SANDIA REPORT

SAND2005-5206

Unlimited Release

Printed August 2005

Group Tele-Immersion:

Enabling natural interactions between groups at distant sites

Andrew Nashel

Department of Computer Science

The University of North Carolina at Chapel Hill

Chapel Hill, NC 27599-3175

Christine Yang

Advanced Software R&D

Corbin Stewart

Videoconference & Collaborative Technologies

Sandia National Laboratories

P.O. Box 969

Livermore, CA 94551-0969

Prepared by

Sandia National Laboratories

Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.doe.gov/bridge>

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: <http://www.ntis.gov/ordering.htm>

SAND2005-5206
Unlimited Release
Printed August 2005

Group Tele-Immersion: Enabling natural interactions between groups at distant sites

Andrew Nashel
Department of Computer Science
The University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175

Christine Yang
Advanced Software R&D

Corbin Stewart
Videoconference & Collaborative Technologies

Sandia National Laboratories
P. O. Box 969
Livermore, CA 94551-0969

Abstract

We present techniques and a system for synthesizing views for video teleconferencing between small groups. In place of replicating one-to-one systems for each pair of users, we create a single unified display of the remote group. Instead of performing dense 3D scene computation, we use more cameras and trade-off storage and hardware for computation. While it is expensive to directly capture a scene from all possible viewpoints, we have observed that the participants' viewpoints usually remain at a constant height (eye level) during video teleconferencing. Therefore, we can restrict the possible viewpoint to be within a virtual plane without sacrificing much of the realism, and in doing so we significantly reduce the number of required cameras. Based on this observation, we have developed a technique that uses light-field style rendering to guarantee the quality of the synthesized views, using a linear array of cameras with a life-sized, projected display. Our full-duplex prototype system between Sandia National Laboratories, California and the University of North Carolina at Chapel Hill has been able to synthesize photo-realistic views at interactive rates, and has been used to video conference during regular meetings between the sites.

This page intentionally left blank.

Contents

Preface	6
1 Introduction	7
2 Related Work	10
3 Line Light Field Rendering	11
3.1 Rendering	11
3.2 Sampling Requirement Analysis	13
4 Implementation and Results	15
4.1 System Architecture	15
4.2 Interactive Results	18
5 Discussion and Conclusion	20
References	21

Appendix

A Implementation Details	25
B Hardware and Software	29

Figures

1	Our vision of a 3D video teleconferencing session. The left photo shows the remote participants; the right photo shows the local participants and a life-size, static seamless image synthesized using our method in Section 3.	7
2	The linear camera array used to capture the conference participants.	8
3	Bird's eye view of the relative positions of local and remote participants in our system.	9
4	Angles between desired view camera C_v and texture cameras C_i , at a vertex V on the focal plane.	12
5	Error Analysis for Creating Orthogonal Views	14
6	The system architecture of our prototype.	15
7	Triggering and synchronization timeline for a group of cameras.	16
8	Data path for a single video stream, from capture to display.	17
9	View dependent effects when the local conferees move to different spots; the projected images show the synthesized views.	18
10	A live teleconferencing session between the University of Kentucky and the University of North Carolina at Chapel Hill.	19

Preface

This report describes ongoing work funded at the University of North Carolina at Chapel Hill under an ASC Tri-Lab University Partnership contract. The goal of this research is to prototype a large format, 2D teleconferencing system that can be deployed and used on a regular basis. The system is intended to provide an immersive conferencing environment for small groups as if they were sitting across the table from each other. Two copies of the system were built at Sandia National Laboratories, California (SNL/CA) in order to prototype local, full duplex operation. Andrew Nashel worked on the prototype systems and rendering algorithms at SNL/CA over three summers as part of his research effort in this area.

Group Tele-Immersion:

Enabling natural interactions between groups at distant sites

1 Introduction

With recent rapid advances in network bandwidth and dropping costs for video equipment, video teleconferencing has been widely deployed for business and education, enabling face-to-face communication between people at remote sites. However, existing systems are limited to capturing and displaying a single two-dimensional image, which does not provide a compelling or convincing presence to the participants [32]. The fixed viewpoint, lack of depth cues, and smaller than life-size imagery are far from replicating the experience of face-to-face conversation. Further challenges arise when groups of people try to communicate with video conferencing. Eye contact is difficult because each participant must look at the same camera, and each person appears smaller in the video image.

Figure 1. Our vision of a 3D video teleconferencing session. The left photo shows the remote participants; the right photo shows the local participants and a life-size, static seamless image synthesized using our method in Section 3.

Motivated by the shortcomings in current teleconferencing systems, our long-term goal is to create a system for teleconferencing in which remote participants can be visualized as if they were sitting across the table, creating the impression of face-to-face conversation [26]. We present our vision of a multi-user video teleconferencing session in Figure 1. The realization of this interface poses significant scientific and engineering challenges in computer graphics (scene visualization), computer vision (scene acquisition), and networking (scene transmission). In the past, we have demonstrated displays that present life-size, three-dimensional images [25, 4] to a single tracked user, given a computer generated model or a pre-acquired environment model through laser scan.

This paper focuses on the more challenging scene acquisition problem of how to capture a 3D environment real time, and how to transmit and display such data. Traditionally, a 3D model needs to be reconstructed from 2D images, and this model can then be visualized using classic computer graphics techniques. This reconstruction problem is one of the central topics in computer vision and remains open. While many algorithms exist, they are quite fragile in practice [17]. In addition, almost all of the highest quality algorithms are computationally expensive, and are not practical in real-time applications like 3D interactive video teleconferencing.

