Combining Approximate Geometry with View-Dependent Texture Mapping – A Hybrid Approach to 3D Video Teleconferencing

CELSO S. KURASHIMA[1,2], RUIGANG YANG[1], ANSELMO LASTRA[1]

[1]Department of Computer Science, University of North Carolina at Chapel Hill
Campus Box 3175, Sitterson Hall, UNC-Chapel Hill, Chapel Hill, NC 27599-3175, USA
{kurashim,ryang,lastra}@cs.unc.edu

[2]Laboratório de Sistemas Integráveis, Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Trav.3, No.158, São Paulo-SP, 05508-900, Brazil

Abstract. We present a hybrid system using computer vision and graphics methods that effectively combines fast automatic 3D model extraction with view-dependent texture mapping for the purpose of real-time person-to-person 3D video teleconferencing. In our approach the problem of creating a dense disparity map is bypassed. We compute only a sparse geometric model, which is textured from multiple cameras using an image-based rendering approach. Using a commodity graphics card and three personal computers arranged in a cluster, we achieve a performance of 3 to 4 frames per second. We demonstrate the ability to provide a 3D tele-immersive environment with our system.

1 Introduction

Current teleconferencing systems are largely implemented with conventional video, which results in a very different experience from that of a face-to-face meeting. We have been researching ways to create more immersive teleconferencing systems using wide field-of-view displays and 3D versus 2D imagery [24] [30]. Unfortunately, real-time extraction of 3D data is a very difficult and computationally demanding problem.

In this paper, we present a hybrid system using both traditional computer vision and image-based rendering approaches to provide 3D tele-immersion. It consists of two steps. In the first step, we use computer vision techniques to create a simple and sparse geometric model or *proxy* of the participant. In the second step, we render the geometric proxy using view-dependent texture mapping (VDTM) [6]. The purpose of the proxy is to provide an approximation of the geometry onto which the textures are applied. The result is a high-resolution virtual environment in which artifacts introduced by the sparse geometry are masked by the textures.

Although the techniques are very general, our prototype system is designed specifically for one-to-one teleconferencing. In this type of system, each participant naturally wishes to have a display positioned directly in front of him or herself, and also to gaze in the direction of the other person's eyes. Since directly in front of each user is also the natural position for a camera, this poses a problem. However, our system can re-project acquired imagery, so we can position the cameras below and to the sides of the screen in order to keep the center of the display clear. In Figure 1, we show a prototype running.

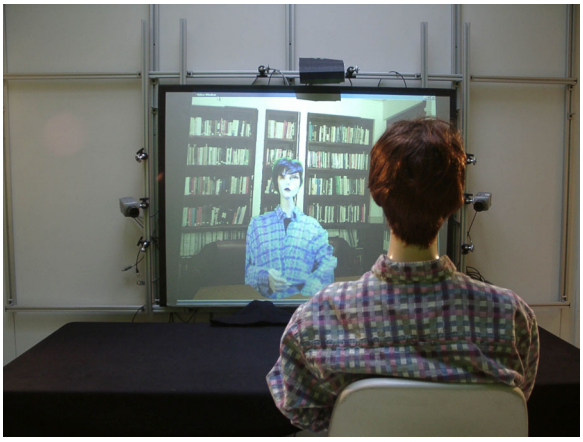


Figure 1: Our Personal 3D Video Teleconferencing Station. The system is displaying a synthesized view of the participant, acting as a 3D mirror (we do not yet have a second system to run in full-duplex mode).

The hybrid nature of our approach simplifies the task. We are not trying to create a dense disparity map at every frame as in [20]. Rather, we only compute a sparse set of depth values at visually important areas, such as object silhouettes and highly textured regions. This greatly reduces the computational complexity to help us meet the real-time requirement. By using computer vision techniques we create a reasonable proxy onto which to map the camera rays used by the VDTM algorithm, enabling us to synthesize new views from a few, sparsely-placed cameras.

The remaining of the paper is organized as follows. The next section includes an overview of related work. Section 3 introduces our proposed system, while the technical details are discussed in Section 4. We present some results in Section 5, and conclude in Section 6.

