpoint changes are subject to user-control. It also enables manipulation of scenes whose three-dimensional structure is unknown and unrecoverable. For instance, a present-day tourist could virtually visit a historic building, long ago destroyed, by warping a set of photographs taken at a time when the building was still standing. The illusion of a real three-dimensional space would be maintained by continually warping the images to correspond to the user's changing perspective. The same approach can be applied to drawings and paintings. For instance, Fig. 1 depicts a stereo pair of Leonardo Da Vinci's *Mona Lisa*, generated using the View Morphing approach described below.

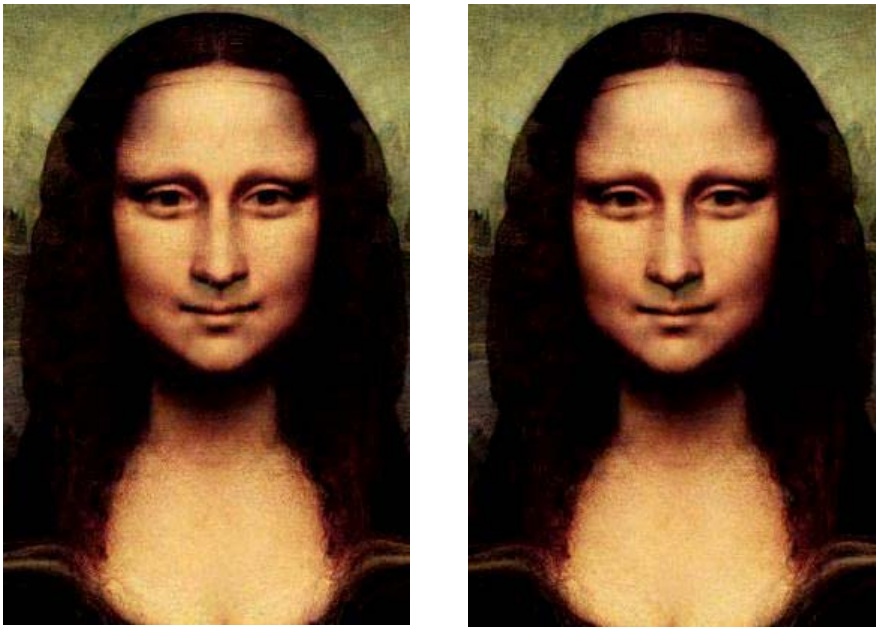


Figure 1. Stereo from View Morphing. *The above two images were produced by View Morphing an image of Leonardo Da Vinci's Mona Lisa with its mirror reflection. Stereoscopic fusing (uncrossed) of these two images produces a three-dimensional view of her face and torso. The View Morphing technique produces a sequence of such images, depicting a 3D rotation of her head and torso.*

The View Morphing approach was inspired by the realistic effects obtained by *image morphing* techniques (Beier & Neely, 1992), often seen in film and television

commercials. These techniques combine 2D interpolations of shape and color to create seamless and often dramatic transitions between a pair of input images. Despite being computed by 2D image transformations, effective morphs can suggest a natural transformation between objects in the real word. Given this property, it is natural to consider if image morphing can be used to synthesize *camera transformations*. Specifically, suppose we applied a standard image morphing technique to interpolate images of an object from two different camera positions—would the resulting sequence of in-between images correspond to correct perspective views of the scene? Perhaps not surprisingly, the answer is no—changes in object pose or viewpoint often cause unnatural distortions in image morphs that are difficult to correct manually.

Using basic principles of projective geometry, however, it is possible to extend image morphing to correctly handle 3D projective camera and scene transformations. This extension, called View Morphing, works by prewarping two images prior to computing a linear morph and then postwarping the interpolated images (Seitz & Dyer, 1996a). A key property of the approach is that it guarantees physically correct views of the scene, i.e., images that are identical to what a real camera would see from the corresponding viewpoint.

Lippman (1980) was one of the first to propose the use of an image-based representation for computer-aided 3D scene visualization. His *Movie-Map* approach enabled interactive virtual exploration of an environment by selectively accessing views that were previously captured and stored on laser disk. An additional contribution was the use of panoramic imaging to capture a wide field of view. These images were optically corrected at playback time using a viewing apparatus employing a conical mirror. By moving one's head relative to this device, the user could effect a rotation of the virtual camera about its optical center.

In their seminal paper on view synthesis, Chen and Williams (1993) described how new views of a scene could be quickly computed from a set of basis images and range-maps via image warping in software on desktop computers. A primary motivation was speed; interpolation could be performed more quickly than rendering directly from a 3D model, especially when high quality rendering methods like ray tracing or radiosity were used. This observation led to the development of novel hardware systems that use view synthesis techniques to achieve real-time rendering rates for synthetic 3D scenes (Regan & Post, 1994; Torborg & Kajiya, 1996; Lengyel & Snyder, 1997).

Laveau and Faugeras (1994) were among the first to develop view synthesis techniques that operate on photographs and therefore apply to real scenes. While others (Tomasi & Kanade, 1992; Ott, Lewis, and Cox, 1993) had devised methods for synthesizing views from known (a priori or derived) depth maps and camera positions, they were the first to demonstrate the feasibility of view synthesis directly from uncalibrated images and correspondence maps. To solve the problem of mapping points between uncalibrated views, they developed techniques based on the *fundamental matrix* (Longuet-Higgins, 1981; Deriche Zhang, Luong, and Faugeras, 1994). A second contribution was a ray-tracing method for resolving occlusion relationships in synthesized views.

The last three years has seen an explosion in interest in view synthesis techniques and applications. Concurrent with the work by one of the proposers (Seitz & Dyer, 1995b; 1996a; 1996b; 1997a; 1997b; 1997d; Seitz, 1997a) and numerous other researchers (Mann & Picard, 1995; 1997; Werner, Hersch, and Hlavac, 1995; Irani, Anandan, and Hsu, 1995; Kanade, Narayanan, and Rander,1995; Szeliski & Kang, 1995; Kumar, Anandan, and Irani, 1995; McMillan & Bishop, 1995a; 1995b; Chen, 1995; Katayama, Tanaka, Oshino, and Tamura, 1995; Szeliski, 1996; Moezzi, Katkere, Kuramura, and Jain, 1996; Scharstein, 1996; Beymer & Poggio, 1996; Vetter & Poggio, 1996; Debevec et al., 1996; Gortler, Grzeszczuk, Szeliski, and Cohen., 1996; Levoy & Hanrahan, 1996; Avidan & Shashua, 1997; Robert, 1997; Horry et al., 1997; Szeliski & Shum, 1997; Kanade, Rander, and Narayana, 1997) developments have been made in other view synthesis techniques. For brevity, we discuss only a few here that are representative of the dominant approaches that currently exist.

Chen (1995) and McMillan and Bishop (1995a) developed systems similar to Lippman's Movie-Maps but using computer-based rather than optical methods. Both of these systems provided panoramic visualization of scenes using one or more cylindrical image mosaics, each of which stored scene appearance from a single camera position. McMillan and Bishop's *Plenoptic Modeling* approach also supported interpolation of mosaics to allow camera translation using an elegant image-space algorithm to resolve visibility order. Apple Computer's commercialization of Chen's system, *QuickTime VR* (1995), has brought image-based scene visualization to the mainstream, enabling interactive viewing on conventional PC's of real scenes like the Great Wall of China or the interior of a new car. The success of QuickTime VR which uses cylindrical projections for panoramic viewing of static images on desktop computers and newer systems like *Surround Video* (1997), *IPIX* (1997), *SmoothMove* (1997), and *RealVR (*RealSpace, 1997) which use spherical projections to achieve omnidirectional viewing has helped spawn an emerging subfield of computer graphics. These are often called *image-based rendering* systems and have attracted growing interest.

Most closely-related to the approach adopted in this proposal are the so-called *view interpolation* methods (Chen & Williams, 1993; Laveau & Faugeras, 1994; McMillan & Bishop, 1995b; Werner et al., 1995; Beymer & Poggio, 1996; Scharstein, 1996; Avidan & Shashua, 1997) which use image warping to produce new views from a small number of basis views. Most of these methods require advance knowledge of camera parameters to produce correct perspective views, with the exceptions of Laveau and Faugeras (1994) and Avidan and Shashua (1997). Furthermore, all of these methods require dense pixel correspondence maps as input. This latter requirement is a serious limitation, given that image-based computation of correct correspondence maps is an ill-posed problem (Poggio, Torre, and Koch, 1985; Verri & Poggio, 1989). In contrast, the view morphing procedure adopted in this proposal operates directly on a pair of images, without knowledge of camera parameters or dense correspondence. It has the unique property of requiring only measurable image information, thus establishing that view interpolation is a well-posed problem.

Despite all of the recent progress in image-based techniques, the problem of simulating viewpoint changes in video remains an unsolved research problem. Video represents a unique challenge, owing to the requirement that all processing must occur in real time. While some of the aforementioned techniques are capable of real-time rendering, all require an expensive off-line processing stage in order to obtain accurate depth or correspondence information. A key innovation of the proposed work is to develop accelerated image processing and rendering approaches that will enable rendering a streaming video sequence from new viewpoints in real time.

2.2 Immersive Omnidirectional Video

Historically, traditional tele-application displays have had numerous drawbacks. Typically, the user views a video monitor displaying the output of a video camera at the other end of the application. The camera's point-of-view is either stationary or can be pan/tilt controlled via a remotely operated servo motor. When available, the servo motor is commonly controlled by the user with a joystick or some other similar pointing device. Mechanical impedance and communication delays make quick camera motions difficult to control (Nayer, 1998; Zimmerman, 1993). Under the best of conditions, it remains difficult for the user to spatially integrate the changing camera perspective into a contiguous percept of the surrounding space captured from the opposite end of the tele-application system (Rock, 1981). Other difficulties are specific to the particular tele-application. As an example, users of video tele-conferencing often complain of a lack of proper eye-gaze information that results from having the view screen and input camera at different angular positions; some even believe that the resulting negative interpretations may undo the attempted gains of adding the video channel (Noro, Kawai, and Takao, 1996).

Donning a head-mounted display (HMD) is a potential method of overcoming the difficulties of integrating multiple viewpoints by allowing the user to control the camera's pan/tilt with natural head movements (Sutherland, 1968). Not only does this simplify the camera control issues, but studies have shown that if a user indeed feels immersed in the visual environment this may consequently have an impact on performance. NASA has contributed to many of the early pioneering efforts to "develop a new kind of interface that would be very closely matched to human sensory and cognitive capabilities" (Fisher, 1990). With regards to tele-applications, however, the deleterious effect of control lags between the sensed head motions and the concomitant scene changes (via a remotely servoed camera) are exacerbated in part because we are accustomed to an extremely stable visual world provided to us by our vestibular-occular reflex (Young & Stark, 1963). Concerns over these control lags have driven the development of immersive but physically large, back-projected panoramic displays such as the CAVE system (Cruz-Neira, Sandin, & DeFantini, 1993). While this technology overcomes several of the concerns raised above, the physical dimensions and lack of

portability of CAVE-type systems make them impractical for many applications such as NASA flight operations.

Time delays between user movements and visual feedback can have deleterious effects on performance. Delays between a user's movement and the visual feedback in a tele-application display can arise from a number of sources, including transmission time for communication with a remote site, sensor delays used in head or hand tracking, and rendering delays of the graphics processor. The effect of delay on manual tracking performance has been thoroughly investigated (Poulton, 1974; Wickens, 1986); as has its effect on remote manipulation in teleoperator systems (Ferrell, 1965). Held, Efstathious, & Greene (1966) conducted an adaptation experiment examining the effects of prism displacements and visual feedback delays; in their study, they found that delays as short as 300 ms eliminated adaptation to the displacements.