Figure 2. The linear camera array used to capture the conference participants.

In this paper, we present an approach that partly circumvents the difficult 3D reconstruction problem, while still conveying the sense of 3D in the output view. Based on light-field rendering (LFR) techniques [17], the basic idea is to use many cameras to record the flow of light in all directions, and thus the task of view synthesis becomes a simple table-lookup of the view rays. LFR typically requires hundreds or even thousands of cameras to cover the scene. For our teleconferencing setup, we observed that during a video teleconferencing session, a participant’s view point is quite limited, usually at the eye level with small lateral motions, looking towards the other participants. Thus, we can use a 1D linear array of cameras, shown in Figure 2, to capture a compact light field, which we refer to as the *Line Light Field*. This compact video representation makes real-time capture, transmission, and rendering possible. The second component of the system is a dual projector display that allows for life size rendering of the participants.

We show the relative positions of local and remote participants in a conferencing session in Figure 3. On the display (or local) side, the participants are seated in front of the projection display surface as if they are on one side of a table, facing the image of the remote group. At the remote site, the participants are captured by the linear array of cameras. Because the display and camera array are replicated at each site, the system is full duplex.

The contributions of the Group Tele-Immersion project are:

- A novel technique that uses a linear array of cameras for real-time view synthesis and an analysis of the sampling requirement, i.e., how many cameras are necessary for a given scene and viewing volume.
- The software and hardware architecture to collect, process, and transmit many simultaneous, frame synchronized video streams within a local network and across a wide area network.

Figure 3. Bird's eye view of the relative positions of local and remote participants in our system.

- A prototype system that has been setup with nodes at the University of North Carolina at Chapel Hill, Sandia National Laboratories, California, and the University of Kentucky, allowing for full duplex communication between two sites.

The remainder of this paper is organized as follows: we first present an overview of related work in real-time 3D scene acquisition and its application in teleconferencing in Section 2. The technical details of our *Line Light Field* approach are discussed in Section 3. We present our prototype system and results in Section 4, and conclude in Section 5 with a discussion of the pros and cons of the technique. We have also included technical appendices with details of the system's hardware and software design.

2 Related Work

3D video teleconferencing, as a natural extension to its mainstream 2D counterpart, has been pursued by many researchers and engineers. Many current research or commercial systems that claim to support 3D teleconferencing are in fact using a 2.5D approach called bill-boarding, where the input video streams are posted on 2D planes [1, 27]. Alternatively, there are *avatar* based systems for simple 3D teleconferencing where virtual figures represent participants [7, 5]. While these two approaches may provide some three-dimensional cues about the spatial relationship of the participants, neither can achieve a level of realism that is comparable to traditional 2D teleconferencing.

The bottleneck in 3D teleconferencing is model acquisition. A classic way to acquire 3D information from 2D images is the stereovision technique. Stereovision is one of the oldest and most active research topics in computer vision (see [28] for a recent survey). While many stereo algorithms obtain high-quality results by performing global optimizations, today only correlation-based stereo algorithms are able to provide a dense (per pixel) depth map in real time. Correlation-based algorithms can be accelerated using either special hardware [8, 13, 31, 14, 6] or assembly level instruction optimization (such as Intel's MMX and SSE extension sets) [23, 11, 10, 24]. Overall these algorithms are quite fragile in practice. The calculated depth can contain substantial outliers due to scene lighting, occlusions, and specular highlights. Increasing the fidelity of scene acquisition leads to higher reconstruction latency and lower frame rates [22].

An alternative reconstruction technique that is amenable to real-time computation is to use the silhouette information to construct the object's *visual hull*, which can be thought as a conservative shell that encloses the actual object [16]. Visual computation does not require exhaustive matching, therefore it is quite efficient and robust. Matusik et. al. designed an efficient method to compute and shade visual hulls from silhouette images, allowing real-time rendering of a dynamic scene from a large viewing volume [20]. Lok presented a novel technique to accelerate visual hull computation using commodity graphics hardware [19]. However, these approaches cannot handle concave objects, which results in less than satisfactory close-up views of such objects.

Recently, Image-Based Modeling and Rendering (IBMR) methods have become a popular alternative for synthesizing novel views. The basis for IBMR is reconstructing the *plenoptic function* that describes the flow of light in all positions and in all directions [21]. With a complete plenoptic function, novel views can be easily synthesized by plugging the location and directions for the novel views into the plenoptic function. A class of IBMR methods, called *Light Field Rendering*(LFR), uses many images to pre-record the plenoptic function [17, 9, 30]. LFR methods often achieve a stunning level of realism without using any geometric information. However, applying it directly to 3D teleconferencing would require hundreds of cameras, which makes real-time acquisition and transmission impossible. Schirmacher et. al. extended LFR with per-pixel depth information computed with a classic stereo algorithm [29]. Their approach allows real-time online view synthesis, but the fidelity is limited by the quality and the speed of the stereo algorithm (1-2 frames/second).

Our research is aimed at finding a practical approach to 3D teleconferencing that overcome the limitations of existing applications. In particular, we would like to use no more than a dozen cameras to produce real-time photo-realistic results that replicate the experience of face-to-face conversation.