2 Related Work

The common teleconferencing system of today uses a small- or medium-sized display with a single camera placed on top of the display. Sometimes this camera can be controlled via pan/tilt and zoom motors, but this is simply a setup adjustment used only once or perhaps a few times during a meeting. Typically the camera position remains fixed through the length of the teleconference.

There is an indication that participants can enjoy a much greater sense of presence if they can experience motion parallax. Systems by Gaver [8] and Cooperstock [4] used a motorized camera that was controlled automatically by the remote viewer's motion. They report greater user satisfaction with motion parallax produced by the slaved camera. Their camera setup presents two problems: round-trip latency and the fact that the camera position is offset. The latter does not allow direct eye contact. A recent development to enable camera positioning behind the screen is glass that can be alternatively made translucent and opaque for projection [13]. Our solution is to position the cameras around the screen and dynamically reproject the imagery to the viewpoint of the viewer.

A major goal of computer vision research is to construct 3D models from camera imagery. Faugeras [7] provides a good overview of the variety of approaches that have been employed, most of which have performed the 3D reconstruction off line. Notable is the Virtualized Reality project at CMU [25] that uses 49 cameras mounted on the walls of a room to create, off line, a 3D model of people moving in the room. Some systems are designed to work in real time [21, 29], but this is very difficult because of the large amount of computation necessary to perform a full pixel-by-pixel 3D reconstruction. Recently, the VIRTUE group [10] used a customized DSP board to speed up the computation.

The difficulty of creating 3D models has motivated research into image-based rendering (IBR). We can divide IBR approaches into three categories, those that deal purely with images, ones that use per-pixel range (or disparity), and hybrid approaches that combine geometry with images. The purest image-only approaches, such as the Light Field [16], Lumigraph [9], and Concentric Mosaic [27] methods, resample captured imagery into a database using a 3D or 4D parameterization of light rays (a simplification of the full 5D location and direction). Unfortunately, these approaches require a very large number of cameras in order to keep reconstruction artifacts to a tolerable level [3]. The Lumigraph used computer vision methods to construct, off-line, a rough geometric model in order to obtain more accurate reconstruction. This sparse geometry used for IBR reconstruction is now referred to as *proxy geometry*. We generate proxy geometry dynamically at interactive rates.

McMillan and Bishop [19] introduced a simple warp to map images from known viewpoints to any desired view. The images contain disparity (a measure inversely related to depth) values along with color at each pixel. For our purposes the capture of a disparity value is equivalent to computing depth, so this is not an acceptable approach.

Debevec, et al. [5] presented a method for capturing and rendering of architectural models that uses a small number of static source images. A human operator marks features and a program uses geometric constraints to create a simple model. The regularity of the architectural models makes this feasible. At run time, *view-dependent texture mapping* (VDTM) is used to render color on the model. This is implemented by assigning texture coordinates and blending weights at the vertices of the geometric model (a real time implementation is described in [6]). Debevec's method is a hybrid of human-assisted computer vision and IBR.

There are newer systems that also employ hybrid approaches. Using computer vision methods, Heigl, et al. [11] create a geometric model, off line, from a sequence of camera views produced by sweeping a handheld camera back and forth across a scene. They then use VDTM from the three cameras nearest (in view angle) to each geometric primitive. Buehler, et al. [2] present a detailed analysis of the issues, such as resolution, angle, etc. involved with selection of VDTM imagery at each pixel. They call the resulting assignment of camera-image weights to pixels of the desired image a *blending field*. A geometric proxy is created manually, and graphics hardware is used to render novel views using the blending field for VDTM at interactive rates.

Our system dynamically creates a geometric proxy in real time and assigns texture to the proxy using a VDTM technique based on that of [2].

