Viewport: A Fully Distributed Immersive Teleconferencing System with Infrared Dot Pattern

Cha Zhang, Qin Cai, Philip Chou, Zhengyou Zhang, and Ricardo Martin-Brualla [1]

First Draft: Apr. 2012
Minor Revision: Oct. 2012

Technical Report
MSR-TR-2012-60

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

[1]Cha Zhang, Qin Cai, Philip Chou and Zhengyou Zhang are with Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA email: {chazhang,qincai,pachou,zhang}@microsoft.com. Ricardo Martin-Brualla is with CSE, University of Washington, Seattle, WA 98195 USA email: rmartin@cs.washington.edu. A shortened version of this Technical Report has been accepted to IEEE Multimedia Magazine, Special Issue on 3D Imaging Techniques and Multimedia Applications.

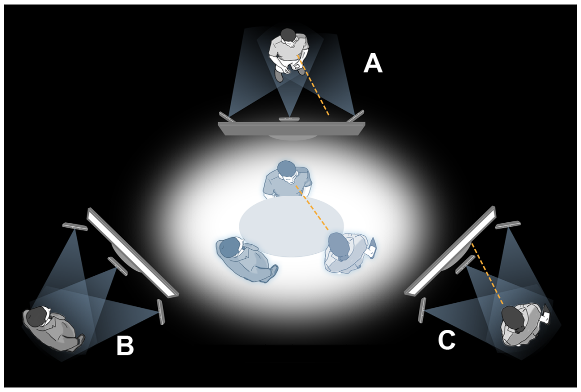


Fig. 1. Fully distributed 3-way videoconferencing that preserves the relative seating geometry as a face-to-face meeting. Such an arrangement will guarantee that the gaze between meeting participants will be faithfully maintained.

***Index Terms*—immersive teleconferencing, 3D, structured light, virtual seating.**

I. Introduction

Immersive teleconferencing has been an attractive research topic for many years, aiming to produce "a videoconferencing experience that creates the illusion that the remote participants are in the same room with you" [28]. An ideal immersive teleconferencing system should be able to replicate the light and sound field (as well as, maybe, taste, smell and touch) observed by the user in a face-to-face meeting, all in real-time. This is certainly a non-trivial task.

A few commercial group video conferencing systems emerged in the past few years, such as the Halo system from Hewlett-Packard and the Telepresence system from Cisco. These systems feature life-size video, spatial audio, and environment excellence by providing furniture, lighting, even wall colors to maximize the conference experience. Although these systems provide substantially improved conference experience than traditional video chatting over the Internet, they are still far from being truly immersive, because the mutual gaze [2] between attendees are not preserved. Research has been done to feed videos of multiple cameras directly into multiview displays at the remote party [16], [19], which was shown to improve the efficiency of group conferencing since the gaze is roughly preserved [19]. Nevertheless, there are still many challenges and inflexibleness in group video conferencing, such as the low visual quality of multiview displays, the limited view position of the user, and the lack of full 3D rendering and motion parallax (free viewpoint video).

In this paper, we study the design of a fully distributed immersive teleconferencing system, which assumes that there is only one user at each station during the conference. Such a system would be of equal importance as a group conferencing system, since many users would prefer to conduct meetings in their own offices for convenience. Compared with group conferencing systems, fully distributed systems permit the rendering of the light and sound fields from a single user's viewpoint at each site, which demands less on the system hardware. However, to create an immersive experience, we still need to support:

- A flexible number of participants for the conference. When the number of cameras at each site is fixed, a full 3D reconstruction is needed to ensure correct mutual gaze when the number of participants is larger than the number of cameras.
- Conference seating geometry identical to face-to-face meetings (Fig. 1). This implies that all equipment at each site shall be calibrated with respect to each other, including cameras that capture the person, display, table height, etc.
- 3D audio/video rendering of remote participants, with motion parallax. For audio, spatial audio rendering is needed as well as multi-channel echo cancelation. For video, to support motion parallax, again high-quality 3D reconstruction is demanded.
- Real-time operation, high network bandwidth and low latency between participants.

Fig. 1 illustrates three people joining a virtual/synthetic meeting from their own offices in three separate locations. A capture device (e.g., one or more Kinect sensors, or a camera rig) at each location captures users in 3D with high fidelity (in both geometry and appearance). They are then placed into a virtual room as if they were seated at the same table. The user's eye positions are tracked by the camera so that the virtual room is rendered appropriately at each location from the user's perspective, reproducing the motion parallax that a user would see in the real world if the three people were meeting face to face. Because a consistent geometry is maintained and the user's eye positions are tracked, the mutual gaze between remote users is maintained. In Fig. 1, users A and C are looking at each other, and B will see that A and C are looking at each other because B sees only their side views. Furthermore, the audio is also spatialized, and the voice of each remote person comes from his location in the virtual room. The display at each location can be 2D or 3D, flat or curved, single or multiple, transparent or opaque, and so forth–the possibilities are numerous. In general, the larger the display is, the more immersive the user's experience would be.

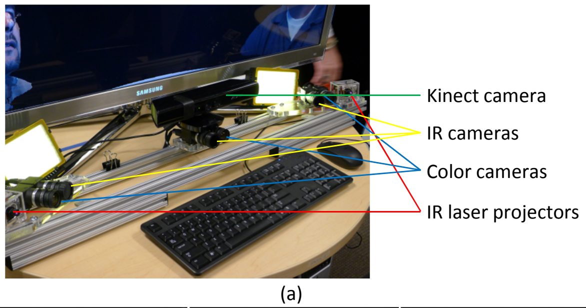
The pursuit of fully distributed immersive teleconferencing has been ongoing for many years. Early representative work includes the Tele-immersion Portal [20], the VIRTUE project [22], the Coliseum [3], etc. These systems shared the same vision that once the 3D models of the users are reconstructed, they can be placed in a common virtual environment to be rendered immersively. However, little attention has been paid to maintaining the conference seating geometry; thus the mutual gaze between attendees has usually been poor in multi-party conferences. In addition, the rendering quality of these systems were often limited due to the small number of cameras used and computational limitations.

Several approaches have been proposed to perform real-time 3D reconstruction for immersive teleconferencing, some based on silhouette [15], [8], some on voxel representation [9], and some on image-matching [17], [23], [30]. Nevertheless, the reconstruction quality of such algorithms are generally unsatisfactory. A straightforward scheme to improve rendering quality is to increase the number of cameras at each site. For instance, the Tele-immersion Portal evolved into a system with over 30 cameras, processed with a computer cluster [18]. Recent work in [11], [26] employed 48 cameras and 12 computers at each station for a multi-resolution mesh based reconstruction, which showed much improved rendering quality. Another recent work is by Feldmann et al. [7], who fused a hybrid-recursive matching scheme with volumetric reconstruction to perform depth reconstruction from 15 HD cameras in real-time, with the help of multiple high-end workstations and graphics cards.