3 Line Light Field Rendering

While applying unconstrained Light Field Rendering (LFR) is not practical due to the sheer amount of data required, we take advantage of the application specific requirements of video teleconferencing. We observed that during a video teleconferencing session, the participant's view point is quite limited, usually at the eye level, with small lateral motions. Thus we can use a 1D linear array of cameras to capture a compact light field, which we refer to as the *Line Light Field*. This compact 1D setup makes real-time capture, transmission, and rendering possible. To achieve the best result, it is desirable to place the camera array horizontally at eye level using a half-silver mirror or actively controlled screen [15]. Novel views at eye level can be generated interactively, allowing the participants to view the remote scene from side to side, or from near to far to gain a sense of 3D. The optical axis of the synthesized view is, however, approximately constrained to the plane that passes through the camera array since there are no cameras to capture information from above or below.

3.1 Rendering

The general concept of Line Light Field rendering is to blend together the appropriate pixels from the nearest cameras in order to compose the correct scene for the user's point of view. This process can be accelerated using texture mapping hardware [12, 2, 33]. Our method, which is in essence a modified version of unstructured lumigraph rendering [2], consists of the following steps:

- Set viewpoint and average scene depth (the *focal plane*)
- Tessellate the image plane of the virtual camera into a list of narrow rectangles
- Compute blending weights and texture coordinates
- Apply multiple texture to rectangles.

Set viewpoint and focal plane The location of the viewpoint is determined by the viewer. He or she can translate horizontally and zoom in and out the virtual camera. In our algorithm, the system starts with an initial (default) view location, and during the conference the viewpoint can be adjusted by simply using the arrows keys on the keyboard. This user viewpoint is also known in the literature as the desired view point, and we refer it as point C_d in the rest of this section. In addition, an approximate *focal plane* is setup, which represents the average depth of the participants. This can be usually determined through camera calibration. The focal plane's depth can be adjusted by user as well.

Allocate vertices in image plane The formation of the image to be displayed begins by tessellating the image plane of the virtual camera. Since our input images are co-linear, we only divide the image plane in the horizontal direction to a series of narrow rectangles. The vertices of these rectangles are back-projected onto the focal plane. In other words, the focal plane is dynamically tessellated in a view-dependent fashion to guarantee uniform tessellation on the image plane, which can lead to better blending.

Blending weights and texture coordinates For each vertex on the focal plane a set of blending weights is computed. The weights are associated with the respective texture coordinates, which are necessary parameters to render the desired image. The computation of the blending weights is done in the following way.

Figure 4. Angles between desired view camera C_v and texture cameras C_i , at a vertex V on the focal plane.

For a given vertex V , we need to find its *valid* cameras, i.e., the set of cameras whose images contain the focal plane point V . If the number of valid cameras is less than the maximum number of blending cameras NBC , then NBC must be updated to that number of valid cameras. Next, for all valid cameras, compute the angle θ_i formed by the desired view point C_d , the focal plane vertex V , and the center of projection of camera C_i , as shown in Figure 4. A small angle means that the desired view is near that particular texture camera. So, for the computation of the blending weights w_i , a group κ of cameras with the NBC smallest angles is selected. The camera with the smallest angle has the largest blending weight, and the weights for the smaller angles are enhanced by an exponential function. The computation of the blending weight is given by the following equation.

$$\hat{w}_i = \exp\left(\frac{-\theta_i^2}{\sigma^2}\right) \quad (1)$$

$$w_i = \frac{\hat{w}_i}{\sum_j \hat{w}_j} \quad (2)$$

for all i and j that are included in κ . The blending factor σ is a user-controllable parameter to control the rate of transition. The smaller value σ is, the less inter-texture blending there is. We typically use a constant value of 2.5° . The blending weights for each vertex are normalized to guarantee a constant brightness of the entire image. For the invalid cameras, a blending weight of zero is assigned. At the end of this process, we have a list of weights and texture coordinates for all cameras.

Multiple texture blend Multiple texture is applied on top of all rectangles formed on the image plane using the respective blending weights and the texture coordinates. The texture images come from all cameras at as high a frame rate as possible. In this procedure, each rectangle is rendered multiple times, and at each time it is rendered using the texture from one camera with the proper

weight. The frame buffer is used as an accumulation buffer. If multi-texture hardware is available, textures from multiple cameras can be rendered at once, reducing the total number of rendering passes. Since all texturing tasks are done by the graphics hardware, the multiple texture blending process is very fast compared to software image processing techniques.

3.2 Sampling Requirement Analysis

A commonly asked question in light field rendering is “how many images are required to generate aliasing-free output.” We provide a sampling analysis for our linear camera array. We first assume that all of the cameras are mounted on a horizontal rail and regularly spaced. The optical axes of the cameras are parallel on a horizontal plane. We then define an error tolerance measure (e) in terms of pixel drift, i.e., the distance from a pixel’s ideal location in the synthesized view. For a given configuration, we would like to find out how much error there will be, or conversely, given an error tolerance measure, how many cameras are needed.

e is obviously a view-dependent factor. If the virtual view corresponds exactly to one of the input view, then e is zero. Here we consider an extreme case when the desired viewpoint is at infinite. In this case, the viewing direction for every vertex on the focal plane is parallel to each other. Given a small enough blending weight σ , our rendering algorithm is in fact cutting the center band of each input image and juxtaposing these bands. The synthesized image is orthogonal in the horizontal direction and perspective in the vertical direction.