3 System Overview

The basic components for two-person teleconferencing are the video acquisition system (e.g. cameras), the display system (e.g. monitor or projector), and a communication channel connecting the two locations. Figure 2 shows a sketch of our tele-immersive system. Location 1 and location 2 are the two different places that communicate with each other and have exactly the same components, i.e. a set of digital IEEE 1394 cameras, a set of personal computers (PC cluster), one projector, and a white projection screen. Note that

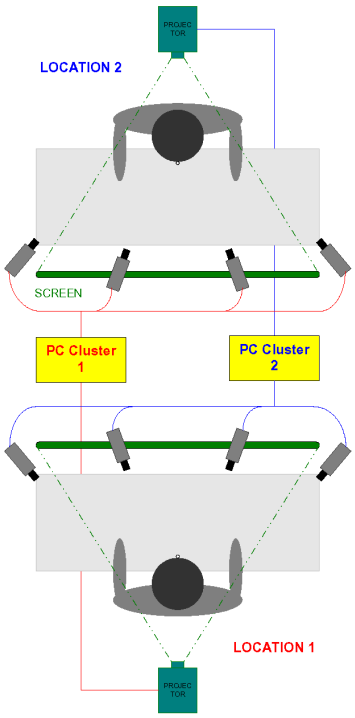


Figure 2: Top view of Tele-Immersion conference System.

our current prototype has only one location and displays a mirror image of what is captured. We plan to soon implement both sides of the system.

The video images captured by the cameras, for example in location 1, are processed by the PC Cluster with image processing and computer vision techniques in order to generate an approximate 3D model of the subject. This 3D model is used to generate a view-dependent texture mapped (VDTM) image, to simulate an image seen from a virtual camera. A great advantage of this method is that the virtual camera can be placed in the middle of the screen. The video generated from the virtual camera is projected onto the screen in the opposite side, location 2 in this example. Our specific approaches and algorithms for these tasks are described in detail in the next section.

The communication channel arrangement, represented by the PC cluster box, is intentionally not detailed here because it is beyond the scope of this work. Rather, we concentrated our attention on the computer graphics and computer vision techniques as well as on the visual presentation aspects.

The cameras shown in Figure 2 are placed below and to the sides of the screen, thus they do not interfere with the projected image (see Figure 5). The projector is placed behind and above the subject so that the subject does not cast a shadow onto the screen. With this arrangement the user of our tele-immersion system is able to see the other conference partner being projected in natural life-size, and from a point of view as if he/she would be sitting right in front of the table. This is one important factor to give the users the so-called sense of presence, the sense of being near the other person in the other location.

4 Method

Our view synthesis method uses a combination of computer vision and computer graphics techniques. It consists of two steps. In the first step we create a *proxy*, a simple and sparse geometric model. In the second step, we render the geometric proxy using view-dependent texture mapping (VDTM) [6]. The purpose of the proxy is just to provide a rough approximation of the geometry onto which to apply the textures.

4.1 Geometry Proxy

We adopt the plane-plus-parallax method [12] to generate our geometric proxy. At each frame, we first segment out the foreground objects, then use the silhouettes of foreground objects to find a best-fitting plane. Salient features on the foreground objects are extracted to provide the offsets from the plane. To facilitate the rendering, the final proxy is a triangular mesh. Figure 3 shows a schematic diagram of the procedure for creating a geometric proxy.

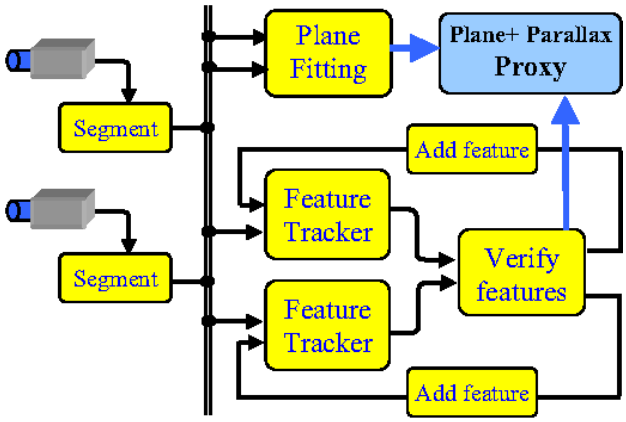


Figure 3: A block diagram for creating a geometry proxy

4.1.1 Plane Approximation

We have developed a robust and fast method to find the best approximating plane of the foreground object from a segmented stereo image pair. The basic idea is to match the silhouettes.