Another venue for real-time 3D reconstruction is through structured light. One popular approach is the phase unwrapping based approach by Zhang and Huang [32], which was adopted in the one-to-many 3D video teleconferencing system in [10]. Phase unwrapping can generally produce very high quality depth maps; however, it is sensitive to depth discontinuity and object motion, and cannot be easily extended to cover a larger view angle. Cotting et al. [6] presented the imperceptible structured light method, where encoded images are visible only to cameras synchronized with the projectors. They focused on reconstructing relatively static scenes through Gray code, and their single-shot method for acquiring dynamic scenes had poor quality. Improvements in rendering quality were made in recent works such as [29] and [1], though few of them can achieve real-time high quality 3D reconstruction.

The third option is to use directly for 3D capturing depth cameras such as the Microsoft Kinect or the SwissRanger depth camera. For instance, Maimone and Fuchs [14] recently presented a teleconferencing system using 6 Kinect sensors at each site. The depth maps are merged and processed to improve rendering quality. However, Kinect sensors cannot be reliably synchronized, and multiple sensors may interfere with each other due to overlapped patterns. As a result, it is difficult to achieve high-quality rendering in their system. Another system by Lee and Ho [12] employed one depth camera and multiple color cameras for depth reconstruction, though they were not considering immersive teleconferencing applications and their depth refinement algorithm takes minutes to finish.

Inspired by the success of the Kinect sensor, we present *Viewport*, a fully distributed immersive teleconferencing system based on a structured light approach that uses an infrared (IR) dot pattern. Each site is equipped with 3 IR cameras, three color cameras and two low cost IR laser projectors (identical to those used in the Kinect sensor). A novel algorithm is developed to reconstruct a sparse point cloud of the user in real-time with a single workstation, which is then transmitted along with the three color video streams to all remote parties. Upon receiving multiple sparse point clouds from the other stations, a virtual seating algorithm places them in a shared virtual environment, and renders them with high quality. The face-to-face meeting geometry is rigorously maintained in the virtual environment by a careful calibration scheme, which computes the relative positions between the camera rig (thus the reconstructed point cloud), the screen, and the table surface. Motion parallax and spatial audio are also enabled to improve the immersive experience.



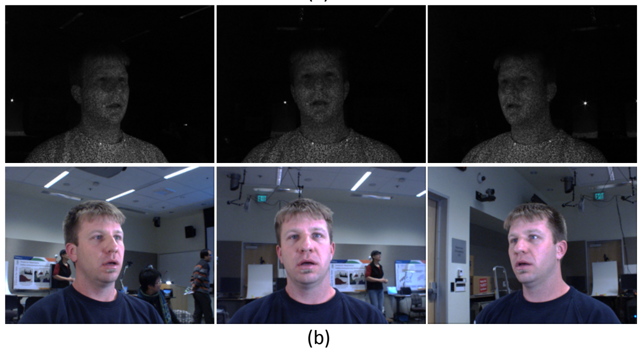


Fig. 2. The Viewport camera rig. (a) The configuration of the camera rig. The Kinect camera in this setup is used for audio capture and head tracking for motion parallax (See more details in Section IV-A). The laser in the Kinect camera is disabled to avoid interference. (b) Example images captured by the camera rig.

The main contributions of this work can be summarized as follows:

- We developed a real working multimedia system for 3-way immersive teleconferencing, which is a very challenging task considering all the technical components involved.
- We presented a solution for sparse 3D reconstruction for real-world scenes, by projecting IR laser dot patterns and reconstruct their 3D positions in real-time. Note surface reconstruction based on quasi-dense points has been done before [13], but mostly initialized with 2D feature points, which is not as robust as laser dot patterns for real-world systems.
- Most importantly, we introduce the scheme of virtual seating, where the point clouds are positioned to maintain the same seating geometry as face-to-face meetings, such that the mutual gaze between participants is faithfully maintained.

The rest of the paper is organized as follows. An overview of the Viewport system is given in Section II. The 3D model reconstruction and rendering are explained in Section III. Virtual seating is described in Section IV. Section V briefly explains the immersive rendering of 3D audio, followed by some experimental results in Section VI. Finally, conclusions and future work are given in Section VII.

II. System Overview

Our Viewport system uses a camera rig for scene capture at each site. The rig consists of three IR cameras, three color cameras and two IR projectors, as shown in Fig. 2(a). The

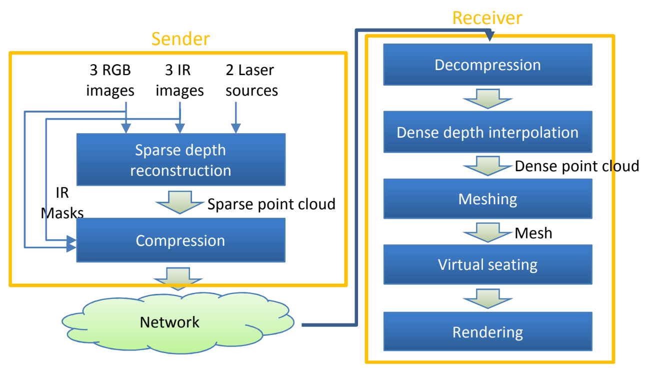


Fig. 3. Overview of the Viewport system.

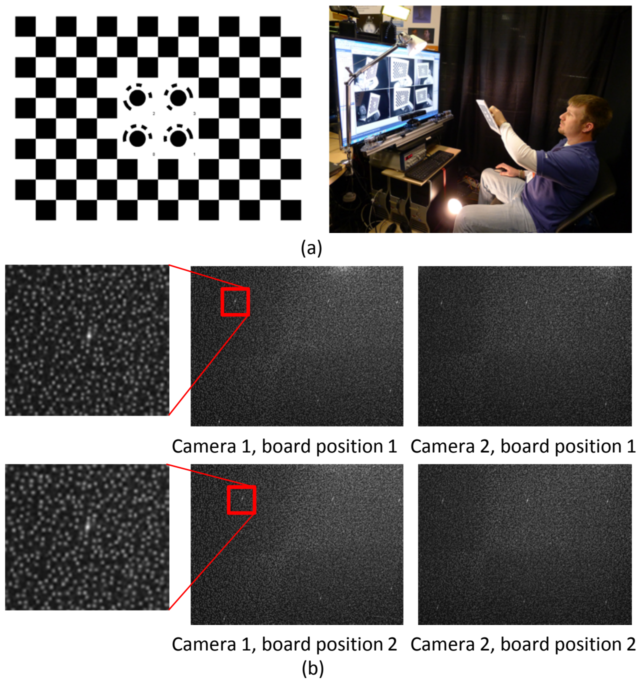

(a)
Camera 1, board position 1
Camera 2, board position 1
Camera 1, board position 2
Camera 2, board position 2
(b)

Fig. 4. Camera rig calibration. (a) The color/IR cameras are calibrated with a specially designed checkerboard. (b) Example images used for calibrating the laser projectors. Note each image contains a few bright dots, which were artifacts of the laser pattern generation mechanism. We take advantage of these bright dots to manually initialize the homography between any two images.

IR projectors are identical to those in the Kinect device, and were placed in an acrylic enclosure for easy mounting. We did not build the system directly with multiple Kinect sensors because they cannot be reliably synchronized and their color image quality is very poor. The IR and color cameras are PointGrey Flea2 cameras, synchronized with an external trigger and operating at 1024×768 pixel resolution. Note for the IR cameras, we removed the IR filter in Flea2 and added a visible light filter (a cheap black acrylic plastic) to increase contrast. Some example images captured by the rig are shown in Fig. 2(b). The system also adopts a Kinect camera for audio capture and head tracking, though the laser in the Kinect camera is disabled to avoid interference.