Inspired by the sampling analysis for LFR in [3, 18], we attack the error tolerance measure e using a geometric approach. Let us define the following parameters:

- Camera’s field of view FOV
- Camera’s horizontal resolution (in number of pixels) W
- Inter-camera distance d

The problem we try to solve here can be stated as follows: Given a set of camera configuration parameters, and a desired error tolerance e , what is the maximum depth deviation ΔD from the optimal depth D ?

From Figure 5(a), it is easy to see that $\alpha = \tan^{-1}(\frac{d}{2}/D)$, $\beta = \angle OPS = 90 + (90 - \alpha) = 180 - \alpha$. In triangle SPO , we have

$$\frac{\Delta D}{\sin(\Delta\alpha)} = \frac{|OP|}{\sin(\angle PSO)}$$

Substituting $\angle PSO = 180 - \beta - \Delta\alpha = \alpha - \Delta\alpha$ and $|OP| = \sqrt{(d/2)^2 + D^2}$, we get

$$\Delta D = \frac{\sin(\Delta\alpha) \sqrt{(d/2)^2 + D^2}}{\sin(\alpha - \Delta\alpha)}$$

Figure 5. Error Analysis for Creating Orthogonal Views

We can then approximate the angular deviation $\Delta\alpha$ in term of pixel drift e , where $\Delta\alpha = (e/W)FOV$. That leads to:

$$\Delta D = \frac{\sin(e/W * FOV) \sqrt{(d/2)^2 + D^2}}{\sin(\alpha - e/W * FOV)}, \quad (3)$$

where FOV is expressed in radians. Furthermore, since $\sin(\alpha) = (d/2) / \sqrt{(d/2)^2 + D^2}$, $(e/W)FOV$ is usually a very small number and $(e/W)FOV \ll \alpha, d \ll D$, we can approximate Equation 3 as

$$\Delta D = \frac{e}{W} FOV \frac{D^2}{d/2} \quad (4)$$

We can derive a similar equation in case S is closer to the camera instead of farther away.

Let us assume $FOV = 30^\circ, W = 640$, and $D = 1000$ mm. Figure 5(b) shows the maximum depth deviation with respect to pixel drift error under different camera placements $d = 25, 50, 100$ mm. The red line shows the results computed using the rough approximation (Equation 4), while the rest are computed using Equation 3. Note these are “one-sided” numbers, i.e., they only represent how much *further* away the real depth can be. The total distance variation is roughly twice as long. From the results we can see that it is indeed possible and practical to create crisp orthogonal images for depth variation under 400 millimeters, a reasonable value to accommodate normal human motions during a conference.

4 Implementation and Results

4.1 System Architecture

We have implemented a 3D teleconferencing prototype system at three sites: the University of Kentucky (UKy), the University of North Carolina at Chapel Hill (UNC-CH), and Sandia National Labs, California. Each site has a total of eight or ten Sony digital firewire cameras arranged in a linear array, as shown in Figure 2. These cameras are regularly placed at 65 millimeter apart, very close to the minimum distance allowed by the form factor of the camera body. All cameras are synchronized by a wire controlled from a PC and fully calibrated using the method from [34]. Figure 6 shows the configuration that supports the acquisition, transmission, and display for two nodes.

Figure 6. The system architecture of our prototype.

Cameras and Synchronization The video acquisition system (one at each site) includes four to five server PCs interconnected through 100Mbit Ethernet. Each server is connected to two Sony cameras and is used to capture and JPEG encode the raw image data at full VGA resolution. The server may optionally segment the foreground participants from the background and encode this in the alpha channel of the image. The JPEG streams are then sent through the network to be decoded on the rendering system.

Camera synchronization ensures that all cameras capture a video frame at the same time. Without synchronization, adjacent video images may show discontinuities in moving objects, such as a participant's body as they move. The cameras used in our system support an external trigger via a special interface, separate from the Firewire used for data transfer. When the camera receives a TTL pulse on the trigger interface, the camera captures a frame during the next frame interval of the camera's internal clock, which runs at a preset frame rate. The video data is then read out over the next clock interval. Due to this two frame process, the maximum triggered capture speed is slightly less than half of the set camera frame rate. Also, the data may arrive at the host machine anywhere from one to two frame times after the trigger signal.

Figure 7. Triggering and synchronization timeline for a group of cameras.

When multiple cameras are triggered by the same signal, then they are guaranteed to be synchronized to the same video frame and will follow the previously described read out process. However, the camera clocks are not synchronized and may be offset by any amount, as shown in Figure 7. Additionally, the video processing (color conversion, segmentation, and compression) and network transmission can take a different amount of time for each video stream.

The triggering signal is controlled by trigger server software via a parallel port device. Ideally, triggering will occur as fast as possible (half the set frame rate of the camera). However, if the trigger signal is received early by a camera that is still capturing or transmitting the video data, the

Figure 8. Data path for a single video stream, from capture to display.

frame will be delayed an entire frame time or not sent at all. To ensure that all of the camera servers are ready to capture a new frame, the trigger server waits until a confirmation message is received from each of the camera servers. Because the time delay in the camera is relatively longer than the other stages of the pipeline, shown according to threaded data processing in Figure 8, it is safe for the camera server to send the confirmation signal (trigger out) as soon as the data is received in host memory.