Much research has been conducted to find image silhouettes. Some are based on the *visual hull* concept that

requires a number of cameras from all directions [18, 14, 15]. Some try to match the silhouette segment directly [23], which is a very challenging task because image silhouettes, usually on the occlusion boundaries, may represent different parts of the object in the stereo images, i.e., the silhouette in the right image is not the same as in the left.

For our application, we only want to find an approximating plane from the image silhouettes representing the rough depth of the object. For a pair of stereo images, a pixel ($\vec{p}$) on the silhouette of the first image defines a view ray; that ray defines an epipolar line in the second image. The *epipolar constraint* [7] states that $\vec{p}$'s corresponding point must lie on the epipolar line. So we can intersect the epipolar line with the silhouette to find $\vec{p}$'s matching point. Note that the matching point may not correspond to a real point on the object, but it is a good approximation of where the object is. In our implementation, we first rectify the images so that the epipolar line is parallel to the scan line. Thus the matching process is very simple. For each scan line, we simply match the leftmost foreground pixel in the first image with the leftmost one in the second image, and similarly for the rightmost pixels. Thus we get two depth points for each scan line containing foreground.

This matching scheme will have difficulty dealing with silhouette edges that are nearly parallel to the epipolar line. Instead we use a robust method to fit these points onto a plane. We first fit a plane using the least squares method. Then we compute the mean and standard deviation of the fitting – the distance to the fitted plane. We remove any point with a residual distance greater than three times the standard deviation (applying the standard "3σ" rule), and refit the plane. We repeat this process until the mean distance is less than a threshold or the number of iterations exceeds a user-set limit. From our experiments, we find that the accuracy of the fitted plane is greatly improved after two or three iterations. In practice, we could use the fitted plane from the last frame to bootstrap the robust fitting process, reducing the number of iterations by exploiting the temporal coherence between frames.

4.1.2 Stereo Feature Tracking

We pose the reconstruction of sparse feature points as a spatio-temporal tracking problem to take advantage of the temporal coherence.[1] Our tracking problem can be formally stated as follows:

Given a pair of stereo images $I_{0,t}$ and $I_{1,t}$ at time t, two sets of 2D points $S_0 = \{\vec{p}=[u,v]^T\}$ and $S_1 = \{\vec{q}=[a,b]^T\}$ from that image pair, their corresponding map $M = \{\vec{p_i} \leftrightarrow \vec{q_j}\}$, and a pair of stereo images $I_{0,t+1}$ and $I_{1,t+1}$ at time $t+1$,

Determine A subset $M' \subseteq M$ whose corresponding $\vec{p}$'s and $\vec{q}$'s have matches, denoted by $S_0' = \{\vec{p}'\}$ and $S_1' = \{\vec{q}'\}$, in $I_{0,t+1}$ and $I_{1,t+1}$.

We first conduct independent feature tracking for each camera from time t to $t+1$. We use the KLT[2] tracker [17, 28, 26], which works quite well.

However, the matched points may have drifted or even be wrong. Therefore, we again apply the epipolar constraint to remove any stray points. In practice, due to inaccuracy in camera calibration and feature localization, we cannot expect the epipolar constraint to be satisfied exactly. For a pair of matches ($\vec{p}'$ and $\vec{q}'$), if the distance from $\vec{q}'$ to $\vec{p}'$s epipolar line is greater than a certain threshold, this triplet is considered to be an outlier and is discarded. We use a distance threshold of 1.5 pixels in our experiments.

An important feature of our tracking method is a regeneration scheme that automatically finds additional "good" feature point pairs to track at every frame. A "good" feature point must have rich texture information to facilitate tracking. We first select 2D points in image $I_{0,t+1}$ using the criteria in [26], and again use the KLT tracker to find its corresponding point in $I_{1,t+1}$ (thinking of it as applying the tracking in the spatial domain). If the matched pair returned by the tracker passes the epipolar constraint test, we will add it to the matched pair set for the next frame. This scheme replenishes the feature points lost due to occlusions or non-rigid motions, thus improving the model accuracy and tracker stability.