The overall work flow of our system is shown in Fig. 3. At the sender site, a sparse point cloud of the user is first reconstructed from the camera rig. After compression, the point cloud is sent over the network, together with the three color video streams and three binary mask videos obtained from the IR video. At the receiver site, multiple sparse point clouds and video streams are decompressed. The sparse point clouds are interpolated into dense point clouds with a regression algorithm. The binary masks are also utilized in this step to improve quality. The dense point clouds are then converted into triangular meshes. Once all remote parties' point clouds are processed, the receiver runs a virtual seating algorithm to arrange the meshes in their correct positions, such that the conference geometry of a face-to-face meeting is maintained. Afterwards, all the meshes are rendered to the user for an immersive videoconferencing experience.

We have set up a three-site immersive teleconferencing system for experiment. Each site is equipped with two high-end workstations, one for capturing/sparse point cloud generation (the capture workstation), and the other for dense point cloud generation, meshing and rendering (the rendering workstation). The workstations have dual Intel Xeon Six-Core X5690 CPUs (3.46 GHz), 24 GB of memory, and an NVidia GeForce GTX 580 graphics card. At the capture workstation, the sparse point cloud reconstruction takes about 40 ms, and the remaining cycles are used for frame grabbing, Bayer pattern demosaicing, and compression of three HD color videos (1024×768), three IR masks, and the sparse point cloud. At the rendering workstation, the dense point interpolating and meshing take about 15 ms for *each* remote party. The remaining cycles are used for audio capture/compression, decoding all compressed streams, and audio/video rendering. There is no dedicated audio/video synchronization across the two machines, and they both run in a best effort mode. The overall system runs at about 10–12 frames per second. Note currently all components except mesh rendering are implemented on the CPUs only. Implementing these algorithms on the GPU is relatively straightforward, and we expect much faster frame rate in the near future.

III. 3D Video Processing

In this section, we examine in detail the pipeline of 3D video processing. The calibration of the camera rig is first presented in Section III-A. The sparse point cloud reconstruction scheme is then discussed in Section III-B. Finally, the rendering of the sparse point cloud is described in Section III-C.

A. Camera Rig Calibration

As shown in Fig. 2, the camera rig in our system contains three color/IR camera pairs, spaced about 35 cm apart. Since the subject is only 50-120 cm from the camera rig, the baseline between cameras is relatively large. Consequently, we found it difficult to reconstruct the full geometry based solely on the cameras. To remedy the problem and keep the number of cameras small, we also calibrate the dot patterns of the laser projectors, giving additional information for reconstructing the side of the face and body (Fig. 6).

The first step in calibration is to calibrate all the color/IR cameras. We use a specially designed checkerboard to facilitate the task, as shown in Fig. 4(a). Thanks to the landmarks in the checkerboard, all checkerboard corners can be automatically detected. The user waves the checkerboard in front of the camera rig and capture a set of 30–50 images, and Zhang's calibration algorithm [33] is adopted to find all the intrinsic and extrinsic parameters of the cameras.

The calibration of the laser projectors is less obvious. We use a planar white board (e.g., a 122×91 cm white form board) and the calibrated camera rig to help the process. For each laser projector, we capture two pairs of images of the planar board from two IR cameras, as shown in Fig. 4(b). It can be observed that the projected laser patterns have a few brighter dots. We ask the user to manually specify the 2D position of these dots, which can be used to compute the homography between any two images. The dots' positions can be estimated with subpixel accuracy by fitting a Gaussian to the pixel intensities of a small patch around the dot. Since the two cameras are already calibrated, it is straightforward to reconstruct the position of the planar board (thus all 3D points) in 3D. Since we capture two pairs of images of the board, and we know the homography between all 4 images, we effectively know a large set of laser beams in 3D. The center of projection of the laser projector is thereafter calculated, with an estimated error of less than 0.02cm, thanks to the large number of correspondences found between the captured images. Furthermore, we compute all the light ray directions of the laser projector. During 3D reconstruction, the IR projectors are treated as IR cameras. That is, given the projectors' centers of projection and their light ray directions, a synthesized 2D IR images is generated by spreading a small Gaussian kernel along the intersection of the light rays and the virtual imaging plane of the IR projectors. These synthesized IR images are combined with the IR/color images captured by the real cameras for sparse point cloud reconstruction, as explained next.

B. Sparse 3D Poind Cloud Reconstruction

Given the camera rig, it is natural to first consider using traditional multiview stereo algorithms to reconstruct the 3D geometry of the user. In fact, Yang et al. [30] demonstrated that with the help of commodity GPUs, simple multiview stereo algorithms such as plane sweeping can perform real-time 3D reconstruction well. However, the cameras in our system are in high resolution, and a series of post processing steps is often required to obtain high quality dense depth maps after plane sweeping, which include combining multiple scores from different cameras, hole filling, occlusion handling, etc. We implemented a CPU version of dense multiview stereo with our high-resolution color/IR cameras that includes the above-mentioned post processing steps, and it takes about 40–45 seconds to perform reconstruction for a single video frame on our 12-core machine (code not fully optimized, but parallelized whenever possible), which is clearly unsuitable for immersive telecommunications.

We therefore resort to a new sparse point cloud reconstruction algorithm for our task. The key idea is to match the dot patterns in the IR images only. Since the dot patterns are sparse in the IR images, we can greatly save computation time.

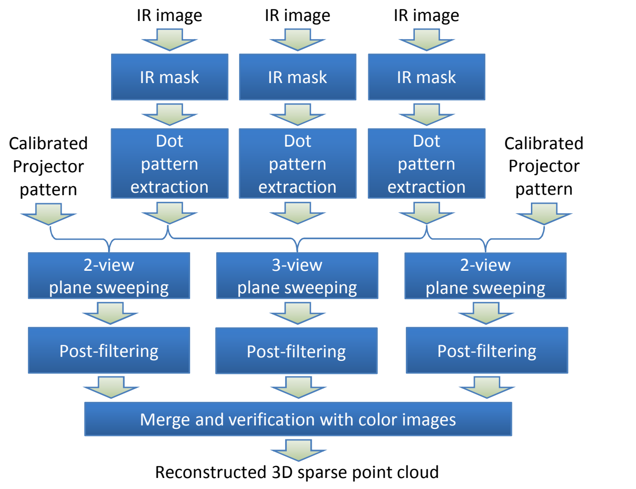


Fig. 5. The work flow of the sparse 3D point cloud reconstruction algorithm.

The work flow of our sparse 3D point cloud reconstruction algorithm is summarized in Fig. 5. For each IR image, we first extract an IR mask that represents the foreground using a simple threshold, as shown in Fig. 6(a). Computing the IR mask has two benefits. First, it improves the speed of the reconstruction, since dot patterns outside the IR masks belong to the background and can be ignored during reconstruction. Second, the masks are used as a silhouette constraint during dense depth map interpolation and rendering at the viewer site, which is very important for achieving high-quality rendering.