Streaming Architecture There are two methods for running the view synthesis program. One is to send all the video streams over the Internet and synthesize novel views at the remote site (remote rendering). Alternatively we can synthesize views locally and only send the final result to the remote site (local rendering). The first approach has a potentially lower latency for a changing viewpoint, but requires extra stream synchronization mechanisms. The second configuration, shown in Figure 6, is easier to manage from a network standpoint, because there is only a single data stream sent over the Internet.

Remote rendering will also typically require more Internet bandwidth than local rendering. In terms of scalability, the bandwidth requirement for remote rendering is $k * R$ where k is the number of cameras and R is the average compressed camera image size. The bandwidth requirement for local rendering, on the other hand, has a resolution fixed by the rendering output, independent of the number of cameras. We expect the number of cameras used in the system to scale at a faster rate than the output resolution.

Given that our current prototype only involves conferencing between two sites, and the viewpoint is not changing rapidly (i.e., no head tracking), we chose to implement the local rendering approach. Notice that the Line Light Field rendering module can be used with or without a geometric proxy.

4.2 Interactive Results

We first show the results from our Line Light Field rendering method in Figure 1. The background is blurry due to the limited number of cameras in use. This can be alleviated using the foreground/background segmentation. To create life-size images, we use a two projector, abutted display at both the UNC-CH and Sandia sites, and a single projector at the UKy site.

Figure 9. View dependent effects when the local conferees move to different spots; the projected images show the synthesized views.

In a teleconferencing session with few participants, we use the view dependent Line Light Field method to synthesized desired views, shown in Figure 9. We use a stationary folder to illustrate the view dependent effect when the local conferees move to different spots. When the local conferees are at right, as in the top image, the view point of the synthesized view is from right, revealing the front cover (on the right) of the folder. When the conferee moves to left, as in the bottom image, the view changes accordingly, revealing the back cover of the folder.

Figure 10. A live teleconferencing session between the University of Kentucky and the University of North Carolina at Chapel Hill.

In Figure 10, we show the setup at UKy in which a *full-duplex* live teleconferencing session with UNC-CH is in progress. The video images were synthesized in real-time using the Line Light Field rendering method.

In terms of performance, we can achieve an update rate of 5-10 frames per second (fps) for VGA input images locally. The bottleneck is in image capture. We can only capture synchronized VGA resolution images at 12-13 fps with one camera per Firewire bus and 7-8 fps with two cameras on the same Firewire bus, as described in section 4.1. When multiple video streams are sent to the rendering machine, network collisions reduce the frame rate to 5-10 fps.

The synthesized view, typically rendered at 1024×512 or 2048×768 , is read back from the rendering program's framebuffer and sent to the remote site through TCP/IP with JPEG encoding. The remote rendering program is capable of decoding and rendering 1024×512 images at over 30 fps. However, the network bandwidth between UKy and UNC-CH is quite limited. The overall frame rate between these two sites varies from 5 fps to 10 fps, depending on the network traffic. We estimated a sustained transfer rate between 3 Mbits and 6 Mbits. Optimization in the network code or the use of a more sophisticated compression scheme is expected to substantially increase the frame rate. Similar performance has been observed between UNC-CH and SNL/CA.

5 Discussion and Conclusion

Conventional video teleconferencing solutions are insufficient for replicating the face-to-face experience of group interactions. Resolution limitations, a lack of depth cues, and smaller than life-size imagery are drawbacks of a conventional single camera and display system. We present techniques and a system for synthesizing novel, high-resolution views rendered in life-size for group tele-immersion.

The Line Light Field method is a practical attempt to bypass the difficult geometry reconstruction problem by using many cameras. Instead of performing dense, computationally intensive 3D scene acquisition, we exploit the fact the participants' motion is rather limited, usually to lateral motions with their eyes remaining at a fixed level, during a video teleconferencing session. This natural restriction allows us to use a limited number of cameras to capture important views. Based on this observation, we have developed a real-time acquisition-through-rendering algorithm based on Light Field Rendering.

The realism of the synthesized view is derived directly from camera images. With smaller, inexpensive cameras becoming available, we believe this method provides a useful solution in the near term. The bottleneck for this method is the bandwidth, both the network bandwidth and the rendering system's internal bus bandwidth. While these can be improved through technical advancement, a more fundamental drawback of this pure image-based method is that the linear arrangement of the cameras limits the possible range of the viewpoints, creating the requirement of placing the camera array at eye level.

Future work will consider these issues in particular:

- New camera arrangements, for example, multiple rows of cameras, above and below the display, with new blending methods to simulate eye level cameras.
- Camera/display integration, such as cameras embedded in the display area.
- Display wall integration, for displaying the rendered images on multi-projector display systems, such as PixelFlex [32].
- Higher speed video capture and transmission, such as automatically synchronized camera systems [24], and improved inter- and intra-stream compression methods.
- Active tradeoff between local and remote rendering to adapt to changing network conditions.
- Combining Light Field-style rendering with improved geometry proxy acquisition for higher fidelity rendering.
- Integrating (multi-)user tracking to set rendering viewpoints and to adapt acquisition algorithms.

We have demonstrated the practicality of the Line Light Field technique using our full duplex 3D video teleconferencing prototype between Sandia and UNC-CH, with regular video teleconferencing between the sites. We have also conducted successful tests between UNC-CH and UKy.