2.3 Augmented Reality

By utilizing natural images, augmented reality overcomes one of the greatest limitations of virtual environment (VE) systems, namely a lack of realism and pictorial richness. As with VEs, there is no single definition for AR, but most investigators in the field would agree that AR: 1) combines real and virtual images, 2) allows real-time interaction with both types of images, and 3) registers the two types of images in 3D (Azuma, 1997). Research in AR is still very much in its infancy, but this is quickly changing as its strengths are discovered and the diversity of its applications is realized. We believe this technology can enhance human perceptual and cognitive systems if done well, and as we argue below our method is well poised to overcome some of the technical obstacles associated with development of a successful AR system.

Compared to VE displays, AR displays require more accurate and responsive head- and hand-tracking systems. A user in a VE system only notices tracking errors when he/she is able to sense the discrepancy across sensory modalities. For example, when head- and hand-tracking errors are great enough, a user will sense that the visually specified location of the hand does not match the proprioceptively specified location of the hand. However, in an AR display the cross-modality match is replaced by a visual-visual match. Now, any static misalignments between the real and virtual images are more easily detected and the visual acuity of most humans will require more stringent calibration tolerances than pure VE displays. Dynamic sources of misalignment can often outweigh even the static sources; any lag in sensing the location or orientation of the head and hand plus any lag in rendering the CGI will contribute to misalignments. To put this into concrete terms, assume a user is making a 50 deg/sec head rotation while wearing a VGA resolution HMD (640 x 480 pixels) with a 60 deg horizontal FOV. For dynamic errors of less than 1 pixel, the end-to-end system lag time would have to be less than 2 ms—not an easily surmountable hurdle given that most HMDs refresh at 60 Hz and not to mention that most graphic engines render at 30 Hz or slower.

There are two basic strategies for building an AR display. In one approach, the user directly sees the environment around him/her. The CGI is added by having the user view through partially transmissive optics that combine the generated images with those of the real world. See-through HMDs are one type of device that are used for this purpose. Since the physical environment is viewed directly, this method's greatest advantage is that the real component of the AR display can be extremely rich and will have no additional lag times due to head movements. However, there are numerous disadvantages of this method, including: 1) combining perceptually stable CGI is difficult due to dynamic misalignments, 2) since the real images are based on the physical environment, this method is limited in its ability to present alternate environments, and 3) added CGI cannot occlude real images but rather appear as semi-transparent "ghost" overlays. Unlike the second and third disadvantages, the first can in principle be solved. However, in practice this is extremely difficult.

In the second approach, the "real" environment is first captured via video and then digitally blended with the output of a graphics generator rather than being optically combined. This method immediately overcomes two of the disadvantages of the optically-based strategy: 1) it is straightforward to change the real images that are blended with the CGI in the display (either through chroma-keying techniques or by using remotely operated input cameras) and it is technically possible to allow virtual images to occlude real images in the scene (see "Z-keying" technique by Kanade, et al., 1997), 2) the difficulty of registering real and virtual images is reduced, though this depends on the actual implementation of the system of the video-based AR. In video augmented reality HMDs, video cameras digitally record the physical environment. The camera need not be near the user's eyepoint. The video signal is sent to a graphics processor that blends the CGI into the video scene, and with it one can closely simulate an optically-based AR system. In this setup, it is possible to eliminate the dynamic registration errors due to tracking lag times by adding a delay in the video stream that exactly matches the tracking and rendering delay of the virtual images. However, with our method of using a virtual omnidirectional view as described above, it is not necessary to purposely add delays in order to eliminate dynamic misalignments. This is true because the virtual images are added in the coordinate system of the stationary camera; delays in sensing the users head motion and in rendering the virtual pan/tilt are independent of the registration process and therefore do not contribute whatsoever to visible misalignments. We believe this is an enormous advantage offered by virtual omnidirectional viewer technology and to our knowledge has not yet been exploited. It is important to note that there are several disadvantages to using video-based AR; namely there is head-tracking lag associated with the entire scene (real and CGI) since the entire imaged scene is electronically mediated; and the imaged scene suffers a significant resolution loss both in the spatial and temporal dimensions.

Research being conducted by Takeo Kanade and his colleagues at CMU have made ground-breaking progress in producing virtual images of time-varying 3D scenes, in the context of their *Virtualized Reality* project (Kanade et al., 1997). The inputs to their system are video signals from 51 video cameras scattered throughout a room, each

viewing the scene from a different position and orientation. These images are processed using stereo matching algorithms and merged together into a three-dimensional model of the scene. Once the 3D model is obtained, it can be texture-mapped, rendered to new viewpoints, and composited with synthetic imagery. At present, the extremely high computational demands of processing 51 video streams and producing accurate 3D models requires significant off-line processing. In contrast, the goal of our proposal is to produce renderings of dynamic scenes from limited ranges of viewpoints in real time. Rather than attempt accurate 3D scene reconstruction, as Kanade and his colleagues have demonstrated, we instead propose to use efficient 2D image warping techniques that are amenable to real-time implementation and have been shown to produce highly realistic results (Seitz & Dyer, 1996a).

In virtual environment research, there is considerable debate about the definition of the term presence and its impact on user performance. Regardless, there is empirical evidence supporting the view that there is a positive relationship between presence, however defined, and task performance. Ellis, Dorighi, Menges, Adelstein, & Jacoby (1997) found evidence for improved manual tracking when a user's sense of presence was enhanced. Pausch, Profitt, and Williams (1997) demonstrated that performance in simple search tasks also increased with added immersion, and arguably more sense of presence. Presence seems to even affect low-level perceptual and motor processes (e.g. Yang, Wade, & Proffitt, 1997; Cohn, DiZio, and Lackner, 1996).

If presence is important, then how best should we attempt to maximize it in tele-application displays? One way to promote a sense of presence is to consider environmental interaction and how a user expects it respond to his/her actions. Typically most developers of VEs concentrate on these features, and although computer graphics are advancing at a remarkably fast pace, we have still not reached the point at which realistic 3D rendering blurs the line between reality and its representation (Ellis, 1991). A second approach is to promote social presence by having interaction with another "sentient" being inside the VE or tele-application. Video and audio tele-applications take advantage of the social aspect of presence.

3. Plan of Work

The goals of this proposed 3-year project are:

- To develop the three component techniques of "Virtual Video": real-time video view morphing, immersive omnidirectional video, and augmented reality techniques
- To demonstrate a proof-of-concept system on a PC platform similar to the NASA Human Research Facility computer workstation on the International Space Station

Our plan for developing the three components of "Virtual Video" is shown in the Fig. 2 below. The work has been divided between our laboratories at MIT and CMU reflecting the technical strength of each group. The developments require several steps, some of which can proceed concurrently. The individual steps are detailed in Section 4, subsections as indicated in Fig. 2. Component techniques are integrated at several milestones as described in Sections 4.5.2 and 4.5.4.

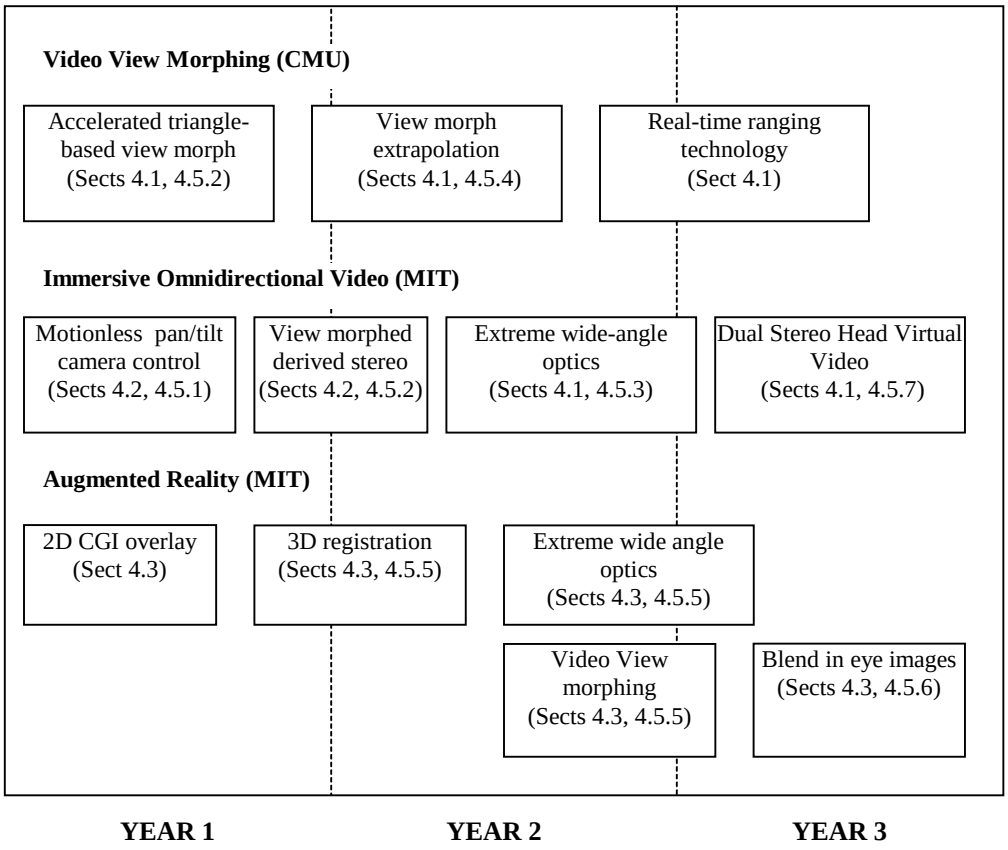


Figure 2. "Virtual Video" Plan of Work

4. Methods and Procedures

4.1. Video View Morphing

In *view* morphing, the basic principles of projective geometry are used to extend *image* morphing to correctly handle 3D projective camera and scene transformations. This technique works in three phases, as shown in Fig. 3. First the original input images are

prewarped using a 2D planar projective transformation. This operation has the effect of making the two image planes parallel to the line *L* between the optical centers. Following this step, it can be shown that linear interpolation of corresponding pixel positions in two such prewarped views yields physically-valid perspective views from new in-between camera viewpoints on *L*. Consequently, applying a linear morph of two prewarped images synthesizes a camera translation. The last step is to specify the gaze direction of the virtual view, which is achieved by applying a second planar projective transformation called a *postwarp*.