4.2 Rendering Video Frames with View Dependent Texture Mapping

The image projected onto the large screen is generated by the view-dependent texture mapping method. For this task, we need the proxy information as well as the texture images captured from the cameras. The general concept is to blend together the appropriate pixels from the cameras in order to compose the correct scene for a virtual point of view.

The location of the viewpoint is determined by the local user, at any time, even during the tele-conference session. The scene on the wide screen can be adjusted so that the projected size of the participants is the same as real life-size. Currently the viewpoint is adjusted manually, but we plan to track the viewer's head to generate new viewpoints. The new viewpoint is also known in the literature as the desired view point, and we refer it as point D in the rest of this section.

4.2.1 Multiple Texture

Multiple textures are applied to all triangles of the geometry proxy using the respective blending weight for each texture.

[1] Our tracking method is based on the work by Yang and Zhang [31].

[2] KLT: Kanade-Lucas-Tomasi [1]

Computation of the weights is described below. The texture images come from all cameras at as high a frame rate as possible.

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.

In the texturing process, each vertex of a triangle is assigned a set of blending weights for each texture image. Note that any of these weights may be zero, i.e. that camera will have no effect. Then, texture is applied onto the triangle as many times as the number of cameras, but only the texture images with non-zero weight value will contribute to the final image for the desired viewpoint.

The use of multiple blended textures from the nearest cameras makes the final rendered image be very similar to the real image for the given viewpoint. This is an advantage of using cameras around the subject to be rendered and assigning them appropriate weight values.

Since all texturing tasks are done by the graphics hardware, the multiple texture process is very fast if compared with software image processing techniques. Our multiple texture method is based on the work reported by Devebec, *et. al.* [6] and Buehler, *et. al.* [2].

4.2.2 Computing the Blending Weights

The geometry proxy is formed by a set of points in 3D space, which are triangulated to form a contiguous surface that approximates the subject's surface. For each proxy vertex, a set of blending weights is computed, one for each camera respectively. Since the nearest cameras from the proxy point gives more contribution to its intensity, they must have the larger weights. So, we compute the angle θ_i formed by the desired view point D, the geometry proxy 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, the computation of the blending weights w_i, is given by the following equations.

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

$$w_i = \frac{\hat{w}_i}{\sum_{j=0}^{N-1} \hat{w}_j} \quad (2)$$

where σ is a constant value for the limit maximum angle, $i = 0, 1, ..., N-1$, and N is the number of cameras. As can be noticed, the blending weights at each vertex in the image plane are normalized so that their sum is one. For cameras with angle higher than the limit σ, a blending weight of zero is assigned. At the end of this process, we have a list of weights for all cameras.

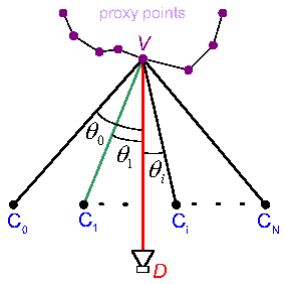


Figure 4: Top view of angles θ_i formed between desired view camera D and texture cameras C_i, at a proxy vertex V.

5 Results

The experiments of our real-time 3D video teleconferencing were performed in the laboratory shown in Figures 1 and 5. Our current prototype (as shown in Figure 5) includes four SONY IEEE 1394 firewire cameras, which were placed around the projection screen and pointed in the user's direction. The system also includes two InFocus projectors, only one of which is used in the current implementation. The other projector was not used. We implemented our methods under the MS-Windows environment running on the PC cluster mentioned in the System Overview section. Two PCs with 1394 cards are used as servers for video acquisition from the four cameras. The video frames are individually encoded to JPEG format in order to save network bandwidth. The sequences of encoded images are sent from the video servers to a Geometry/Renderer PC through a 100Mbit/s local area network. This latter PC, a 2.2GHz Pentium 4, has a graphics video card using the NVidia GeForce3 chip set, and directly drives the projector.