References

- [1] M. Billinghurst, H. H. Kato, S. Weghorst, and T. Furness. A Mixed Reality 3D Conferencing Application. Technical Report R-99-1, Human Interface Technology Laboratory, University of Washington, 1999.
- [2] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured Lumigraph Rendering. In *Proceedings of SIGGRAPH 2001*, pages 43–54, Los Angeles, August 2001.
- [3] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic Sampling. In *Proceedings of SIGGRAPH 2000*, page 307318, New Orleans, August 2000.
- [4] W. Chen, H. Towles, L. Nyland, G. Welch, and H. Fuchs. Toward a Compelling Sensation of Telepresence: Demonstrating a portal to a distant (static) office. In *Proceedings of IEEE Visualization*, 2000.
- [5] A. Colburn, M. Cohen, and S. Drucker. The Role of Eye Gaze in Avatar Mediated Conversational Interfaces. Technical Report MSR-TR-2000-81, Microsoft Research, Redmond, Washington, 2000.
- [6] A. Darabiha, J. Rose, and W. J. MacLean. Video-Rate Stereo Depth Measurement on Programmable Hardware. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 203–210, 2003.
- [7] J. Ohya et al. Real-Time Reproduction of 3D Human Images in Virtual Space Teleconferencing. In *Proceedings of Virtual Reality Annual International Symposium*, pages 408–414, 1993.
- [8] O. Faugeras, B. Hotz, H. Mathieu, T. Viville, Z. Zhang, P. Fua, E. Thron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. Real time correlation-based stereo: Algorithm, implementations and application. Technical Report 2013, INRIA, August 1993.
- [9] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. In *Proceedings of SIGGRAPH 1996*, pages 43–54, New Orleans, August 1996.
- [10] Heiko Hirschmüller. Improvements in Real-Time Correlation-Based Stereo Vision. In *Proceedings of IEEE Workshop on Stereo and Multi-Baseline Vision*, pages 141–148, Kauai, Hawaii, December 2001.
- [11] H. Hirschmüller, P. Innocent, and J. Garibaldi. Real-Time Correlation-Based Stereo Vision with Reduced Border Errors. *International Journal of Computer Vision*, 47(1-3), April-June 2002.
- [12] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically Reparameterized Light Fields. In *Proceedings of SIGGRAPH 2000*, pages 297–306, August 2000.
- [13] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka. A Stereo Engine for Video-rate Dense Depth Mapping and Its New Applications. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 196–202, June 1996.
- [14] K. Konolige. Small Vision Systems: Hardware and Implementation. In *Proceedings of the 8th International Symposium in Robotic Research*, pages 203–212. Springer-Verlag, 1997.

- [15] Andreas M. Kunz and Christian P. Spagno. Technical System for Collaborative Work. In *Proceedings of Workshop on Virtual Environments 2002*, May 2002.
- [16] A. Laurentini. The Visual Hull Concept for Silhouette Based Image Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, February 1994.
- [17] M. Levoy and P. Hanrahan. Light Field Rendering. In *Proceedings of SIGGRAPH 1996*, pages 31–42, New Orleans, August 1996.
- [18] Z.-C. Lin and H.-Y. Shum. On the numbers of samples needed in light field rendering with constant-depth assumption. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.
- [19] B. Lok. Online Model Reconstruction for Interactive Virtual Environments. In *Proceedings 2001 Symposium on Interactive 3D Graphics*, pages 69–72, Chapel Hill, North Carolina, March 2001.
- [20] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-Based Visual Hulls. In *Proceedings of SIGGRAPH 2000*, pages 369–374, New Orleans, August 2000.
- [21] L. McMillan and Gary Bishop. Plenoptic Modeling: An Image-Based Rendering System. In *Proceedings of SIGGRAPH 1995*, pages 39–46, 1995.
- [22] J. Mulligan and K. Daniilidis. View-independent Scene Acquisition for Tele-Presence. Technical Report MS-CIS-00-16, Computer and Information Science Dept., U. of Pennsylvania, 2000.
- [23] J. Mulligan, V. Isler, and K. Daniilidis. Trinocular Stereo: A New Algorithm and its Evaluation. *International Journal of Computer Vision (IJCV)*, Special Issue on Stereo and Multi-baseline Vision, 47:51–61, 2002.
- [24] Point Grey Research Inc. <http://www.ptgrey.com>.
- [25] R. Raskar, M. Brown, R. Yang, W. Chen, G. Welch, H. Towles, B. Seales, and H. Fuchs. Multi-projector displays using camera-based registration. In *Proceeding of IEEE Visualization 1999*, pages 161–168, 1999.
- [26] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays. *Computer Graphics*, 32(Annual Conference Series):179–188, 1998.
- [27] H. Regenbrecht, C. Ott, M. Wagner, T. Lum, P. Kohler, W. Wilke, and E. Mueller. An Augmented Virtuality Approach to 3D Videoconferencing. In *International Symposium on Mixed and Augmented Reality*, 2003.
- [28] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47(1):7–42, May 2002.
- [29] Hartmut Schirmacher, Li Ming, and Hans-Peter Seidel. On-the-Fly Processing of Generalized Lumigraphs. *EUROGRAPHICS 2001*, 20(3), 2001.
- [30] H. Y. Shum and L. W. He. Rendering with Concentric Mosaics. In *Proceedings of SIGGRAPH 1997*, pages 299–306, 1997.