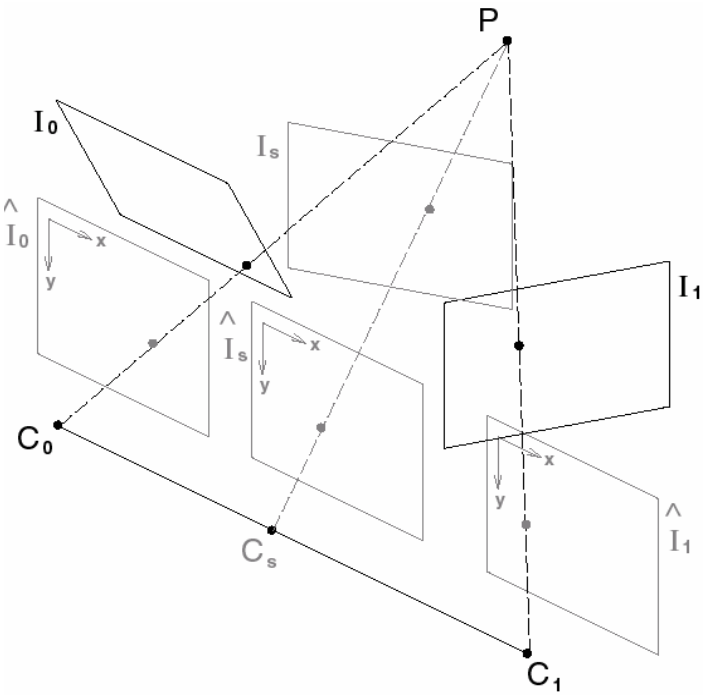


Figure 3. View Morphing in Three Steps: *(1) Original images I_0 and I_1 are prewarped to form parallel views $\hat{I}_0$ and $\hat{I}_1$. (2) $\hat{I}_s$ is produced by morphing (linear interpolating) corresponding points in the prewarped images. (3) $\hat{I}_s$ is postwarped to form I_s, the desired virtual view.*

Because the View Morphing approach is based on simple 2D warping operations, it is well-suited for real-time implementation. However, previous implementations of the algorithm were not tailored to efficient rendering. By exploiting widely available PC graphics acceleration boards, however, we have demonstrated one-to-two orders of magnitude improvement in performance over these previous implementations, enabling real-time interpolation of two images at approximately 10 Hz. In the first year of the

proposed work, we will implement further optimizations that are expected to provide rendering speeds of 30Hz using triangle-rendering acceleration available on low-cost PC boards. Rather than warp each individual pixel from its position in the input view, the triangle-based technique will work by aggregating each group of pixels that move similarly into a single triangle that is efficiently transformed using the texture-mapping operations. The number of triangles may be varied according to tradeoff accuracy for speed. We will experiment with different parameters in order to tune the system to provide the best performance subject to the constraints on lag time.

One limitation of the View Morphing approach is that it is limited to virtual camera motions on the line between the two camera centers C_0 and C_1. In the second year of the proposed work, we intend to develop view *extrapolation* capabilities that will enable generating views off of the line between the two camera centers. Such capabilities have already been demonstrated for other image-based techniques (McMillan & Bishop, 1995b) and are adaptable to the view morphing paradigm. The basic approach we will follow is to transform the correspondence map $C_{0,1}$ between I_0 and I_1 to create a correspondence map $M_{0,s}$ between I_0 and the desired view I_s. $M_{0,s}$ is a projective transformation of $M_{0,1}$ that can be computed based on the camera translation, rotation, and change in parameters between I_0 and I_s. Once $M_{0,s}$ is computed, the new view may be generated in real time using the triangle rendering algorithm described previously.

As shown in Fig. 4, two views enable generating reliable virtual view extrapolations for only a limited range of camera viewpoints. The ability to synthesize a wider range of views will require additional input images that cover a greater range of scene surfaces. To accomplish this goal, we will extend the view morphing approach in the second and third years of the proposed work to integrate input images from multiple camera viewpoints. The proposed approach will be to use two or more stereo heads, each yielding a pair of images, to extrapolate to new views within a limited radius (see Fig. 4). Specifically, to generate the virtual view V shown in the figure, we would extrapolate one image from each of the two stereo heads shown. The resulting two images would then be merged into a single composite view. Performing this merging step requires properly accounting for occlusions, i.e., handling differences in visibility in the images. We propose to solve this problem using Z-buffering (Chen & Williams, 1993). The Z-buffer algorithm is a standard technique in computer graphics for coping with occlusions in synthetic imagery that is easily adapted to image-based rendering. The basic principle is to estimate the depth value of each point in the image representing the virtual view. When a point from each stereo head is mapped to the same position in the virtual image, the point with smallest depth is drawn. This simple Z-buffer strategy ensures that scene visibility is correctly modeled in the view morphing procedure. For points with the same or similar Z-values, both images will contribute with weights based on proximity to V. This weighting step ensures that the virtual view agrees with the input image when V coincides to an input viewpoint.

In order to apply the View Morphing approach, it is necessary to have correspondence information between the images that are interpolated. For off-line rendering problems, this information can be supplied by hand (Seitz & Dyer, 1996a) or computed using off-

line stereo correspondence algorithms (Seitz & Dyer, 1995b). In order to obtain correspondence information in real time, we propose to use inexpensive real-time range finders that have recently become available. One example of such technology is the

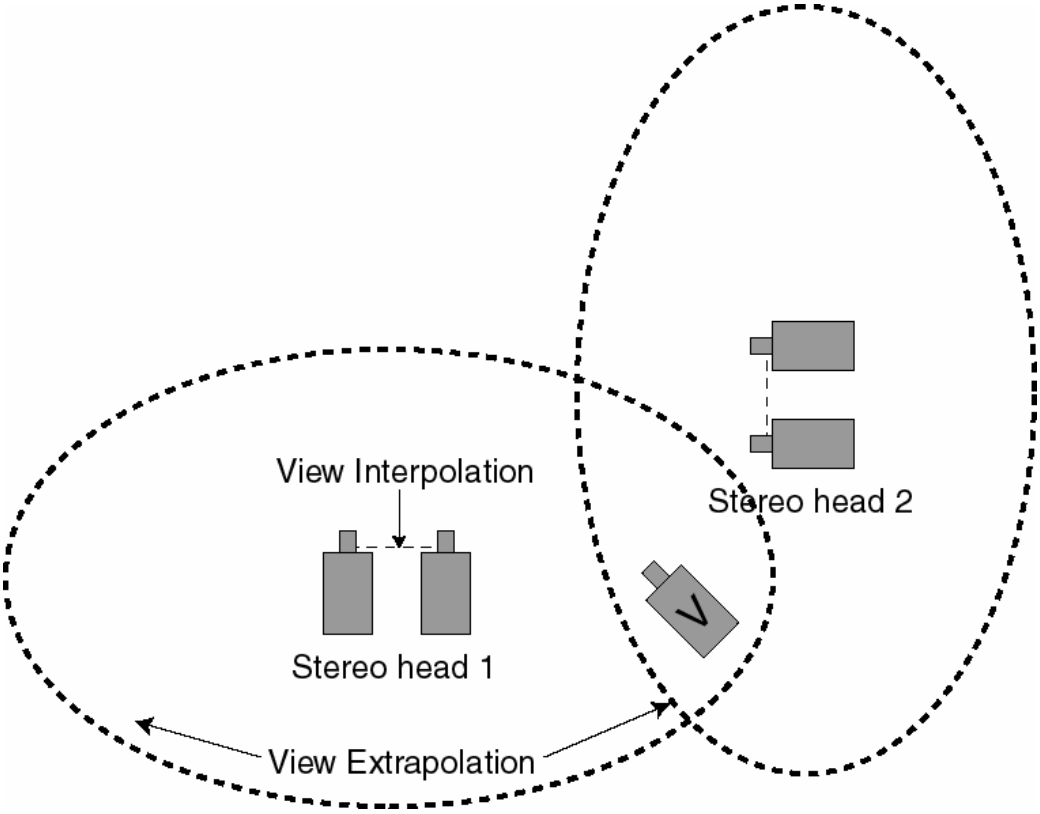


Figure 4. View Morph Extrapolation. *A video stereo head provides two images and correspondence information at 30 frames per second. View interpolation enables generating a limited range of new virtual viewpoints between the two cameras positions in the stereo head. In contrast, view extrapolation can produce new views for any virtual camera position and orientation. Since artifacts increase with distance from the stereo head, the range of extrapolation is limited in practice to a finite volume as shown. Two stereo heads provide a greater range of views, and the potential to create views of improved quality for camera positions that are in the extrapolation range of both heads.*

Small Vision System (SVS) developed by SRI (http://www.ai.sri.com/~konolidge/svs.html). The SVS software library, coupled with a low-cost stereo head available from Videre Designs (http:/www.dnai.com/~mclaughl), frame-grabber, and running on a 400 MHz PC, provides real-time stereo correspondence at over 30Hz for an image resolution of 320x240. The depth resolution and frame-rate are tunable, to enable trading off spatial and temporal resolutions. We believe that this system provides sufficient resolution, speed, and flexibility to satisfy our requirements in regards to this proposal. We intend to

use the requested funds to purchase two SVS systems (including stereo head, frame-grabber, PC, and software) to develop the real-time components of this proposal.

4.2. Immersive Omnidirectional Video

There have been several recent advances in optical, omnidirectional imaging devices (Nayar, 1998; Zimmerman, 1993). These have been combined with the Quicktime VR-like technology to provide the ability to digitally synthesize pan/tilt effects of a remotely mounted camera. We feel that these techniques are ideally suited for integration with an HMD immersive tele-application display. For the first year of this project, one of our objectives will be to develop our own limited version of a virtual pan/tilt system that will provide the research platform for integrating stereoscopic displays and view-morphing algorithms.

At MIT we have already constructed a simple development platform consisting of a PC and a low-cost frame capture board for testing virtual pan/tilt technology (see Fig. 5). We capture video images on the fly and use them as a texture that is updated in real time. In the simplest approach, the texture is mapped on a large rectangular polygon and a view is rendered, simulating a view of a portion of the originally captured scene. Moving the viewpoint achieves pan, tilt, and roll.

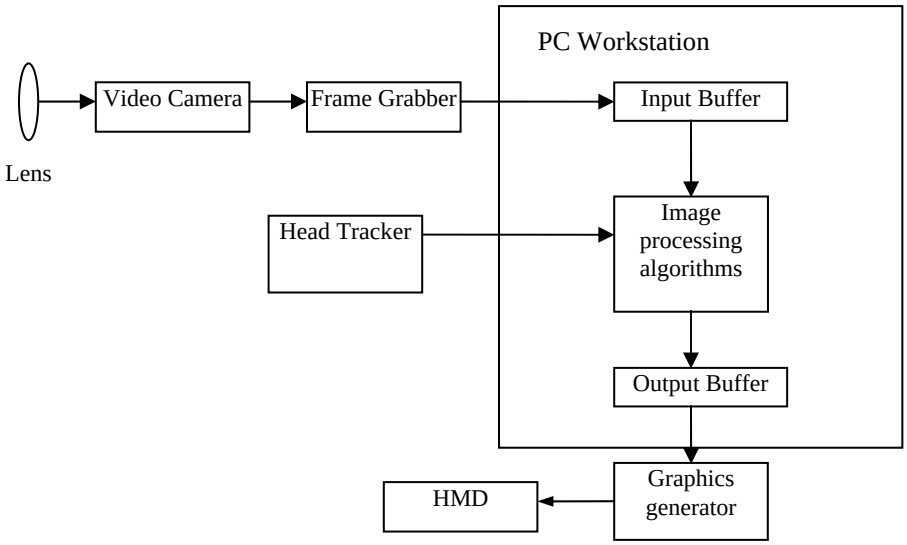


Figure 5. Immersive Omnidirectional Video Development Platform

Our next planned virtual omnidirectional viewer development step will be to control the virtual pan/tilt with a head-tracking device and to display the output in an HMD. At MIT we have several head-tracking devices (Intersense 600 Mark II, acoustic-inertial; Motion Star Flock of Birds, electromagnetic; and a Shooting Star, mechanical) and several HMDs (Kaiser Proview 80 and a Virtual Research V8 with built-in internal eye-tracking capabilities).