The dot patterns in the IR images are then extracted using local maximum detection based on their pixel intensity. No Gaussian fitting is used since that would be too slow. For each dot that is detected, we extract a 15×15 pixel patch surrounding the dot as its descriptor. Since the cameras and IR projectors cover a large view angle for the user, multiple plane sweeping procedures are launched: a 2-view plane sweeping between the left most IR camera and the virtual image of the left laser projector, a 3-view plane sweeping between all three IR images, and another 2-view plane sweeping between the right most IR camera and the virtual image of the right laser projector. The three IR images are used as references in the three plane sweeping routines, respectively. Although the IR projectors may interfere with each other near the frontal facing region of the user, the 3-view plane sweeping is insensitive to such interference, and can help the overall system to perform robustly.

The plane sweeping algorithm implemented in our system is similar to the traditional scheme [5], except for one major difference – it focuses only on the dot patterns. That is, given the reference frame, only the pixels that coincide with the detected dots are used for multiple depth hypothesis testing. Moreover, since a dot in one image must match with another dot in another image, many of the hypothesized depth values can be quickly rejected, leading to significant speed up for reconstruction. When a hypothesized depth value satisfies the dot-dot-matching criteria, a score is computed based on normalized cross-correlation of the descriptors, which is commonly used in the stereo matching literature.

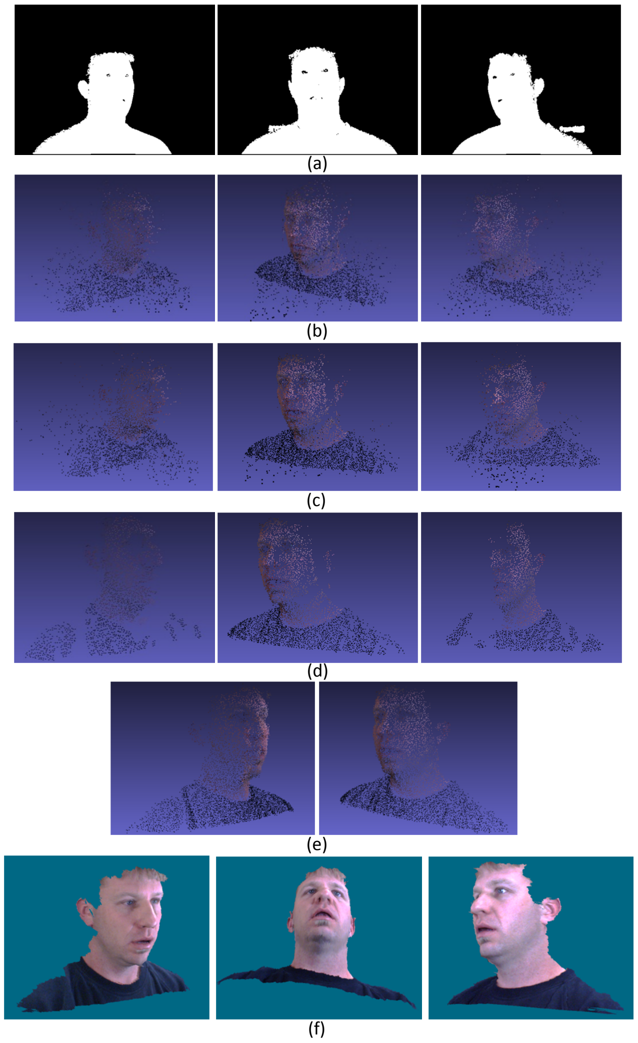


Fig. 6. Results illustrating different steps during sparse point cloud reconstruction. The input images were shown in Fig. 2 (b). (a) The IR masks. (b) Raw sparse point clouds of the three plan sweeping as shown in Fig. 5, without using collaborative filtering in score-volume. (c) Raw sparse point clouds of the three plan sweeping as shown in Fig. 5, using collaborative filtering in score-volume. (d) Post-filtered sparse poind clouds. (e) Two views of the merged sparse point cloud. (f) Rendering results from various viewpoints based on the sparse cloud cloud.

After plane sweeping, for each dot in the reference image, we obtain a score vector, each element corresponding to one hypothesized depth value. The higher the score, the more likely that the scene object resides at that depth. The most straightforward scheme would then be to select the depth value of the highest score as the output depth for each dot (Fig. 6(b)). However, the output point cloud is generally very noisy. More sophisticated schemes such as belief propagation and graph cut could be used to find a globally optimal solution similar to those used in dense multiview stereo [21], but these methods are too slow for real-time applications. In our implementation, we adopted a filtering approach to encourage nearby dots to have similar depth values. Denoting the score vector of a dot $\mathbf{x}$ as $\mathbf{s}(\mathbf{x})$, we filter the score vector as:

$$\mathbf{s}^f(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{N}(\mathbf{x})} w_i \mathbf{s}(\mathbf{x}_i), \qquad (1)$$

where $\mathcal{N}(\mathbf{x})$ is a predefined neighborhood of $\mathbf{x}$, e.g., $\mathbf{x}$'s k nearest neighbor (including $\mathbf{x}$ itself) in 2D coordinate in the reference image. The weights w_i are determined by the distance between $\mathbf{x}$ and $\mathbf{x}_i$, e.g.:

$$w_i = \exp\{-\frac{||\mathbf{x}_i - \mathbf{x}||^2}{2\sigma^2}\}, \qquad (2)$$

where σ is a constant. The depth value corresponding to the highest element in $\mathbf{s}^f(\mathbf{x})$ is chosen as the output, as shown in Fig. 6(c). It can be noted that the point cloud has reduced noise, compared with Fig. 6(b).

We then conduct another round of post-filtering to further remove outliers in the point cloud. Given a dot $\mathbf{x}$ and its k nearest neighbors $\mathcal{N}(\mathbf{x})$, we require that a certain percentage of the dots in the neighborhood have depth values similar to the depth of $\mathbf{x}$. Otherwise, the dot will be excluded from the final output. This post-filtering scheme is very effective, as shown in Fig. 6(d). Finally, the three point clouds are merged into one final result. During the process, the 3D points are also projected onto the three color images to make sure they are consistent in color projection. In this regard, we measure the color consistency as the L2 distance between the projected pixel colors in the UV color space. If the distance were larger than a certain threshold, the point will be rejected. The final point cloud is shown in Fig. 6(e).

For a typical scene, we reconstruct 8,000–12,000 sparse points. Each point is also associated with three binary flags, indicating which of the three IR images were used to obtain the highest score. These flags are useful because they record visibility information of the sparse points, which is helpful for rendering. Together with the binary flags, the 3D points are quantized and transmitted to the remote sites for rendering (Fig. 6(f)). The rendering scheme will be described next.

C. 3D Video Rendering

As mentioned earlier, in our Viewport system, each site is equipped with two workstations, one for capturing, and the other for rendering. The rendering workstation receives multiple sparse point clouds and their related color videos/IR masks. The goal is to render these with high quality, and be consistent in seating geometry with face-to-face meetings. We will elaborate more on seating geometry in Section IV. In this section, we describe the process of generating a 3D mesh for rendering from the input sparse point cloud.