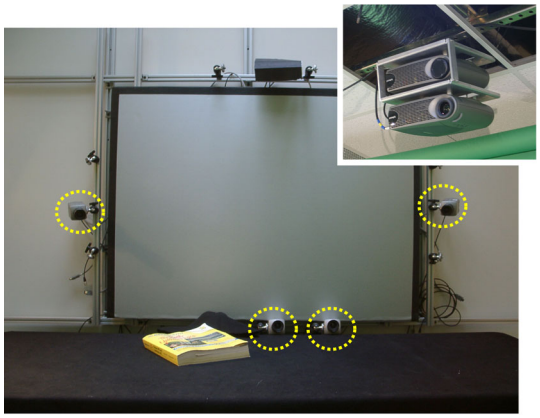


Figure 5: The projectors and cameras setup. The dotted circles in the image highlight the four cameras we use. The inset shows the projectors, designed to display stereo image pairs. We are currently only using one of them.

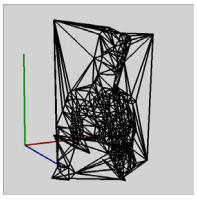
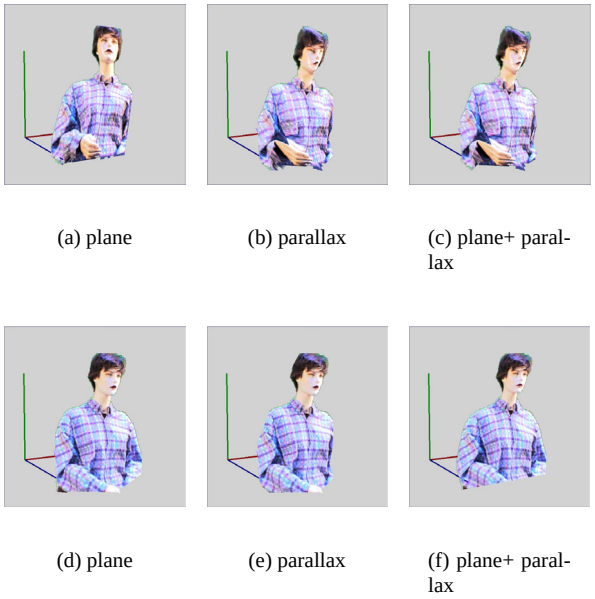


Figure 6: Oblique view of our geometry proxy.



(a) plane (b) parallax (c) plane+ parallax

(d) plane (e) parallax (f) plane+ parallax

Figure 7: The proxy of Figure 6 texture-mapped with two blending cameras (a,b,c) and four blending cameras (d,e,f).

At each video frame, the Geometry/Renderer PC decodes the stream of JPEG images, then performs the geometry proxy extraction as well as the texture mapping processes. In Figure 6, we show an oblique view of the geometry proxy that was generated by our method. This same geometry proxy was textured mapped with images from the cameras resulting in the views shown in Figure 7. Figures 7(a), 7(b), and 7(c) are the results for two blending cameras, and Figures 7(d), 7(e), and 7(f) are the results for four blending cameras.

We can also notice in Figure 7 the effect of the geometry proxy on the final image. In Figures 7(a) and 7(d) the geometry proxy is just a plane that can be easily noticed from the oblique view point. Figures 7(b) and 7(e) show the results with the parallax method only. When comparing the parallax with the plane-only method, we can notice that the shape is nearer to the real subject geometry. The plane-plus-parallax method combines the two approaches and results in the best images in our experiments, as shown in Figures 7(c) and 7(f).

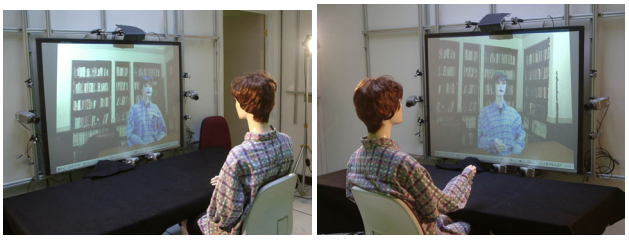


Figure 8: Different views of the participant.