Once the virtual pan/tilt capability is in place, we plan to synchronize the video and audio channels (S_V and S_A, respectively, in Fig. 6). These two data streams diverge either before or after a frame grabber digitizes the video stream. The video images arrive in a sequence

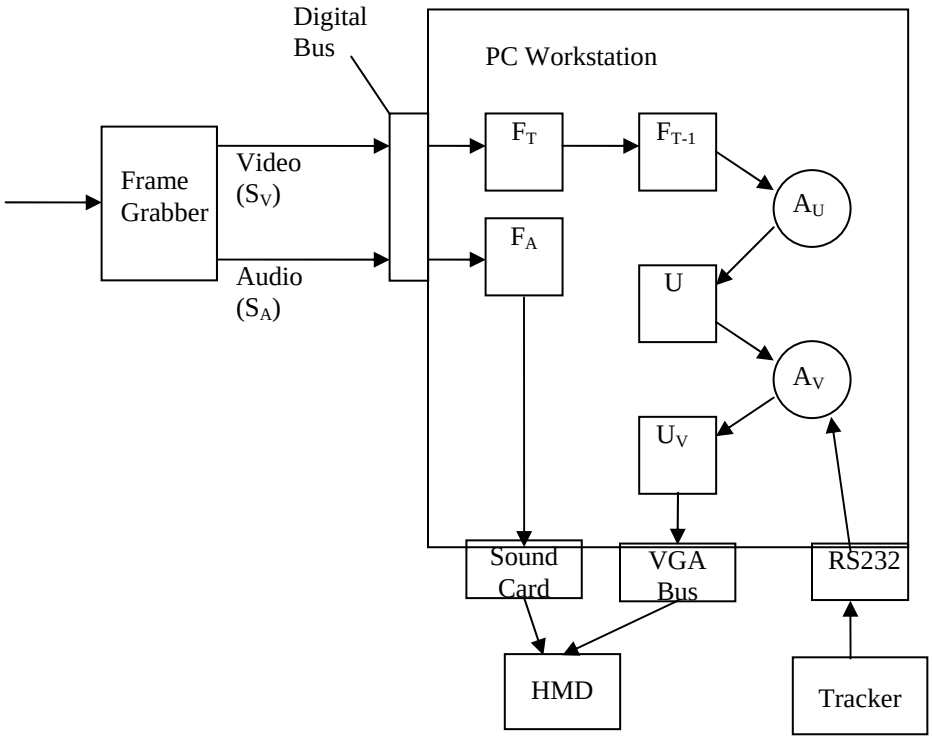


Figure 6. Pictorial Representation of the Immersive Omnidirectional Video Processing

of frames through a digital bus of the PC workstation into a two-frame deep RAM buffer (F_T and F_{T-1}). Initially the frames are optically compressed due to the distortion from the wide-angle lens of the video camera. A first component, A_U, undistorts the frames and transfers them to another RAM buffer, U. Both hardware and software techniques are possible depending on the type of transformation used. A second component, A_V, uses signals from the head tracker to determine which portion or "window" of the U frame to

be displayed in the HMD. This "window" frame, U_V, is sent through a VGA bus to the HMD. The time delay for the processing of one full frame to the next (F_T to F_{T-1}) is on the order of 16 msec, the image processing delay from the A_V routine to the U_V buffer is on the order of 16 msec, the delay from the U_V buffer to the HMD is approximately 16 msec—for a total delay of approximately 48 msec. The audio channel, S_A, is stored in buffer, F_A, so that it arrives synchronously with the appropriate video images at the HMD.

The second step will be to test the ability to extract stereoscopic views based on the view-morph algorithms being developed by CMU (see section 4.1). CMU has already developed the preliminary real-time algorithms for triangle-based view-morphing so we can begin the derived-stereo objectives immediately. Because these morphing algorithms process clusters of pixels they are expected to introduce some artifacts which will have to be experimentally minimized.

The third step of the virtual omnidirectional viewer technique is the integration of extreme wide-angle optics into our development platform. There are several alternatives that we will consider: a) cylindrical projections, b) spherical or "fish-eye" projections, and c) parabolic projections. Others have considered each of these projection methods and cite their distinct advantages and disadvantages (Nayar, 1998; Zimmerman, 1993). Extreme wide-angle systems enable motionless, 360-degree camera panoramas which while desirable for some uses may detract from our needs both for reasons of low local resolution and possible distortion artifacts. The need for full 360 deg. X 360 deg. spherical panorama depends on the application. The number of cameras used to achieve it depends on resolution requirements, and the number of cameras that it is practical to use. In this project, we will only evaluate the use of pairs of cameras, and not large numbers of them, and in many cases for simplicity in testing we will not compute the full spherical panorama. However, we believe our techniques can eventually be extended.

We will test how various projections interfere with our ability to derive accurate stereoscopy and how it impacts our ability to render view-morphed images. Although we have proposed to purchase some special purpose extreme wide-angle lenses in our budget, it is possible for us to test these projection (lens) systems in simulation by mathematically modeling the distortions and testing their respective effects on our real-time algorithms. More specifically, we can create virtual 3D scenes using 3D Studio Max, set up a virtual camera with various lens properties (for instance, 3D Studio has built in fish-eye lenses), and then render the scene for a particular type of lens. In general, the rendered output from 3D Studio would serve as a static scene input to the virtual video algorithms. We could further simulate a dynamic scene by rendering an .AVI movie file.

4.3. Augmented Reality

We propose to develop the AR technology in conjunction with the omnidirectional and view-morph components of this project. In addition, we will test develop a target application to demonstrate the utility of this technology. These endeavors are described below.

MIT lab has already begun experimenting with video-based AR. The first step will be to combine the virtual pan/tilt capabilities described in section 4.1 with a 2D CGI overlay. This step should be a straightforward merging using proven techniques and will provide a compelling demonstration that video-based AR eliminates many of the registration problems inherent in optical-based AR.

The next step will be to generalize the CGI blending techniques to include 3D geometric objects. To do this, we will need to accurately model the field of view of the camera (referred to as the lens model) in order to render perspectively correct CGI of 3D objects that can be placed anywhere in the scene. Once done, we will be able to place a rendered object anywhere in the tele-application display. To test the accuracy of the lens model, we will measure relative locations of real objects in the camera's FOV and then place virtual "twins" of the real objects in a blended scene. If there are any significant errors in the lens model then correspondence between the video and CGI should be obvious. Registration errors do not depend on head-tracking accuracy because the camera's eyepoint is stationary and head tracking is used only to pan and tilt that particular viewpoint.

With video-based AR, by combining the Z buffer information available with the video component and the depth information in a 3D geometric model, it is possible to correctly render occlusions of the CGI geometry with real objects in the video scene, since one knows the relative depths of all video and virtual objects. This is a powerful feature of video blending; this form of blending remains out of reach for optically based systems because of the difficulties in trying to selectively obstruct rays in the optical path (Azuma, 1997). We will evaluate this in the third year.

As the virtual pan/tilt methods progress throughout this project and use wider-angle optics and especially once the extreme wide-angle optics are employed, it is likely that the registration issues will have to be re-visited since inaccuracies in modeling these more sophisticated lens systems will inevitably result in alignment errors between the CGI and real images. We expect to solve these problems by empirically testing our models and adjusting the lens parameters until the registration errors between the two image types are minimized. Since we do not yet know what real-time lens correction algorithms will be best to undistort the projected images from the extremely wide-angle optics, we do not know how compatible these algorithms will be with the goal of minimizing registration errors. This is an important topic and must be addressed as part of the effort toward building a practical system and likely it will reveal important practical difficulties in developing wide-angle systems.

Once we have developed the basic methods of 3D CGI rendering, we will then merge those techniques with the methods of view-morphing. When this happens, the AR

problem changes; the added CGI must be rendered from a new vantage point. This is really not a problem, however, because the algorithms that will be used to render the CGI for the cameras' perspectives are general and can just as easily handle any arbitrary intermediate vantage point. In this sense, the problem is trivial. However, because view-morphing is an approximation technique, there will be distortions in the video scene while no such distortions will occur in the rendered scene. This will result in yet another source of registration errors. Without actually testing both the view-morphing algorithms in conjunction with the lens models, we cannot predict the severity of these artifacts. Testing and evaluating these combined technologies comprise a portion of our validation studies as discussed in section 4.5.

A TELECONFERENCING AUGMENTED REALITY APPLICATION

One application of "Virtual Video" is to allow an astronaut wearing an immersive HMD in flight to teleconference with his/her family on ground who are watching on a video monitor while both are being photographed by video cameras. However, a practical problem is that the HMD obscures the family's video view of the astronaut's eyes. This prevents the family from making eye contact, and impairs two-way social interaction. However, it is possible to remedy this by using an augmented reality technique to make the HMD effectively disappear (see Fig. 7). The solution is to use a head tracker to create a virtual cylindrical visor which follows the HMD wearer's head, and an eye movement monitor in the HMD to determine the wearer's angle of gaze. The virtual visor is mapped with a prerecorded texture showing the HMD user's upper head and moving eyes. Since this "visor" should always occlude the video image, no detailed depth map of the user's head is necessary, and by simply employing back-face culling of the visor, it will be easy to give the impression that it wraps tightly around the user's entire face.

Figure 7. Teleconferencing Augmented Reality Application. *Rendering an overlay of a previously photographed image of the user's eyes in 3D that will effectively mask the image of the HMD.*

We propose photographing the user's head in a number of different eye gaze positions. For instance, one might have the user fixate at an evenly spaced 10 by 10 (vertical by horizontal) grid and at each fixation record an image of the eyes. Then, by dynamically swapping the textures used to wrap around the CG visor, it will be possible to animate eye movements. To bridge the gap between the animated eye-movements and where the

user is truly looking, it is necessary to track the user's eye gaze while wearing the HMD in real time. MIT has a Virtual Research V-8 HMD equipped with a ISCAN video eye tracker. Using the eye tracker gaze direction data to select previously captured eye images, it should be possible to render the user's head, including eye-movement animation that closely mimics the user's actual behavior. We believe that empirical testing of such a system will allow us to prove this approach works, and answer important questions such as how finely sampled the eye gaze matrix must be made in order to portray realistic eye movements, and how accurately an outsider observer can interpret the eye gaze direction of the HMD user.

4.4 System Limitations

One of our goals is to thoroughly explore the major practical limitations of virtual video methods. Morphing techniques work best with objects which have no deep, narrow cavities (which occlude portions of surfaces from both views). Ideally, surfaces should have Lambertian reflectance properties (e.g. matte surfaces). How critical is this, as a practical matter ? There may be important tradeoffs between system update rate and the granularity chosen for triangle based morphing. What are the best tradeoffs ? When a surface is occluded in one or both views, how should this surface be textured ? (e.g. should it simply be rendered as gray ?) System field of view and resolution will depend on the number of cameras used, and the choice of lenses. What are the practical limits ?

4.5. Development Milestones & System Evaluation Tests

In the development process shown in Fig. 2 we have defined sequential milestone points where MIT and CMU integrate their software and evaluate the combined system. These details are summarized in the following subsections.

4.5.1. Immersive omnidirectional video evaluation

The goal is to demonstrate that the basic immersive omnidirectional video implementation works successfully. For this milestone, we will test and measure the following system properties: 1) end-to-end system lag for changing the viewpoint (expected: 50-60 ms), and 2) ability of an observer to tell the difference between the immersive omnidirectional video system versus a conventional HMD with helmet mounted video cameras.