The most straightforward way of performing meshing is to use Delaunary triangulation on the sparse point cloud directly. However, due to noises during the reconstruction step, such a scheme does not produce satisfactory results (Fig. 8 (d)). We propose a rendering algorithm as in Fig. 7. Given the sparse point cloud, we first project it to the three IR cameras, resulting in a sparse 2D point set on each image. Note the binary flags of each point are used here to make sure only the visible IR cameras receive the corresponding points. For each projected image j, denote the sparse projections as $\mathbf{y}_i, i = 1, \cdots, K_j$,

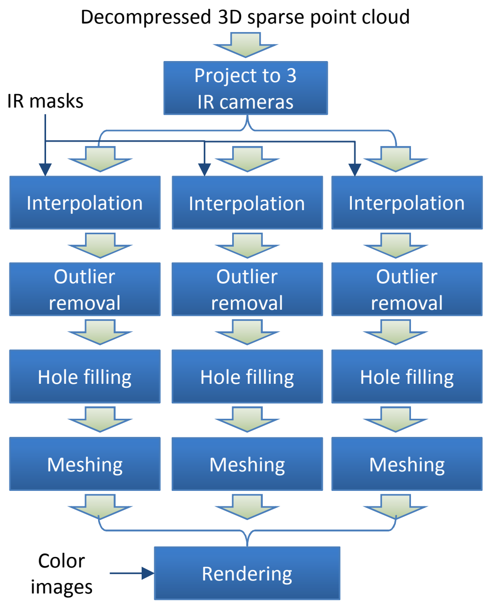


Fig. 7. The work flow of the rendering algorithm.

where K_j is the number of visible IR dots in the j^{th} image, and their depth values as $d(\mathbf{y}_i)$. For each valid pixel $\mathbf{y}$ inside the IR mask, we perform a regression style interpolation as:

$$d(\mathbf{y}) = \sum_{\mathbf{y}_i \in \mathcal{N}(\mathbf{y})} \alpha_i d(\mathbf{y}_i), \tag{3}$$

where $\mathcal{N}(\mathbf{y})$ represents a 2D neighborhood of pixel $\mathbf{y}$. The weights α_i are related to the distance $\mathbf{y}_i - \mathbf{y}$ as:

$$\alpha_i \propto \exp\{-\frac{||\mathbf{y}_i - \mathbf{y}||^2}{2\sigma^2}\}, \tag{4}$$

and $\sum_i \alpha_i = 1$. The resultant dense depth maps are shown in Fig. 8(e). Note because the sparse point cloud is noisy for this particular frame, the interpolated dense depth maps contain blobs of outliers that need to be processed before meshing.

We run a gradient fill algorithm to cluster the above dense depth maps into connected regions. The area of each connected region is compared against a threshold. If the area of the region is too small, the region will be removed from the dense depth map. Afterwards, a simple hole filling algorithm is conducted to remove holes that are not connected with the background region. The processed dense depth map is shown in Fig. 8(f). It can be seen that the quality of the dense depth maps has been significantly improved.

The dense depth maps can be used directly for rendering through surface splatting, as shown in Fig. 9(a). However, it is generally difficult to determine the splat radii, as radii that are too small will cause holes in rendering, and those that are too large will reduce the rendering resolution. In our system, we create a mesh out of each dense depth map, and render them using triangle stripes. The rendering results are shown in Fig. 9(b). Note the improvement in regions where the splatting based rendering has many holes.

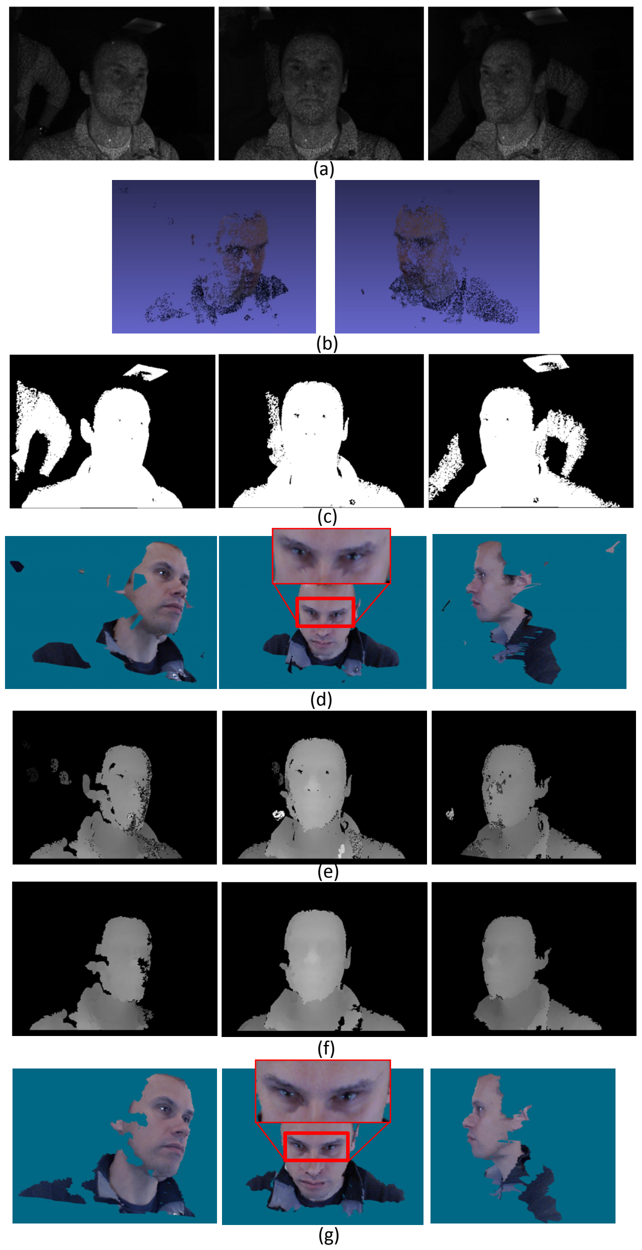


Fig. 8. Results illustrating different steps during rendering. (a) Input IR images. (b) Reconstructed sparse point cloud viewed from two different viewpoints. (c) IR masks. (d) Meshing on the sparse point cloud using Delaunay triangulation. Edges longer than a threshold are discarded to avoid artifacts. Note resultant mesh is noisy and contains outliers. (e) Dense depth map after interpolation. (f) Dense depth map after outlier removal and hole filling. (g) Rendering results after meshing the dense depth maps.

IV. Virtual Seating

In immersive teleconferencing, it is important to preserve the seating geometry of the participants in a face-to-face meeting, as was shown in Fig. 1. While many existing systems have attempted to reconstruct the 3D geometry of the meeting participants and place them in a common virtual environment, few were able to faithfully maintain the seating geometry, leading to poor mutual gaze during the meeting. Indeed, careful calibration of the camera rig, the monitor, and the table surface is required in order to achieve such a goal. In this

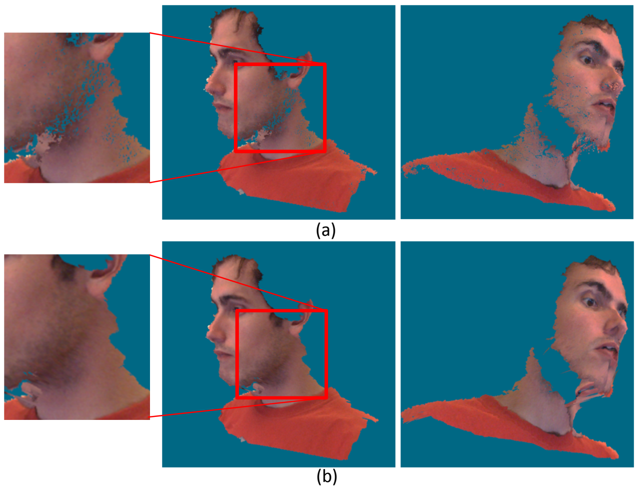


Fig. 9. Comparing splatting and meshing for rendering. (a) Splatting. (b) Meshing.