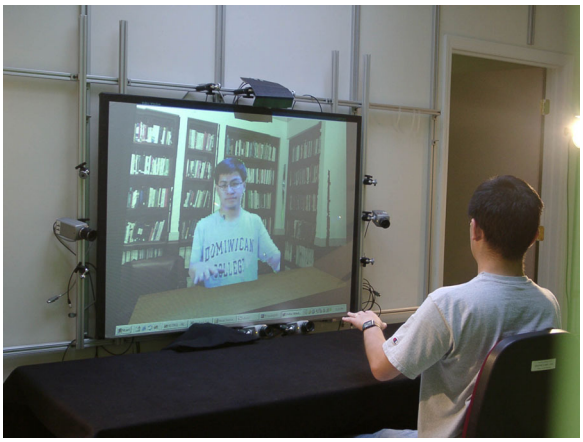


Figure 9: Another user testing out our system.

We also find the number of cameras to be used an important issue in 3D video teleconference. While in Figure 7(c) we have used only two cameras, i.e. the same two cameras used for the proxy extraction, in Figure 7(f) all four cameras were blended for the texture-mapping of the 3D geometry proxy. Recalling that the proxy is the same for both cases, the use of more cameras with appropriate blending weights results in a more natural view of the subject in the scene.

The performance of our prototype system achieves an average rate of 3 to 4 frames-per-second. The segmentation process is the most time consuming taking 180ms to 200ms. The geometry proxy extraction surprisingly takes only 27ms to 57ms, being fast enough to enable our system to work close to real time if we could reduce the cost of segmentation. We are therefore currently porting the segmentation code to run on the video server PCs. We expect this change will substantially improve the overall system performance.

In Figures 1, 8 and 9 we demonstrate the ability to change the point of view of the virtual camera. The user can choose to see the distant location from any point and from any angle. We will couple this with a head tracker. Since this teleconference experiment is in loop-back mode, the participant just sees himself/herself on the screen. How-

ever, in this way it is easy to demonstrate the change of the scene position on the screen while the subject remains in the same position, as in Figures 1 and 8.

A compressed movie file that demonstrates the features of our prototype system can be accessed at [22].

6 Conclusions

We presented a prototype hybrid system for 3D video teleconferencing. It uses our fast, automatic geometry proxy extraction method and view-dependent texture mapping to render the subjects in a 3D tele-immersion environment.

From these experiments we have learned that it is possible to have a real-time 3D video conference by using a combination of fast vision and graphics methods with an off-the-shelf computational system. Since our techniques are implemented in software, the process does not need any specialized hardware, just commodity graphics cards and personal computers, lowering the total cost of the equipment.

Also, despite the trade-off between the number of cameras and the rendered image quality, our prototype system was able to deliver natural images at reasonable frame rate with a small quantity of cameras.

We will continue working on this system in order to build a new version with head-tracking for each user. The frame rates of the rendering portion of the system are fast enough for this purpose, but we would like to increase the update rate. This will require a faster segmentation processing.

Acknowledgements

This work was supported in part by the U.S. Department of Energy ASCI VIEWS program and Sandia National Labs, and the U.S. National Science Foundation grants ACR-9876914 and IIS-0121293.

The first author (CSK) was supported in part by NEC-CPDIA (project No.1113).

The authors would like to thank Herman Towles and the *Office of the Future* group of UNC-CH for the valuable collaboration and for provision of the test environment.

References

[1] S. Birchfield. KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker. http://vision.stanford.edu/∼birch/klt.

[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*, pages 307–318, New Orleans, August 2000.

[4] J.R. Cooperstock, K. Tanikoshi, and W. Buxton. Turning Your Video Monitor into a Virtual Window. In *Proceedings of IEEE PACRIM, Visualization and Signal Processing*, Victoria B.C. Canada, May 1995.