To measure the end-to-end system lag, we will place a head sensor on a turntable equipped with a high-speed phototransistor switch. By calibrating the sensor with the

light switch under static conditions, we can accurately measure the onset of a predetermined sensor position. Then, by placing another phototransistor inside the HMD, we can measure the onset of a display field set to be illuminated at the predetermined sensor position. Using an oscilloscope, we will measure the time delay between the phototransistors with high accuracy (this method has been used in the past by one of the authors of this proposal).

Further, we will see if naive users notice important differences when they are looking through an HMD with head-mounted cameras, as opposed to an immersive omnidirectional video system. In the first case, the input video images to the HMD will be buffered in computer memory so that the end-to-end system lag exactly matches that of the immersive omnidirectional video system. We will match the contents of the visible scene, FOV and limit the field of regard so the two cases correspond.

4.5.2 Interpolated view-morph stereoscopy integration and evaluation

The goal is to show that stereoscopic views of a scene can be rendered by view-morph *interpolation* and that these derived images can be fused by human viewers to perceive the 3D depth of a scene. (View extrapolation will be tested later). Using human depth perception to test our fast view-morphing methods will provide an immediate indication of the success of our algorithms. We will integrate the preliminary MIT and CMU software components and 1) measure both overall system lag and the processing lag of the CMU view morphing software, and 2) assess the accuracy of the perceived stereo depth.

The end-to-end system lag will be measured again in order to determine the incremental cost of adding the algorithms to undistort the video images. To determine the accuracy of the derived stereoscopic view, we will perform a simple distance judgment experiment. In this experiment, luminous spheres will be displayed at varying distances from an observer in an otherwise darkened room. The spheres will vary in size such that the angular subtense of each sphere is constant at the observer's viewpoint. The dependent measure will be an magnitude estimate of the egocentric distance to each target sphere. We will ask users with normal stereo vision to compare 3 conditions: 1) normal stereo direct viewing (i.e., not using any display), 2) stereo HMD viewing using helmet mounted cameras, and 3) view-morph derived stereo HMD viewing.

4.5.3 Omnidirectional viewing demonstration

Our objective is to display a portion of a hemispheric video image captured with wide-angle optics on an HMD, updating the stereo morphed image pairs based on the user's head movement. We will test and measure the following system properties: 1) residual

static image distortion due to errors in the lens model, 2) processing lag of the algorithms to undistort the video images, and 3) compare results with “fish-eye”, cylindrical, and parabolic projections.

The end-to-end system lag will again be measured in order to determine the incremental cost of adding the algorithms to undistort the video images. We will again match both the end-to-end system lags and the field-of-regard of the video camera HMD to the omnidirectional video system and test whether users notice striking differences between the two display methods. However, the resolution of the video camera HMD must be decreased to mimic that of the omnidirectional system; this will be accomplished by capturing the video at an appropriately lower resolution and using a fast enlarging technique (e.g., pixel-doubling to fill the display area). A test of stereoscopic depth similar to that above will also be performed. Performance should be worse (show greater variance) with a wide angle lens for two reasons: 1) optical compression by the lens will have reduced the effective angular resolution, and 2) errors in the lens model will introduce distortions in the stereo disparity which in turn will result in depth errors.

4.5.4 Extrapolated view-morph integration and evaluation

The objective is now to integrate view morphing *extrapolation* from a single stereo head with the MIT omnidirectional viewer. As the virtual eye point is translated further from the real eye point, the match between the extrapolated image and a real image decreases. To characterize this, we will use view-morph algorithms to construct a progression of views of test scenes as the extrapolated viewpoint moves in both the horizontal and vertical directions. In addition to these extrapolated morphed views, we will capture real views from corresponding positions using a video camera. The evaluation will then consist of asking observers to judge the point in the sequence when it is clear that the two images are no longer identical, and why. We recognize that the extrapolation algorithm’s performance depends on scene content. We will use several different realistic scenes. This test will be done with normal optics and then repeated with extreme wide-angle lenses.

4.5.5 Augmented reality 3D registration

The objective is to assess registration errors between real and virtual images for 3D objects placed at different locations in the scene. The first step is implement 3D registration of CGI models in a captured video scene. To do this, we will use the lens models developed in 4.5.3 to match the 3D view frustum in the graphics generator. This also requires a 3D position/orientation sensor (e.g. Ascension Flock of Birds) to locate test objects in the camera FOV. Registration errors in video-based AR system are relatively easy to measure in the 2D picture plane. For example, by placing a target object at various locations in the scene, and using its sensed 3D position/orientation data to place a CGI twin, one need only record the rendered image and measure the distance between the real and virtual objects’ centers to characterize the errors. We will use

techniques developed by others to iteratively correct the system parameters until registration errors are minimized (Bajura, 1993; Bajura, Fuchs, & Ohbuchi, 1992; Tuceryan, Greer, Whitaker, Breen, Crampton, Rose, & Ahlers, 1995). We do not know ahead of time how effective the registration will be for both the near omnidirectional and view-morph systems—this is largely an empirical question. Our battery of experiments will allow us to empirically characterize these effects and explore methods to improve the system design.

4.5.6 Teleconferencing application evaluation

We want to evaluate the technique of making an HMD appear "transparent" to observers so they can sense an HMD user's eye-gaze direction. It is important to measure the accuracy of the rendered eye-gaze information. To do this, we will develop an eye-gaze direction test that will be conducted both in the real and virtual environments. In the "real" version of the test, a user will be seated looking at a transparent grid, while not wearing the HMD An observer will view a video monitor showing the user's image from a camera facing the user but on the opposite side of the transparent grid. Using a double-blind procedure, the user will fixate various locations on the grid and the observer will estimate the perceived grid fixation location. How accurately can the observer tell where the user is looking ? In the virtual version of the test, the user will wear an HMD. An identical "virtual" grid will be presented to the user through an HMD and on the monitor to the observers, ensuring that the real and virtual perspective images are equivalent. The same task will be repeated. Does the observer do as well as in the real version of the test?

4.5.7 Final integration of a Virtual Video System

For the final integration stage, we will temporarily relocate the CMU stereo head to the MIT laboratory in order to extend "Virtual Video" from more than one simultaneous viewpoint extrapolation sources (i.e., the two stereo heads illustrated in Fig. 4) and to merge each candidate view into a single combined view. Using a weighted Z-buffering technique, we will show how it is straightforward to extend the virtual video system so that the potential vantage locations span a greater volume and the potential viewing directions subtend a greater solid angle. We will evaluate this final system by testing how seamlessly the virtual vantage point can be made to move back and forth between the fields of each stereo head while changing in view direction.

5. References

Avidan, S. & Shashua, A. (1997). Novel view synthesis in tensor space. Proceedings of Computer Vision and Pattern Recognition Conference. pp. 1034-1040.

Azuma, R. T. (1997). A survey of augmented reality. Presence: Teleoperators and Virtual Environments, 6, 355-385.

Bajura, M. (1993). Camera calibration for video see-through head-mounted display. Technical Report TR93-048. Dept. Of Computer Science, University of North Carolina at Chapel Hill.

Bajura, M., Fuchs, H., & Ohbuchi, R. (1992). Merging virtual reality with the real world: Seeing ultrasound imageery within the patient. Computer Graphics (Proceedings of SIGGRAPH '92), 26, 203-210.

Beier, T. & Neely, S. (1992). Feature-based image metamorphosis. Proceedings of SIGGRAPH '92. pp. 35-42.

Beymer, D. & Poggio, T. (1996). Image representations for visual learning. Science, no.272. pp. 1905-1909.

Chen, S.E. & Williams, L. (1993). View interpolation for image synthesis. Proceedings of SIGGRAPH '93. pp. 279-288.

Chen, S.E. (1995). Quicktime VR --- An image-based approach to virtual environment navigation. Proceedings of SIGGRAPH '95. pp. 29-38.

Cohn, V. J., DiZio, P., & Lackner, J. R. (1996). Reaching movements during illusory self-rotation show compensation for expected Coriolis forces. Society for Neuroscience Abstracts, 22, 654.

Cruz-Neira, C., Sandin, T. A., & DeFantini, R. V. (1993). Surrounding screen projection-based virtual reality: the design and implementation of the cave. Proceedings of SGIGRAPH '93, 135-142.

Debevec, P.E., Taylor, C.J. & Malik, J. (1996). Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. Proceedings of SIGGRAPH '96. pp. 11-20.

Deriche, R., Zhang, Z., Luong, Q.T. & Faugeras, O. (1994). Robust recovery of the epipolar geometry for an uncalibrated stereo rig. Proceedings of European Conference on Computer Vision. pp. 567-576.

Ellis, S. R. (1991). Pictorial Communication in Virtual and Real Environments, London:Great Britain, Taylor & Francis.

Ellis, S. R., Dorighi, N. S., Menges, B. M., Adelstein, B. D., & Jacoby, R. H. (1997). In search of equivalence classes in subjective scales of reality. In M. Smith, G. Salvendy, & R. Koubek (Eds.), Design of computing systems: Social and ergonomic considerations (pp. 873-876). Amsterdam, Netherlands: Elsevier Science Publishers.

Ferrell, W. R. (1965). Remote manipulation with transmission delay, M.I.T., NASA TN D-2665.

Fisher, S. (1990). Virtual interface environments. In B. Laurel (Ed.), The Art of Human Computer Interface (pp. 423-438). Reading, MA: Addison-Wesley.

Gortler, S.J., Grzeszczuk, R., Szeliski, R. & Cohen, M.F. (1996). The lumigraph. Proceedings of SIGGRAPH '96. pp. 43-54.

Held, R., Efstathiou, A., & Greene. (1966). Adaptation to displaced and delayed visual feedback from the hand. Journal of Experimental Psychology, 72, 887-891.

Horry, Y., Anjyo, K. & Arai, K. (1997). Tour into the picture: Using a spidery mesh interface to make animation from a single image. Proceedings of SIGGRAPH '97. pp. 225-232.

IPIX, version 1.0. (1997). Interactive Pictures Corporation, Inc.

Irani, M., Anandan, P. & Hsu, S. (1995). Mosaic based representations of video sequences and their applications. Proceedings of Fifth International Conference on Computer Vision. pp. 605-611.

Kanade, T., Narayanan, P.J. & Rander, P.W. (1995). Virtualized reality: Concepts and early results. Proceedings of IEEE Workshop on Representation of Visual Scenes. pp. 69-76.

Kanade, T., Rander, P. & Narayanan, P.J. (1997). Virtualized reality: Constructing virtual worlds from real scenes. IEEE Multimedia, vol.4, no.1. pp. 34-46.

Katayama, A., Tanaka, K., Oshino, T. & Tamura, H. (1995). A viewpoint dependent stereoscopic display using interpolation of multi-viewpoint images. Proceedings of SPIE, Vol. 2409A. pp. 21-30.

Kumar, R., Anandan, P., Irani, M., Bergen, J. & Hanna, K. (1995). Representation of scenes from collections of images. Proceedings of IEEE Workshop on Representation of Visual Scenes. pp. 10-17.

Kutulakos, K. N. & Seitz, S. M. (1998c, submitted). A theory of shape by space carving. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998.

Laveau, S. & Faugeras, O. (1994). 3-D scene representation as a collection of images. Proceedings of International Conference on Pattern Recognition. pp. 689-691.

Lengyel, J. & Snyder, J. (1997). Rendering with coherent layers. Proceedings of SIGGRAPH ’97. pp. 233--242.

Levoy, M. & Hanrahan, P. (1996). Light field rendering. Proceedings of SIGGRAPH ‘96.