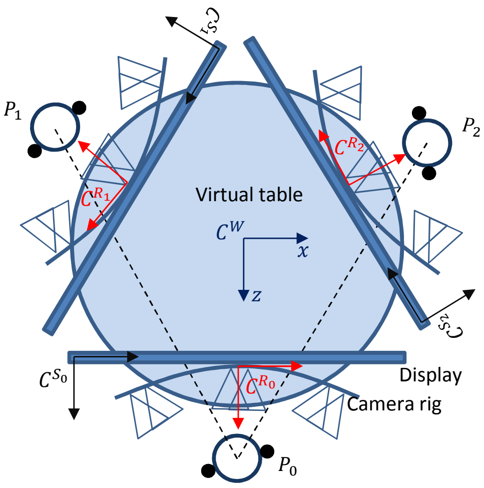


Fig. 10. Illustration of virtual seating.

section, we discuss various steps involved for seating the 3D models correctly in the virtual environment.

A. Virtual Seating

Fig. 10 illustrates the basic idea of virtual seating for a three-party meeting. The reconstructed point clouds are in their respective camera rig coordinate systems, denoted as C^{R_0}, C^{R_1} and C^{R_2}. We would like to place them around a virtual table in the virtual environment. Let the virtual world's coordinate be C^W. If the rotation and the translation between the camera rigs' coordinates and the virtual world's coordinate can be defined, for a particular 3D point $\mathbf{X}_i^k$, where i is the index of the points in the point cloud reconstructed from the kth camera rig, we have:

$$\mathbf{X}_{ki}^W = \mathbf{R}^{R_k W}\mathbf{X}_i^k + \mathbf{t}^{R_k W}, \tag{5}$$

and the points will be placed in the correct location in the virtual world. Note here $\mathbf{R}^{R_k W}$ and $\mathbf{t}^{R_k W}$ denote the rotation and translation from rig k's coordinate system to the virtual world's coordinate system.

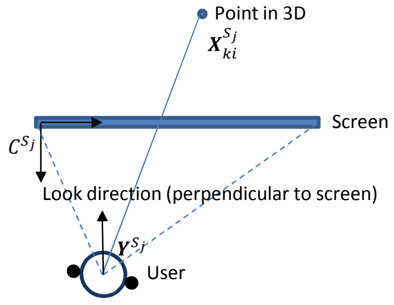


Fig. 11. Rendering in the screen coordinate.

During rendering, given the viewer's eye location in the virtual world's coordinate system, and the screen's display region in the same coordinate system, all point clouds or meshes can be rendered correctly. Note the viewer's eye location is tracked in order to provide a sense of motion parallax, which is an important effect for immersive teleconferencing [31]. Ideally, eye tracking can be conducted using the images captured from the camera rig, which are already in the camera rig's coordinate system. However, recall that due to intensive computation still required with our current CPU implementation, we used two workstations for each site, one for capture and the other for rendering. To reduce the motion parallax latency, we added a kinect camera on top of the rig (Fig. 2) to perform head tracking and audio capture at the rendering workstation. Consequently, the Kinect camera must also be calibrated with respect to the virtual world.

For the jth user, denote C^{S_j} as local coordinate system of the screen, and C^{C_j} as local coordinate system of the Kinect tracking camera. Let the user's eye position with respect to C^{C_j}be $\mathbf{Y}^{C_j}$. We may transform it to world coordinates as:

$$\mathbf{Y}_j^W = \mathbf{R}^{C_j W}\mathbf{Y}^{C_j} + \mathbf{t}^{C_j W}, \tag{6}$$

where $\mathbf{R}^{C_j W}$ and $\mathbf{t}^{C_j W}$ are the rotation and translation from the Kinect's local coordinate system to the virtual world's coordinate system. To perform rendering, it might be more convenient to compute everything in the coordinate system of the screen. One can easily transform the point cloud and the viewer position to the screen coordinate system C^{S_j} by using:

$$\mathbf{X}_{ki}^{S_j} = (\mathbf{R}^{S_j W})^{-1}(\mathbf{X}_{ki}^W - \mathbf{t}^{S_j W}), \tag{7}$$
$$\mathbf{Y}^{S_j} = (\mathbf{R}^{S_j W})^{-1}(\mathbf{Y}_j^W - \mathbf{t}^{S_j W}), \tag{8}$$

where $\mathbf{R}^{S_j W}$ and $\mathbf{t}^{S_j W}$ are the rotation and translation from the screen's local coordinates to the virtual world's coordinates. Once in screen coordinates, the final rendering is simple and illustrated in Fig. 11. Note that the renderer's look direction should be the same as the normal direction of the screen.

B. System Calibration

To perform virtual seating accurately, we need to perform calibration to obtain the related rotation matrices and translation vectors, including:

- The transforms from camera rig to virtual world coordinate $\mathbf{R}^{R_k W}$ and $\mathbf{t}^{R_k W}$;

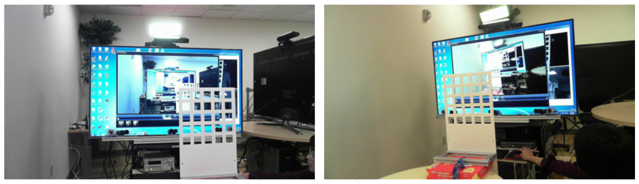


Fig. 12. An auxiliary pattern and an external camera is used to help calibrate the relationship between the camera rig, the screen, and the table top.

- The transforms from tracking camera to virtual world coordinate $\mathbf{R}^{C_jW}$ and $\mathbf{t}^{C_jW}$;
- The transforms from screen to virtual world coordinate $\mathbf{R}^{S_jW}$ and $\mathbf{t}^{S_jW}$.

To this end, we introduce an auxiliary pattern and an external camera to help register the cameras and the screen, as shown in Fig. 12. The pattern is positioned at the edge of the participants' table, roughly 80 cm from the screen. Such an arrangement will help properly map everyone around the table, since it automatically aligns all table tops to the same height when mapping to the virtual world (more details in the next paragraph). The pattern has a number of hollow squares, which can be seen by the camera rig, the tracking camera, and the external camera that is facing the screen. We define a local coordinate system on the pattern, denoted as C^{P_j}. Since the pattern is known, we can compute the transforms from the camera rig to the pattern coordinates as $\mathbf{R}^{R_kP_k}$ and $\mathbf{t}^{R_kP_k}$, and the transforms from the tracking camera to the pattern coordinates as $\mathbf{R}^{C_jP_j}$ and $\mathbf{t}^{C_jP_j}$. With the external camera, we also capture multiple images that include both the pattern and the screen (e.g., Fig. 12), thus obtaining the transforms from the screen to the pattern coordinates as $\mathbf{R}^{S_jP_j}$ and $\mathbf{t}^{S_jP_j}$. If we can find the relationship between the pattern's coordinate and the virtual world coordinate, all transforms mentioned earlier will be easily computed.