- [31] John Woodfill and Brian Von Herzen. Real-Time Stereo Vision on the PARTS Reconfigurable Computer. In Kenneth L. Pocek and Jeffrey Arnold, editors, *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 201–210, Los Alamitos, CA, 1997. IEEE Computer Society Press.
- [32] K. Yamaashi, J. Cooperstock, T. Narine, . and W. Buxton. Beating the limitations of camera-monitor mediated telepresence with extra eyes. In *SIGCHI 96 Conference Proceedings on Human Factors in Computer Systems*, 1996.
- [33] Jason Yang, Matthew Everett, Chris Buehler, and Leonard McMillan. A Real-Time Distributed Light Field Camera. In *Proceedings of Eurographics Workshop on Rendering*, pages 77–86, 2002.
- [34] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

This page intentionally left blank.

A Implementation Details

The Group Tele-Immersion project consists of hardware and software built to support two groups of collaborators separated by geographical distance. The primary hardware components are an array of cameras and the associated capture PCs, the projector-based display system, and the networking connections between sites. The two primary software components are the camera server capture program and the rendering program. In general, and unless otherwise specified, the system is coded in C++ using Microsoft Visual Studio 6.0 and Visual Studio .NET 2003 on the Win32 platform. This combination was chosen because it best supported Firewire video capture and graphics rendering. As of summer 2004, all projects are built with Microsoft Visual Studio .NET 2003 (7.1), and require the .NET 2003 runtime and redistributable DLLs (MFC71.dll, msver71.dll, msxcp71.dll) to run.

A.1 Calibration

The calibration software takes a set of synchronized camera images of a checkerboard target in various positions and orientations. The software is a MATLAB script based on Jean-Yves Bouguet's camera calibration toolbox. The image input and calibration output paths, camera orientations, etc. are specified in a parameters file.

The calibration software produces a set of MATLAB data files (with .mat extension), one for each camera. For each camera data file, a second script, yangformat.m, is used to extract the calibration data from the MATLAB format and write two text files, a calibration file for the camera position, and a radial distortion file for radial lens effects. The calibration file includes a 3x4 floating point variable projection matrix and two floating point values for the camera center. The radial distortion file contains the set of floating point values for the radial distortion parameters, and can be optionally used to improve image quality with image distortion correction.

A.2 Camera Server (DxCamServer.exe)

The camera sever connects to a Firewire video camera via the Microsoft DirectShow interface. It sets the camera parameters, starts the video capture, processes the images, and then compresses with a JPEG codec for transmission over the network with TCP.

Upon execution, the camera server initializes the video camera, setting camera parameters such as frame rate, resolution, contrast, shutter speed, etc. based upon values in a specified configuration file. The server then starts a thread to handle TCP network connections. The program builds a DirectShow filter graph that includes the source and a user definable function that is called whenever a new video frame arrives. Finally, the filter graph is started and video begins to arrive from the camera.

When a new frame is available, the callback function is started, and image processing begins. The video can be transmitted uncompressed (RAW), in YUV or RGB mode, or compressed with JPEG. In order to use compression or background subtraction, and the input video type is YUV format (2 bytes per pixel), the video frame is converted to RGB format. If background subtraction is enabled, the video image is compared to an RGB reference image of the scene without users, and a filtered difference image is created. This mask image is compressed using run length encoding (RLE). This

is combined with the video image into a single byte buffer which is made available to the TCP-based network thread.

If region-of-interest coding is enabled, the same image processing steps are taken, but only on a subregion of the image, reducing computational load. A smaller RAW or JPEG (and optionally a mask) image results. The ROI offsets from the parent image are included in the network header for future recombination. Previously, the alpha mask generated by the background subtraction process was compressed, using run length encoding, separately from the RGB values, and sent after the JPEG encoded image. Now, the background mask is combined with the RGB image to make a single RGBA 4 channel image that is compressed using JPEG and transmitted using TCP.

The camera server can connect directly to a trigger server. It can send a trigger out command early, as soon as a new frame is received from the camera driver, or it can wait until video processing is complete and the compressed video image is transmitted to the renderer.

A.3 Camera Receiver (CamRecv.exe)

The CamRecv.exe program is a simple OpenGL-based client for the camera servers. It connects to a single camera server, specified on the command line, and receives images at whatever rate the camera is running. Images are sent over a TCP/IP connection, compressed in JPEG format.

A.4 Remote Receiver (FBRecv.exe)

The remote receiver is based on the CamRecv.exe program; it receives streams of JPEG compressed images from the renderer, and display them by rendering to a resizable quad.

A.5 Renderer (ULR.exe)

The rendering program reads a set of calibration files created during the camera calibration process, connects to a specified set of camera servers, and renders using the Line Light Field algorithm. Each connection to a camera server runs in its own thread, and the rendering thread runs separately. Between rendering passes, the renderer enters an idle function that polls each camera server connection. If a new video image is available, it is copied from system memory to video texture memory.

The primary new feature is the ability of the renderer to read back the display image and forward to a remote rendering client. A separate server thread is initialized when the renderer start on a given port. When a remote renderer connects to the server, an OpenGL read back call is made to get the frame buffer contents into main memory. This RGB image is then compressed using the IJL and sent to the client using the same frame definition as video connections between the camera server and renderer.

The renderer is modified to accept a 4 channel RGBA image via the network and decompress and write to texture memory. Texture blending is now accomplished via an OpenGL fragment shader program, which requires OpenGL 1.3 compliant video card and driver, and support for the `GL_ARB_fragment_program` extension.