Lippman, A. (1980). Movie-maps: an application of the optical videodisc to computer graphics. Proceedings of SIGGRAPH ‘80. pp. 32-42.

Longuet-Higgins, H.C. (1981). A computer algorithm for reconstructing a scene from two projections. Nature, vol. 293. pp. 133-135.

Mann, S. & Picard, R. (1995). Video orbits of the projective group. A new perspective on image mosaicing. A. I. Memo, No. 338. MIT Media Lab, Cambridge, MA.

Mann, S. & Picard, R. (1997). Video orbits of the projective group: A simple approach to featureless estimation of parameters. IEEE Transactions on Image Processing, vol.6, no.9.

McMillan, L. & Bishop, G. (1995a). Head-tracked stereoscopic display using image warping. Proceedings of SPIE, Vol. 2409A. pp. 21-30.

McMillan, L. & Bishop, G. (1995b). Plenoptic modeling. Proceedings of SIGGRAPH ’95. pp. 39-46.

Moezzi, S., Katkere, A., Kuramura, D.Y. & Jain, R. (1996). Reality modeling and visualization from multiple video sequences. IEEE Computer Graphics and Applications, vol.16, no.6. pp. 58-63.

Nayer, S. K. (1998). Omnidirectional imaging apparatus. United States Patent 5,760,826.

Noro K, Kawai T., Takao, H. (1996). The development of a dummy head for 3-D audio visual recording for transmitting teleprsence. Ergonomics, 39, 1381-1389.

Ott, M., Lewis, J.P. & Cox, I. (1993). Teleconferencing eye contact using a virtual camera. Proceedings of INTERCHI `93, 109-110.

Pausch, R., Proffitt, D., & Williams, G. (1997). Quantifying immersion in virtual reality. Proceedings of SIGGRAPH’97, 13-18.

Poggio, T., Torre, V. & Koch, C. (1985). Computational vision and regularization theory. Nature, vol.317. pp. 314-319.

Poulton, E. C. (1974). Tracking Skill and Manual Control. New York: Academic Press.

QuickTime VR, version 1.0. (1995). Apple Computer, Inc.

RealSpace Viewer, version 2.0. (1997). Live Picture, Inc.

Regan, M. & Post, R. (1994). Priority rendering with a virtual reality address recalculation pipeline. Proceedings of SIGGRAPH '94. pp. 155-162.

Robert, L. (1997). Realistic scene models from image sequences. Proceedings of Imagina 97 Conference, Monaco. pp. 8-13.

Rock, I. (1981). Anorthoscopic perception. Scientific American, 244, 145-53.

Scharstein, D. (1996). Stereo vision for view synthesis. Proceedings of Computer Vision and Pattern Recognition Conference. pp. 852-858.

Seitz, S. M. (1997a). Bringing photographs to life with view morphing. Proceedings of Imagina 1997 Conference, Monaco, (pp. 153-158).

Seitz, S. M. (1998a). Toward interactive scene walkthroughs from images. Proceedings of IEEE and ATR Workshop on Computer Vision for Virtual Reality Based Human Communications, (pp. 14-19).

Seitz, S. M., & Dyer, C. R. (1994a). Affine invariant detection of periodic motion. Proceedings of Computer Vision and Pattern Recognition Conference, (pp. 970-975).

Seitz, S. M., & Dyer, C. R. (1994b). Detecting irregularities in cyclic motion. Proceedings of Workshop on Motion of Non-Rigid and Articulated Objects, (pp. 178-185).

Seitz, S. M., & Dyer, C. R. (1995a). Complete structure from four point correspondences. Proceedings of Fifth International Conference on Computer Vision, (pp. 330-337).

Seitz, S. M., & Dyer, C. R. (1995b). Physically-valid view synthesis by image interpolation. Proceedings of IEEE Workshop on Representation of Visual Scenes, (pp. 18-25).

Seitz, S. M., & Dyer, C. R. (1996a). View morphing. Proceedings of SIGGRAPH '96, (pp. 21-30).

Seitz, S. M., & Dyer, C. R. (1996b). Toward image-based scene representation using view morphing. Proceedings of International Conference on Pattern Recognition, (pp. 84-89).

Seitz, S. M., & Dyer, C. R. (1997a). Photorealistic scene reconstruction by voxel coloring. Proceedings of Computer Vision and Pattern Recognition Conference, (pp. 1067-1073).

Seitz, S. M., & Dyer, C. R. (1997b, in press). View-invariant analysis of cyclic motion. International Journal of Computer Vision, vol.25, no.3.

Seitz, S. M., & Dyer, C. R. (1997c). Cyclic motion analysis using the period trace. In Shah, M. & Jain, R. (Eds) Motion-Based Recognition (pp. 61-85). Kluwer Academic Publishers, Boston, MA.

Seitz, S. M., & Dyer, C. R. (1997d). View morphing: Uniquely predicting scene appearance from basis images. Proceedings of Image Understanding Workshop. pp. 881-887.

Seitz, S. M., & Kutulakos, K. N. (1998b). Plenoptic image editing. Proceedings of Sixth International Conference on Computer Vision, (pp. 17-24).

Sims, D. (1994). New realities in aircraft design and manufacture. IEEE Computer Graphics and Applications, 14, 91.

SmoothMove, version 1.0. (1997). Infinite Pictures, Inc.

Surround Video, version 1.0. (1997). Black Diamond, Inc.

Sutherland, I. (1968). A head-mounted three dimensional display. Proceedings of the Fall Joint Computer Conference, 757-764.

Szeliski, R. & Kang, S.B. (1995). Direct methods for visual scene reconstruction. Proceedings of IEEE Workshop on Representation of Visual Scenes.

Szeliski, R. & Shum, H.Y. (1997). Creating full view panoramic image mosaics and environment maps. Proceedings of SIGGRAPH '97. pp. 251-258.

Szeliski, R. (1996). Video mosaics for virtual environments. IEEE Computer Graphics and Applications,vol.16, no.2. pp. 22-30.

Tomasi, C. & Kanade, T. (1992). Shape and motion from image streams under orthography: A factorization method. International Journal of Computer Vision, vol.9, no.2. pp.137-154.

Torborg, J. & Kajiya, J.T. (1996). Talisman: Commodity realtime 3D graphics for the PC. Proceedings of SIGGRAPH '96. pp. 353-363.

Tuceryan, M., Greer, D. S., Whitaker, R. T., Breen, D., Crampton, C., Rose, E., & Ahlers, K. H. (1995). Calibration requirements and procedures for augmented reality. IEEE Transactions on Visualization and Computer Graphics, 1, 255-273.

Verri, A. & Poggio, T. (1989). Motion field and optical flow: Qualitative properties. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.11, no.5. pp. 490-498.

Vetter, T. & Poggio, T. (1996). Image synthesis from a single example image. Proceedings of European Conference on Computer Vision. pp. 653-659.

Werner, T., Hersch, R.D. & Hlavac, V., (1995). Rendering real-world objects using view interpolation. Proceedings of Fifth International Conference on Computer Vision. pp. 957-962.

Wickens, C. D. (1986). The effects of control dynamics on performance. In Boff, K. R., Kaufman, L., Thomas, J. P. (Eds), Handbook of Perception and Human Performance, Vol. II, Ch. 39. New York: John Wiley & Sons.

Yang, T. L., Wade, M. N., & Proffitt, T. L. (1997). Heights are overestimated more in real and immersive environments than in pictures. Paper presented at the Psychonomic Society meeting, Philadelphia, PA, November 20-23, 1997.

Young, L. R., & Stark, L. (1963). Variable feedback experiments testing a sampled data model for eye tracking movements. IEEE Transactions on Human Factors in Electronics, HFE-4, 38-51.

Zimmerman, S. D. (1993). Omniview motionles camera orientation system. United States Patent 5,185,667.

Appendix B - VFW/OpenGL API

The VFW/OpenGL API consists of two programs, "Newtest.cxx" and "320x240newquadhemiTexview.c", and they are listed separately in this appendix:

Newtest.cxx

The primary functions of this program are to grab a frame buffer from the video camera, one frame at a time, and then "transfer" that buffer via a global variable, called "HutchVizData", to a program called, "320x240newquadhemiTexview.c" to render the video frame buffer to the final output hemispheric polygon. This program utilizes Video-for-Windows (VFW) calls to control the frame processing and its structure was formulated largely by trial-by-error and from what little documentation that could be found. Once the necessary windows have been created and the connection with the frame grabber's driver has been opened, for each input frame event, the frame buffer is written to the global variable, "HutchVizData". Then control is shifted to the entry point, HandleRendering(), of the program, "320x240newquadhemiTexview.c" to render the video frame buffer.

```
// Bill Hutchison, MVL, MIT
// Latest Change 8/3/99


#include <windows.h>
#include <windowsx.h>
#include <commdlg.h>
#include <mmsystem.h>
#include <mmreg.h>
#include <io.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <dos.h>
#include <shellapi.h>
#include <vfw.h>

#include <conio.h>
#include <memory.h>

#include <gl\gl.h>            // OpenGL
#include <gl\glu.h>           // GLU library


#define FPS_TO_MS(f)          ((DWORD) ((double)1.0e6 / f))



HWND  hWndC;
HWND    ghWndMain;
CAPSTATUS     gCapStatus ;
CAPDRIVERCAPS  gCapDriverCaps ;
CAPTUREPARMS   gCapParms ;
```

```
char          gachAppName[]  = "vidcapApp" ;
char          gachAppTitle[20] = "Test";    //VidCap
int nID, framecount;

static      gfIsRTL = 0;

extern "C"  void HandleRendering();
extern "C"  void HandleRendering2();
extern "C"          HWND InitOGL2(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow);
extern "C" void InitOGL(void);

extern "C" GLubyte *HutchVizData;
extern "C" GLubyte *HutchVizData1;



//
// MainWndProc: Application Main Window Procedure
//
LONG FAR PASCAL vidframeProc(HWND hWnd, UINT Message, UINT wParam, LONG lParam)
{
//////////////////////////////////////////////////////////////////////
//  hWnd:      Application main window handle
//  Message:   Next message to be processed
//  wParam:    WORD param for the message
//  lParam:    LONG param for the message
//////////////////////////////////////////////////////////////////////

            //return 0L;

   switch (Message) {

      static BOOL fMinimized;

      case WM_SYSCOMMAND:
            if ((wParam & 0xfff0) == SC_MAXIMIZE)
                  fMinimized = FALSE;
            else if ((wParam & 0xfff0) == SC_RESTORE)
                  fMinimized = FALSE;
            else if ((wParam & 0xfff0) == SC_MINIMIZE)
                  fMinimized = TRUE;
            return DefWindowProc(hWnd, Message, wParam, lParam);
            break;


      default:
         return DefWindowProc(hWnd, Message, wParam, lParam) ;
   }

   return 0L;
}   // End of MainWndProc
```

```
#include <time.h>

// This procedure saves the frame buffer to the global variable, HutchVizData
void
saveMyFrame(void *buffer, int size)
{

        int fOut, result;
        static firstTime = 1, firstTime2 = 1;
        static FILE  *dOut;

        HutchVizData = (unsigned char *)buffer;
        fOut = _open("mydata0.raw", _O_BINARY | _O_CREAT | _O_WRONLY );
        result = _write(fOut, buffer, size);
        _close(fOut);

}


LRESULT CALLBACK myFrameProc(HWND hWnd, LPVIDEOHDR lpVHdr)
{

        saveMyFrame( (void *)lpVHdr->lpData, lpVHdr->dwBytesUsed );

   return (LRESULT) TRUE;
}


int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int
nCmdShow)

{

//HWND hwnd, hwndCap;
        WNDCLASS wc;
   static BOOL bInitDone = FALSE;
        CHAR szAppName[]="tex1";
        int                     result;
        int     i;
        static FILE  *dOut1;
        static firstTime3 = 1;
        LPBITMAPINFOHEADER lpbi;
        int sz, fsz;
        HWND  ghWnd;

        //Create Main Window
        ghWnd = InitOGL2(hInstance, hPrevInstance, lpszCmdLine, nCmdShow);
        InitOGL();
   if (!bInitDone) {
      WNDCLASS wc;


      if (!hPrevInstance) {
         // If it's the first instance, register the window class
```

```
        wc.lpszClassName = szAppName;
        wc.hInstance     = hInstance;
        wc.lpfnWndProc   = vidframeProc;
        wc.hCursor       = LoadCursor(NULL, IDC_ARROW) ;
        wc.hIcon         = NULL;
        wc.lpszMenuName  = NULL;
        wc.hbrBackground = GetStockObject(WHITE_BRUSH);
        wc.style         = CS_HREDRAW | CS_VREDRAW ;
        wc.cbClsExtra    = 0 ;
        wc.cbWndExtra    = 0 ;

        if(!RegisterClass(&wc)) {
          return(NULL);
        }
      }
      bInitDone = TRUE;
    }

    ghWndMain = CreateWindowEx(
            gfIsRTL ? WS_EX_LEFTSCROLLBAR | WS_EX_RIGHT | WS_EX_RTLREADING : 0,
            szAppName,
            NULL,
           WS_CHILD|WS_VISIBLE|WS_HSCROLL|WS_VSCROLL|WS_CLIPCHILDREN,
                                    CW_USEDEFAULT, CW_USEDEFAULT,
                                    CW_USEDEFAULT, CW_USEDEFAULT,
                                    //0, 0, 640, 480,
            //x, y, cx, cy,
                                    ghWnd,
                                    //NULL,
            (HMENU) 0,
            hInstance,
            NULL);

    if (ghWndMain == NULL) {
      return(NULL);
    }
///



    if (! hPrevInstance) {
      // If it's the first instance, register the window class
      wc.lpszClassName = gachAppName ;
      wc.hInstance     = hInstance ;
      wc.lpfnWndProc   = MainWndProc ;
      wc.hCursor       = LoadCursor(NULL, IDC_ARROW) ;
      wc.hIcon         = NULL;
      wc.lpszMenuName  = NULL;
      wc.hbrBackground = GetStockObject(WHITE_BRUSH) ;
      wc.style         = CS_HREDRAW | CS_VREDRAW ;
      wc.cbClsExtra    = 0 ;
      wc.cbWndExtra    = 0 ;

      if (!(result = RegisterClass(&wc))) {
        return(FALSE);
```

```
    }
        }


        // Create Frame's window
  ghWndMain = CreateWindowEx(
        gfIsRTL ? WS_EX_LEFTSCROLLBAR | WS_EX_RIGHT | WS_EX_RTLREADING : 0,
        gachAppName,
        gachAppTitle,
        WS_CAPTION     |
        WS_SYSMENU     |
        WS_MINIMIZEBOX  |
        WS_MAXIMIZEBOX  |
        WS_THICKFRAME   |
        WS_CLIPCHILDREN |
        WS_OVERLAPPED,
        CW_USEDEFAULT, CW_USEDEFAULT,
        CW_USEDEFAULT, CW_USEDEFAULT,
        NULL,
        NULL,
        hInstance,
        0) ;


        hWndC = capCreateCaptureWindow ( "Capture Window",
                WS_CHILD , 0, 0, 640, 480, ghWnd, nID);

        capCaptureGetSetup(hWndC, &gCapParms, sizeof(CAPTUREPARMS));
        gCapParms.dwRequestMicroSecPerFrame = FPS_TO_MS(30);

// For each frame event, save the frame buffer to HutchVizData in the procedure,
// "myFrameProc"
        capSetCallbackOnFrame(hWndC, myFrameProc);

// Connect to Frame Grabber Driver
        capDriverConnect(hWndC, 0);

// For each frame enter program, "320x240newquadhemiTexview.c" to render
the video frame buffer to the final output hemispheric polygon

        while (framecount <= 150) {
                if (capGrabFrameNoStop(hWndC)) {
                HandleRendering();
                framecount = framecount + 1;

                }
                else
                        return(FALSE);

        }

        capCaptureStop(hWndC);
        capDriverDisconnect (hWndC);

        return TRUE;
```

}

320x240newquadhemiTexview.c

The primary functions of this program are to render the video frame texture to the hemispheric polygon, and to the translate and rotate the output image based on inputs from the head tracker. This program utilizes OpenGL commands and its structure was adapted from Dr. Andy Beall, MVL, MIT.

This program is initiated from "newtest.cxx" (the corresponding VFW program) , once it is ready to begin the render processing. The input video image frame buffer obtained from "newtest.cxx" is obtained from a global variable called "HutchVizData". The majority of the commands for this program describe the mapping of geometric locations from the video frame texture to the hemispheric polygon. These 'mapping' commands are produced in a Matlab script called, 'hemiquadtransform.m'.

```
// Bill Hutchison, MVL, MIT
// Latest Change 8/3/99

#include <windows.h>          // Window defines
#include <gl\gl.h>            // OpenGL
#include <gl\glu.h>           // GLU library
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/* Windows globals, defines, and prototypes */
CHAR szAppName[]="tex1";
HWND  ghWnd;
HDC   ghDC;
HGLRC ghRC;

extern void InitializeSensor(void);
extern void UpdateSensor(void);
extern void ApplyNullOrientation(void);

GLubyte *HutchVizData;
GLubyte *HutchVizData1;

float myYaw, myPitch, myRoll;

 // Define width and height of frame buffer
#define WIDTH  320
#define HEIGHT 240

LONG WINAPI MainWndProc (HWND, UINT, WPARAM, LPARAM);
BOOL bSetupPixelFormat(HDC);

GLvoid resize(GLsizei, GLsizei);
GLvoid initializeGL(GLsizei, GLsizei);
```

```
GLvoid drawScene(GLvoid);
void initLighting(void);
void setTransform(void);


GLint ell;
GLfloat dist=0.0;

// This is the entry point, from "newtest.cxx" (the VFW program) , to begin render processing in OpenGL

void
HandleRendering(void)
{

  //MVL Goni
  static float xpos = 0.0, ypos = 0.0, zpos = -2.0;//zpos = 7.0;
  static char firstTime = TRUE;
  static char firstTime2 = TRUE;
  static int width, height, depth;
  static int xoffset, yoffset;
  int i, j;
  static int framecnt = 0;
  static GLubyte *rescaledData;
  static GLubyte *data;

  // Height and width of texture area
    width = 1024;
  height = 1024;
  depth = 3;

// First time through program intialize head tracker sensor and create texture area limits

  if (firstTime2) {

            InitializeSensor();
            data = malloc(width * height * depth);

// Zero out texture area
            for (i=0; i<height; i++) {
                      for (j=0; j<width; j++) {
                        data[(i + j*width)] = 0;
             }
            }

             firstTime2 = FALSE;
  }

// First time through define the texture area

  if (firstTime) {

           glPixelStorei(GL_UNPACK_SWAP_BYTES, 1);
           glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
           glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
           glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

```
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

            glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height,
             0, GL_RGB, GL_UNSIGNED_BYTE, data);

   firstTime = FALSE;
  }


// Associate the video frame buffer with the texture area

    xoffset = 0; yoffset = 0;
    glTexSubImage2D(GL_TEXTURE_2D, 0, xoffset, yoffset, 320, 240,
                                GL_RGB, GL_UNSIGNED_BYTE, HutchVizData);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

// Update head tracker inputs
  glLoadIdentity();
  UpdateSensor();
  ApplyNullOrientation();
  glRotatef(myRoll, 0, 0, 1);
  glRotatef(myYaw, 0, 1, 0);
  glRotatef(-myPitch + 90, 1, 0, 0);
  glTranslatef(xpos, ypos, zpos);


// Render video frame texture to hemipsheric polygon
  glPushMatrix();
  glEnable(GL_TEXTURE_2D);
  glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
  glBegin(GL_QUADS);

  glColor3f(1,1,1);

// The following commands are produced by the Matlab script, 'hemiquadtransform.m'.  A large number
// of the almost 1400 comands have been removed the sake of brevity.

            glTexCoord2d(0.15859,0.11719);
            glVertex3d(-0.21988,0,6.9965);
            glTexCoord2d(0.16422,0.11719);
            glVertex3d(-0.74628,0,6.9601);
            glTexCoord2d(0.1641,0.11857);
            glVertex3d(-0.73494,-0.12959,6.9601);
            glTexCoord2d(0.15856,0.11759);
            glVertex3d(-0.21653,-0.038181,6.9965);
            glTexCoord2d(0.15856,0.11759);

            glTexCoord2d(0.24423,0.043367);
            glVertex3d(-5.3597,4.4973,0.21988);
            glTexCoord2d(0.23992,0.046983);
            glVertex3d(-5.3318,4.4739,0.74628);
            glTexCoord2d(0.23992,0.046983);
            glVertex3d(-5.3318,4.4739,0.74628);
            glTexCoord2d(0.24423,0.043367);
            glVertex3d(-5.3597,4.4973,0.21988);
```

```
            glTexCoord2d(0.25571,0.059766);
            glVertex3d(-6.0592,3.4983,0.21988);
            glTexCoord2d(0.25084,0.062578);
            glVertex3d(-6.0276,3.4801,0.74628);
            glTexCoord2d(0.25084,0.062578);
            glVertex3d(-6.0276,3.4801,0.74628);
            glTexCoord2d(0.25571,0.059766);
            glVertex3d(-6.0592,3.4983,0.21988);
            glTexCoord2d(0.26417,0.077909);
            glVertex3d(-6.5746,2.393,0.21988);
            glTexCoord2d(0.25888,0.079832);
            glVertex3d(-6.5404,2.3805,0.74628);
            glTexCoord2d(0.25888,0.079832);
            glVertex3d(-6.5404,2.3805,0.74628);
            glTexCoord2d(0.26417,0.077909);
            glVertex3d(-6.5746,2.393,0.21988);
            glTexCoord2d(0.26935,0.097245);
            glVertex3d(-6.8903,1.2149,0.21988);
            glTexCoord2d(0.26381,0.098222);
            glVertex3d(-6.8544,1.2086,0.74628);
            glTexCoord2d(0.26381,0.098222);
            glVertex3d(-6.8544,1.2086,0.74628);
            glTexCoord2d(0.26935,0.097245);
            glVertex3d(-6.8903,1.2149,0.21988);
            glTexCoord2d(0.27109,0.11719);
            glVertex3d(-6.9965,1.7137e-015,0.21988);
            glTexCoord2d(0.26547,0.11719);
            glVertex3d(-6.9601,1.7047e-015,0.74628);



  glEnd();
  glFlush();
  glDisable(GL_TEXTURE_2D);

  glPopMatrix();

  SwapBuffers(ghDC);
}



BOOL bSetupPixelFormat(HDC hdc)
{
   PIXELFORMATDESCRIPTOR *ppfd;
   int pixelformat;