Not surprisingly, the transforms between the patterns' coordinates and the virtual world can be defined based on the face-to-face meeting geometry. For instance, for a 3-way distributed teleconference as Fig. 10, the three transforms can be defined as:

$$\mathbf{R}^{P_jW} = \begin{bmatrix} \cos(\theta_j) & 0 & \sin(\theta_j) \\ 0 & 1 & 0 \\ -\sin(\theta_j) & 0 & \cos(\theta_j) \end{bmatrix}, \mathbf{t}^{P_jW} = \begin{bmatrix} r\sin(\theta_j) \\ 0 \\ r\cos(\theta_j) \end{bmatrix}, \tag{9}$$

where $\theta_0 = 0$, $\theta_1 = -\frac{2\pi}{3}$, and $\theta_2 = \frac{2\pi}{3}$. r is the distance from the center of the virtual table to the patterns, which is determined by the size of the virtual table wanted. Note between the patterns and the virtual world there is no translation in y axis. This is because the patterns are all placed at the table top level when calibrating each site, and it guarantees that the meeting attendees are at about the same level in the virtual world.

V. 3D Audio Processing

An immersive teleconference would not be possible without immersive audio. Although this is not the main focus of this paper, we briefly review the techniques used in our system in this Section.

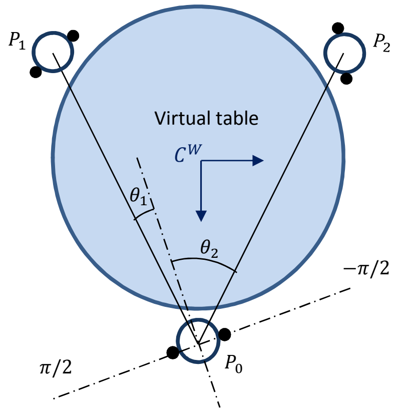


Fig. 13. 3D spatial audio rendering.

During virtual seating, the positions of all the participants have been computed. An example is shown in Fig. 13. Given the local user's head orientation and location in the virtual world, the relative angle θ_i between his/her look direction and the remote participants can be easily computed. As a result, the remote participants' audio can be spatialized to the correct locations for generating faithful immersive audio.

If the user is using headphones, the spatialization task is relatively easy, and can be accomplished by convolving the audio with an appropriate head related transform function (HRTF). As presented in [4], to increase the realism of of spatial audio, we directly measured the combined room impulse response and HRTF (using a dummy head) at a number of fixed locations, and interpolated them to cover additional locations. The combined head and room impulse responses (CHRIRs) are then convolved with the mono (single-channel) audio signal coming from each remote party to perform the spatialization.

If the users are not using headphones with close microphones, then additional steps have to take place. For example, on the capture site, one needs to use acoustic echo cancelation (AEC) potentially followed by dereverberation to get the true speaker signal. On the playback site, if the loudspeakers are close to the user's ears, then potentially one can simply use the spatialized audio as is done for the headphones. If the speakers are reasonably far from the listener, then one may have to compensate for speaker cross-talk and the room impulse response from the loudspeakers to the listener, as has been done in [24]. For the current implementation, we deploy the scheme described in our earlier work in [34], where spatialization is naturally combined with multichannel AEC with constrained Kalman filtering.

VI. Experimental Results

Fig. 14 compares the proposed sparse point reconstruction and interpolation based rendering scheme with our earlier implementation of dense multiview stereo. The dense stereo algorithm also performs three reconstructions using the three IR images as reference, and involves all color/IR cameras and the

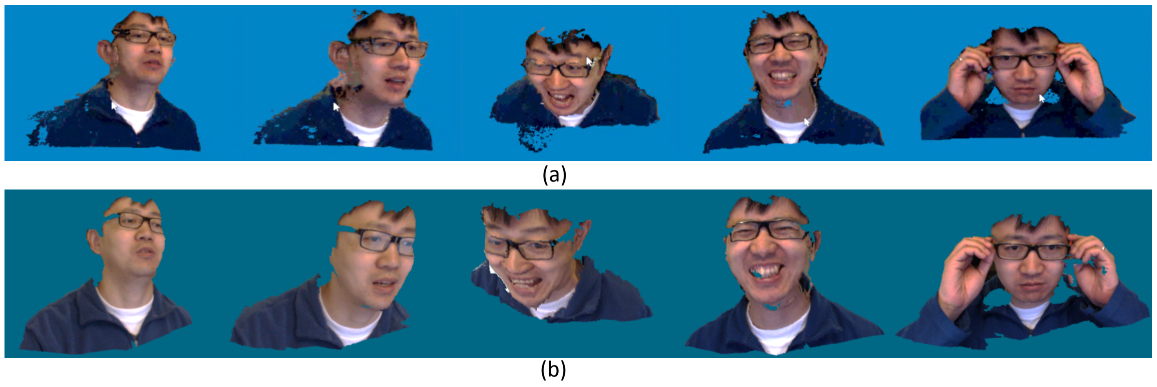


Fig. 14. Comparison between dense multiview stereo and the proposed scheme. (a) Rendering results from dense multiview stereo. (b) Rendering results from the proposed scheme.

calibrated laser patterns. The scores from different modalities are combined with the robust voting scheme presented in [27]. The combined depth and outlier estimation approach proposed by [25] was also implemented to improve the quality of the final dense depth map. The dense multiview stereo algorithm takes about 40–45 seconds to reconstruct one frame, while the sparse point cloud based method takes only 40 ms for reconstruction and an additional 15 ms to render. We achieved almost three orders of magnitude speed up with the proposed method.

In terms of rendering quality, we find the two schemes comparable. The dense multiview stereo scheme does better in some aspects, such as eyeglasses, and has fewer holes. On the other hand, due to the low-resolution nature of the sparse point cloud, the proposed method generates a smoother mesh, and has fewer outliers. The outlier filtering scheme described in Section III-C is aggressive, leading to more holes in the final rendering results. We hope some of these holes will be filled once we include the color images for matching score computation in our future implementation.

Regarding the network bandwidth required for our system, we currently transmit three HD color videos at 1024×768 pixel resolution, three IR binary masks at 512×384 pixels resolution, and 8,000–12,000 sparse points. The color videos can be compressed with a typical video codec, e.g., H.264. The IR masks are currently compressed with run-length coding, which consumes about 10 kb per mask. The 3D coordinates of the sparse points are quantized into 10 bits per dimension (around 1 mm accuracy), plus 3 bits for the visibility flags. Therefore, the overall upload bandwidth of a site is the sum of three HD videos (in total around 75 kB per frame) plus about 50 kB per frame for the masks and sparse point clouds. At 10 fps, the total bandwidth is around 10 Mbps. We are currently exploring better schemes for compressing the sparse point clouds.