A.6 Trigger Server (TrigServ.exe)

Camera synchronization is controlled by an external TTL pulse into the cameras. A multi-camera synchronization device was built using a parallel port connector with multiple camera sync connections. Each time a parallel port signal is issued from the host computer, all connected cameras receive the trigger simultaneously. The trigger server software issues these signals according to three different usage modes, automatic, manual, and server. In automatic mode, a trigger rate is specified as a command line parameter. Manual mode sends a trigger with each keystroke. Server mode sends a trigger when a message is received over a network connection.

The trigger server uses the ParaPort 2000 library for parallel port control. This library allows for user mode applications to directly access the parallel port under Windows NT and 2000. With this library, only specific device drivers could access the parallel port hardware.

Completely reliable triggering performance is available in single client trigger server mode mode, with a connection from the renderer. Highest overall triggering performance is achieved with separate connections from each camera server, resulting in a shorter feedback loop and a higher frame rates.

A.7 Libraries

The Intel JPEG Library (IJL), version 1.5, uses hardware specific optimizations to allow for fast encoding and decoding of images to and from the JPEG format. In particular, the discrete cosine transform (DCT) and inverse DCT, used for JPEG processing, are optimized. The OpenCV (Computer Vision) Library, version b1.5, provides functions to support real-time image processing for computer vision applications. Updated versions of the Intel JPEG library, version 1.51 and OpenCV library, version b2.1 are now used.

The Image Debugger (imdebug) library is required for debugging and monitoring images, and functions as a "printf" for images. The OpenGL Extension Wrangler Library (GLEW) is required to handle OpenGL extensions for multitexturing and shader programs. Direct 9.0 is required.

This page intentionally left blank.

B Hardware and Software

Note that this is the UNC configuration; however, Sandia and UKy configurations are similar, and any similar configuration will support the Group Tele-Immersion project. Each camera server system connects to one or two cameras, using one or two Firewire interface boards. Each board supports one camera running at 30fps or three cameras at 15fps; two boards are required to support two cameras running at 30fps each.

B.1 Camera Servers

Hardware

- Dell PowerEdge Workstation
- Dual Intel Pentium III 1.13GHz
- 1.0GB RAM
- 2x OHCI Compliant IEEE 1394 controllers
- 2x Sony DFW-VL500 Cameras
- Intel onboard 100Mbps networking

Software

- Microsoft Windows 2000 Service Pack 4
- Microsoft DirectX 8.0
- Sony 1394 DFW-VL500 Camera driver 5.0.3000.22

Libraries

- Intel JPEG Library 1.51 (1.51.12.44)
- Intel Image Processing Library 2.5 (2.5.2.82)
- OpenCV beta 1.5

B.2 Renderer

Hardware

- Dell Precision 530 Workstation
- Dual Intel Xeon 2.4GHz with Hyperthreading enabled
- 1.0GB RAM
- ATI Radeon 9700 Pro
- 3Com 3C920 Integrated Fast Ethernet Controller

Software

- Microsoft Windows XP Professional Service Pack 1
- Microsoft DirectX 9.0
- MATLAB 6.5 (for calibration)
- Microsoft Visual Studio .NET 2003

Libraries

- Intel JPEG Library 1.51 (1.51.12.44)
- Intel Image Processing Library 2.5 (2.5.2.82)
- OpenCV beta 1.5
- GLEW 1.2.1
- GLUT 3.7
- GLVU 040206
- Image Debugger 1.13b
- ParPort2000 1.0.2.0

DISTRIBUTION:

1 Andrew Nashel
Department of Computer Science
The University of North Carolina at
Chapel Hill
358 Sitterson Hall
Chapel Hill, NC 27599-3175

1 Henry Fuchs
Department of Computer Science
The University of North Carolina at
Chapel Hill
209 Sitterson Hall
Chapel Hill, NC 27599-3175

1 Herman Towles
Department of Computer Science
The University of North Carolina at
Chapel Hill
233 Sitterson Hall
Chapel Hill, NC 27599-3175

1 MS 0321
Robert W. Leland, 09220

1 MS 0822
Constantine Pavlakos, 09326

1 MS 0822
David R. White, 09227

1 MS 9152
Jerrold A. Friesen, 08963

1 MS 1002
Philip D. Heermann, 15230

1 MS 0823
Carl S. Leishman, 09324

1 MS 0822
David S. Logsted, 09326

1 MS 9152
Alan R. Pomplun, 08963

1 MS 0822
Brian N. Wylie, 09227

1 MS 0822
Carl F. Diegert, 09227

1 MS 0822
Andrew T. Wilson, 09227

1 MS 9916
James C. Berry, 08947

1 MS 9916
Diane Gomes, 08947

1 MS 9916
Corbin J. Stewart, 08947

5 MS 9159
Christine L. Yang, 08964

5 MS 9159
Michael F. Hardwick, 08964

1 MS 9159
Zach Heath, 08964

1 MS 9151
James L. Handrock, 08960

1 MS 9151
Charles T. Oien, 08940

1 MS 9151
Kenneth E. Washington, 08900

3 MS 9018
Central Technical Files, 8940-1

1 MS 0899
Technical Library, 9616

1 MS 9021
Classification Office, 8511, for
Technical Library, MS 0899, 9616
DOE/OSTI via URL

This page intentionally left blank.