            PIXELFORMATDESCRIPTOR pfd = {
   sizeof(PIXELFORMATDESCRIPTOR),  //  size of this pfd
   1,                     // version number
   PFD_DRAW_TO_WINDOW |   // support window
   PFD_SUPPORT_OPENGL |    // support OpenGL
   PFD_DOUBLEBUFFER,      // double buffered
   PFD_TYPE_RGBA,         // RGBA type
```

```
    24,                 // 24-bit color depth
    0, 0, 0, 0, 0, 0,     // color bits ignored
    0,                  // no alpha buffer
    0,                  // shift bit ignored
    0,                  // no accumulation buffer
    0, 0, 0, 0,           // accum bits ignored
    32,                 // 32-bit z-buffer
    0,                  // no stencil buffer
    0,                  // no auxiliary buffer
    PFD_MAIN_PLANE,       // main layer
    0,                  // reserved
    0, 0, 0             // layer masks ignored
    };

            pfd.cColorBits = GetDeviceCaps(ghDC,BITSPIXEL);

            ppfd = &pfd;

    pixelformat = ChoosePixelFormat(hdc, ppfd);

    if ( (pixelformat = ChoosePixelFormat(hdc, ppfd)) == 0 )
    {
        MessageBox(NULL, "ChoosePixelFormat failed", "Error", MB_OK);
        return FALSE;
    }

    if (SetPixelFormat(hdc, pixelformat, ppfd) == FALSE)
    {
        MessageBox(NULL, "SetPixelFormat failed", "Error", MB_OK);
        return FALSE;
    }

    return TRUE;
}


HWND
InitOGL2(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    WNDCLASS wc;
            MSG       msg;
            RECT rect;
            static     gfIsRTL = 0;
            char          gachAppName[]  = "vidcapApp" ;
            char          gachAppTitle[20] = "Test";    //VidCap


    if (! hPrevInstance) {
        // If it's the first instance, register the window class
        wc.lpszClassName = gachAppName ;
        wc.hInstance     = hInstance ;
        wc.lpfnWndProc   = MainWndProc ;
        wc.hCursor       = LoadCursor(NULL, IDC_ARROW) ;
        wc.hIcon         = NULL;
        wc.lpszMenuName  = NULL;
```

```
    wc.hbrBackground = GetStockObject(WHITE_BRUSH) ;
    wc.style         = CS_HREDRAW | CS_VREDRAW ;
    wc.cbClsExtra    = 0 ;
    wc.cbWndExtra    = 0 ;



    if (!RegisterClass(&wc)) {
       return NULL;
    }
        }



        MSG        msg;
  WNDCLASS   wndclass;
        RECT rect;

  // Register the Main class
  wndclass.style         = 0;
  wndclass.lpfnWndProc   = (WNDPROC)MainWndProc;
  wndclass.cbClsExtra    = 0;
  wndclass.cbWndExtra    = 0;
  wndclass.hInstance     = hInstance;
  wndclass.hIcon         = LoadIcon (hInstance, szAppName);
  wndclass.hCursor       = LoadCursor (NULL,IDC_ARROW);
  wndclass.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
//   wndclass.hbrBackground = GetStockObject(BLACK_BRUSH);
  wndclass.lpszMenuName  = szAppName;
  wndclass.lpszClassName = szAppName;

  if (!RegisterClass (&wndclass) )
     return FALSE;



        // specify exact client size

        rect.left = 5;
        rect.top = 20;
        rect.right = 1020;
        rect.bottom = 760;


        AdjustWindowRect(&rect,WS_OVERLAPPEDWINDOW | WS_CLIPSIBLINGS |
WS_CLIPCHILDREN,TRUE);


  /* Create the Main Window
  ghWnd = CreateWindow (szAppName,
        "View",
     WS_OVERLAPPEDWINDOW | WS_CLIPSIBLINGS | WS_CLIPCHILDREN,
        rect.left,
        rect.top,
        rect.right - rect.left,
```

```
            rect.bottom - rect.top,
            NULL,
            NULL,
            hInstance,
            NULL);

    // make sure window was created
    if (!ghWnd)
       return FALSE;
            */// end new delete

 // Create Application's Main window from Vidcap.c
    ghWnd = CreateWindowEx(
          gfIsRTL ? WS_EX_LEFTSCROLLBAR | WS_EX_RIGHT | WS_EX_RTLREADING : 0,
          gachAppName,
          gachAppTitle,
          WS_CAPTION     |
          WS_SYSMENU     |
          WS_MINIMIZEBOX |
          WS_MAXIMIZEBOX |
          WS_THICKFRAME  |
          WS_CLIPCHILDREN |
          WS_OVERLAPPED,
          rect.left,
          rect.top,
          rect.right - rect.left,
          rect.bottom - rect.top,
          NULL,
          NULL,
          hInstance,
          0) ;

    if (ghWnd == NULL) {
       return NULL;
    }


    /* show and update main window */
    ShowWindow (ghWnd, nCmdShow);

    UpdateWindow (ghWnd);

//          return msg.wParam;
            return ghWnd;
}

/* main window procedure */

LONG WINAPI MainWndProc (
    HWND    hWnd,
    UINT    uMsg,
    WPARAM  wParam,
    LPARAM  lParam)
{
    LONG    lRet = 1;
```

```
    PAINTSTRUCT    ps;
    RECT rect;
            POINT pt;
            char str[80];

    switch (uMsg) {

    case WM_CREATE:
       ghDC = GetDC(hWnd);
       if (!bSetupPixelFormat(ghDC))
          PostQuitMessage (0);

       ghRC = wglCreateContext(ghDC);
       wglMakeCurrent(ghDC, ghRC);
       GetClientRect(hWnd, &rect);

       break;

    default:
       lRet = DefWindowProc (hWnd, uMsg, wParam, lParam);
       break;
    }

    return lRet;
}



void
InitOGL(void)
{

  /*
  glutInitDisplayMode(GLUT_RGB |
                                GLUT_DOUBLE |
                                GLUT_DEPTH |
                                GLUT_MULTISAMPLE);


  glutCreateWindow("Hutch VR");

  */

  /* Register callback routines using GLUT */
  //glutReshapeWindow(500, 500);
  //glutDisplayFunc(HandleRendering);
  //glutIdleFunc(HandleRendering);
  //glutKeyboardFunc(HandleKeyboard);
  //glutSpecialFunc(HandleSpecial);



  /* Initialize default rendering parameters */
  glEnable(GL_DEPTH_TEST);
  glEnable(GL_POINT_SMOOTH);
```

```
  glEnable(GL_BLEND);
  glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);


  glMatrixMode(GL_PROJECTION);
//  gluPerspective(21.2, 1.26, .1, 1000);
  gluPerspective(48, 1.33, .1, 1000);

  glMatrixMode(GL_MODELVIEW);

  //glutMainLoop();

 }
```

Appendix C- Matlab Script

Hemiquadtransform.m

This Matlab script produces the commands for the "320x240newquadhemiTexview.C" program. There are a significant number of commands produced (on the order of 1400) which in their entirety describe the correpsondence between the distorted circular texture (video frame buffer) and the truncated hemispheric polygon. One set are 2-D texture commands that are associated with the video frame buffer, the other set are 3-D commands that correspond to the hemispheric polygon.

```
% Bill Hutchison, MVL, MIT
% Latest Change 8/3/99

outcommands = [];
% Define the number intervals (thetastep) to rotate about the video frame buffer
% This number gives the horizontal resolution of the hemispheric polygon
% Theta must begin and end at 0 to 360, respectively
thetabegin = 0; thetastep = 10; thetaend = 360;
% Define the resolution limits of the video frame buffer
subimagewidth = 320; subimageheight = 240;
% Define the resolution limits of the texture area
framewidth = 1024; frameheight = 1024;
% Define the upper and lower limits of the video frame buffer
clipmax = 0.98; clipmin = 0.02;
% Define the center of the video frame buffer
xoffset = 0.5*subimagewidth/framewidth;
yoffset = 0.5*subimageheight/frameheight;
% Define the min and max limits of the radius of video frame buffer
rmin = clipmin*0.5*subimageheight/frameheight;
rmax = clipmax*0.5*subimageheight/frameheight;
% Define the number intervals for the radius of video frame buffer
% This number gives the vertical resolution of the hemispheric polygon
rintervals = 20;

rstep = (rmax - rmin)/rintervals;
deg2rad = pi/180;

% Produce Texture commands that describe a particular location on the video frame buffer

for r = rmin:rstep:rmax
        for theta = thetabegin:thetastep:thetaend
    x = xoffset + r*cos(theta*deg2rad);
        y = yoffset + r*sin(theta*deg2rad);
    outcommands = strvcat(outcommands, strcat('glTexCoord2d(',num2str(x),...
      ',',num2str(y),');'));
  end
end

% Define the dimensions of the Hemispheric Polygon
% R = radius, Z = depth, phi = vertical angular displacement, theta = horizontal
% angular displacement
```

```
R = 7;
Z = 7;
phibegin = clipmin*-90; phiend = clipmax*-90;
phistep = (phiend - phibegin)/rintervals;

% Produce the associated commands that describe a particular location on the
% hemispheric polygon

for phi = phibegin:phistep:phiend
  for theta = thetabegin:thetastep:thetaend
    x = R*sin(phi*deg2rad)*cos(theta*deg2rad);
    y = R*sin(phi*deg2rad)*sin(theta*deg2rad);
    z = R*cos(phi*deg2rad);
    outcommands = strvcat(outcommands, strcat('glVertex3d(',num2str(x),...
      ',',num2str(y),',',num2str(z),');'));
  end
end

% Sort & Match Vertices to Texture
newcommand = [];
texind = strmatch('glTex',outcommands);
verind = strmatch('glVer',outcommands);
for ti = 1:length(texind)
  newcommand((2*ti)-1,:) = outcommands(texind(ti),:);
end
for vi = 1:length(verind)
  newcommand(2*vi,:) = outcommands(verind(vi),:);
end

% Put commands in Index Order
newestcommand = [];
lengthz = (phiend - phibegin)/phistep + 1;
lengththeta = (thetaend - thetabegin)/thetastep + 1;

for zcnt = 0:lengthz-2
  for theta = thetabegin:thetastep:thetaend-thetastep
    index = 2*((zcnt*lengththeta)+(theta/thetastep));
    index1 = 2*(((zcnt+1)*lengththeta)+(theta/thetastep));
    index+1;
    index+2;
    index1+1;
    index1+2;
    index1+3;
    index1+4;
    index+3;
    index+4;

        newestcommand = strvcat(newestcommand, strvcat(newcommand(index+1,:),...
                                newcommand(index+2,:)));
        newestcommand = strvcat(newestcommand, strvcat(newcommand(index1+1,:),...
                                newcommand(index1+2,:)));
    newestcommand = strvcat(newestcommand, strvcat(newcommand(index1+3,:),...
                                newcommand(index1+4,:)));
    newestcommand = strvcat(newestcommand, strvcat(newcommand(index+3,:),...
                                newcommand(index+4,:)));
```

```
  end


end


% Write to File
% The commands from the file will be incorporated into the program,
% "320x240newquadhemiTexview.C"

fid = fopen('n:/hutch/vidcap1/undistort/transform/code.txt','w');
[numLines numChars] = size(newestcommand);
for lineCnt = 1:numLines,
  for charCnt = 1:numChars,
    fprintf(fid,'%c',newestcommand(lineCnt, charCnt));
  end
  fprintf(fid,'\n');
end

fclose(fid);
```