[5] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *SIGGRAPH*, pages 11–20, August 1996.

[6] Paul E. Debevec, George Borshukov, and Yizhou Yu. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. In *9th Eurographics Rendering Workshop*, Vienna, Austria, June 1998.

[7] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.

[8] W. Gaver, G. Smets, and K. Overbeeke. A Virtual Window on Media Space. In *Proceedings of CHI'95*, pages 257–264, Denver CO, USA, May 1995. ACM Press.

[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] VIRTUE group. VIRtual Team User Environment. http://www3.btwebworld.com/virtue/.

[11] B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. Van Gool. Plenoptic Modeling and Rendering from Image Sequences taken by Hand-held Camera. In *Proceedings of DAGM'99*, pages 94–101, 1999.

[12] R. Kumar, P. Anandan, and K. Hanna. Direct recovery of shape from multiple views: A parallax based approach. In *ICPR94*, pages 685–688, 1994.

[13] Andreas M. Kunz and Christian P. Spagno. Technical System for Collaborative Work. In *Proceedings of Eighth Eurographics Workshop on Virtual Environments 2002*, Barcelona, Spain, May 2002.

[14] A. Laurentini. How Far 3D shapes Can be Understood from 2D Silhouettes? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.

[15] A. Laurentini. How Many 2D Silhouettes Does It Take to Reconstruct a 3D Object? *Computer Vision and Image Understanding*, 67(1):81–87, 1997.

[16] M. Levoy and P. Hanrahan. Light Field Rendering. In *Proceedings of SIGGRAPH 1996*, pages 31–42, New Orleans, August 1996.

[17] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[18] 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.

[19] L. McMillan and Gary Bishop. Plenoptic Modeling: An Image-Based Rendering System. In *Proceedings of SIGGRAPH 1995*, pages 39–46, 1995.

[20] 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.

[21] Jane Mulligan and Kostas Daniilidis. Real Time Trinocular Stereo for Tele-immersion. In *Proceedings of the 2001 International Conference on Image Processing (ICIP'01)*, Thessaloniki, Greece, October 2001.

[22] UNC-CH-Department of Computer Science. A Hybrid Approach to 3D Video Teleconferencing. http://www.cs.unc.edu/Research/stc/Media/Movies/hybrid-3Dvtc.html.

[23] S. Pollard and S. Hayes. View synthesis by edge transfer with applications to the generation of immersive video objects. In *Proc. of VRST*, pages 91–98, November 1998.

[24] 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. In *Proceedings of SIGGRAPH 1998*, pages 179–188, Orlando, FL, July 1998.

[25] H. Saito, S. Baba, M. Kimura, S. Vedula, and T. Kanade. Appearance-Based Virtual View Generation of Temporally-Varying Events from Multi-Camera Images in the 3D Room. In *Proceedings of Second International Conference on 3-D Digital Imaging and Modeling*, pages 516–525, October 1999.

[26] J. Shi and C. Tomasi. Good features to track. In *the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 593–600, Washington, June 1994.

[27] H. Y. Shum and L. W. He. Rendering with Concentric Mosaics. In *Proceedings of SIGGRAPH 1999*, pages 299–306, 1999.

[28] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[29] E. Trucco, C. Plakas, N. Brandenburg, P. Kauff, M. Karl, and O. Schreer. Real-Time Disparity Analysis for Immersive 3-D Teleconferencing by Hybrid Recursive Matching and Census Transform. In *Proceedings of Workshop on Video Registration, ICCV 2001*, Vancouver, Canada, July 2001.

[30] R. Yang, M. S. Brown, W. B. Seales, and H. Fuchs. Geometrically Correct Imagery for Teleconferencing. In *Proceedings of ACM Multimedia 99*, pages 179–186, Orlando, November 1999.

[31] R. Yang and Z. Zhang. Model-based Head Pose Tracking With Stereovision. In *Proc. Fifth IEEE International Conference on Automatic Face and Gesture Recognition (FG2002)*, pages 255–260, Washington, DC, May 20-21, 2002.