The 3-way Viewport system was demonstrated in an internal event with hundreds of visitors to the booth (Fig. 15), and a demo video of the system is available at:

http://research.microsoft.com/apps/video/default.aspx?id=169617.
The general feedback was very positive. People found it has perfect gaze awareness, and were able to tell easily whether a remote party is paying attention to him or her. At one of the stations we used a 3D display to show that the rendering algorithm can also output stereoscopic video. Visitors enjoyed the 3D conferencing experience, and commented that although the rendering still has artifacts, on a 3D display the artifacts were not as disturbing as on a 2D display. We are working on a formal user study to compare gaze awareness in our system to traditional systems, and the results will be reported in a future publication.

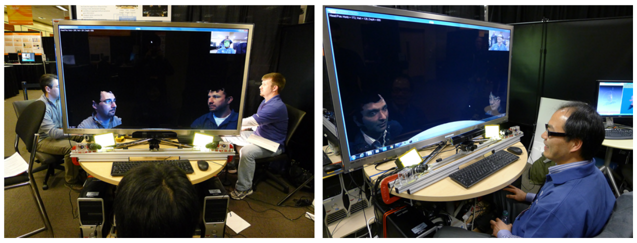


Fig. 15. The 3-way Viewport system was demonstrated in an internal event.

VII. Conclusions and Future Work

We presented *Viewport*, a fully distributed immersive teleconferencing system. With the help of infrared dot patterns, we are able to reconstruct high quality 3D models for each user in real-time, and embed these 3D models into a common virtual environment. Thanks to our virtual seating algorithm, the seating geometry of face-to-face meetings is faithfully maintained, leading to accurate mutual gaze between meeting participants.

There is still much room to improve in Viewport. In particular, the sparse point cloud based reconstruction algorithm does not perform well on hair, since it generally reflects little IR light. In addition, very thin objects such as eyeglass arms tend to be missing in the reconstruction, as the proposed method often smoothes these structures out or removes them as outliers. When the requested virtual viewpoint is very far from the camera's capturing viewpoint, the rendering quality is still not guaranteed. A denser camera array may be able to solve this problem. Another interesting research issue is whether the presented virtual seating algorithm can be applied on small screens, where faithful seating geometry could cause the remote participants to be rendered outside the display. We are exploring schemes to distort the seating geometry, and still maintain the mutual gaze by tracking the users' head orientations.

Acknowledgment

The authors would like to thank many people who have helped building the Viewport system, including Nirupama Chandrasekaran, John Anderson, Guillaume Simonnet, Sean Anderson, Ryan Waller, Mike Starr, Dave Eberly, Yinpeng Chen, Eric Lang, Neil Fishman, Tom Blank, Patrick Therien, Anoop Gupta, and others.

References

[1] K. Akasaka, R. Sagawa, and Y. Yagi. A sensor for simultaneously capturing texture and shape by projecting structured infrared light. In *Sixth International Conference on 3-D Digital Imaging and Modeling*, 2007.

[2] M. Argyle and M. Cook. *Gaze and Mutual Gaze*. Cambridge University Press, 1976.

[3] H. H. Baker, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, W. B. Culbertson, and T. Malzbender. The coliseum immersive teleconferencing system. In *International Workshop on Immersive Telepresence*, 2002.

[4] W. Chen and Z. Zhang. Highly realistic audio spatialization for multiparty conferencing using headphones. In *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2009.

[5] R. T. Collins. A space-sweep approach to true multiimage matching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 1996.

[6] D. Cotting, M. Naef, M. Gross, and H. Fuchs. Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In *Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2004.

[7] I. Feldmann, W. Waizenegger, N. Atzpadin, and O. Schreer. Real-time depth estimation for immersive 3D videoconferencing. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video*, 2010.

[8] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. V. Gool, S. Lang, K. Strehlke, A. V. Moere, and O. Staadt. Blue-c: a spatially immersive display and 3D video portal for telepresence. *ACM Trans. on Graphics*, 22(3):819–827, 2003.

[9] J. Hasenfratz, M. Lapierre, and F. Sillion. A real-time system for full-body interaction with virtual worlds. In *Proceedings of Eurographics Symposium on Virtual Environments*, 2004.

[10] A. Jones, M. Lang, G. Fyffe, X. Yu, J. Busch, I. McDowall, M. Bolas, and P. Debevec. Achieving eye contact in a one-to-many 3D video teleconferencing system. In *SIGGRAPH Emerging Technologies*, 2009.

[11] G. Kurillo, R. Vasudevan, E. Lobaton, and R. Bajcsy. A framework for collaborative real-time 3D teleimmersion in a geographically distributed environment. In *The 10th IEEE International Symposium on Multimedia*, 2008.

[12] E.-K. Lee and Y.-S. Ho. Generation of high-quality depth maps using hybrid camera system for 3-d video. *Journal of Visual Communication and Image Representation*, 22(1), 2011.

[13] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(3), 2005.

[14] A. Maimone and H. Fuchs. Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras. In *International Symposium on Mixed and Augmented Reality*, 2011.

[15] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *ACM SIGGRAPH*, 2000.

[16] W. Matusik and H. Pfister. 3D TV: A scalable system for real-time acquisition, transmission and autostereoscopic display of dynamic scenes. In *ACM SIGGRAPH*, 2004.

[17] J. Mulligan and K. Daniilidis. View-independent scene acquisition for tele-presence. In *International Symposium on Augmented Reality*, 2000.

[18] J. Mulligan, X. Zabulis, N. Kelshikar, and K. Daniilidis. Stereo-based environment scanning for immersive telepresence. *IEEE Trans. on Circuits and Systems for Video Technology*, 14(3):304–320, 2004.

[19] D. Nguyen and J. Canny. Multiview: Spatial faithful group video conferencing. In *ACM CHI*, 2005.

[20] A. Sadagic, H. Towles, L. Holden, K. Daniilidis, and B. Zeleznik. Tele-immersion portal: Towards an ultimate synthesis of computer graphics and computer vision systems. In *Proceedings of 4th Annual International Workshop on Presence*, 2001.

[21] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3), 2002.

[22] O. Schreer, N. Brandenburg, S. Askar, and E. Trucco. A virtual 3D video-conferencing system providing semi-immersive telepresence: A real-time solution in hardware and software. In *Proc. Intl. Conf. eWork and eBusiness*, 2001.

[23] G. Slabaugh, R. Schafer, and M. Hans. Image-based photo hulls. In *Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission (3DPVT)*, 2002.

[24] M.-S. Song, C. Zhang, D. Florencio, and H.-G. Kang. An interactive 3D audio system with loudspeakers. *IEEE Trans. on Multimedia*, 13(5), 2011.

[25] C. Strecha, R. Fransens, and L. V. Gool. Combined depth and outlier estimation in multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[26] R. Vasudevan, Z. Zhou, G. Kurillo, E. Lobaton, R. Bajcsy, and K. Nahrstedt. Real-time stereo-vision system for 3D teleimmersive collaboration. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2010.

[27] G. Vogiatzis, C. H. Esteban, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(12), 2007.

[28] Wainhouse Research. Telepresence 2007: Taking videoconferencing to the next frontier. *Revision 2.0*, 2007.

[29] M. Waschbsch, S. Wrmlin, D. Cotting, and M. Gross. Point-sampled 3D video of real-world scenes. *Signal Processing: Image Communication*, 22(2), 2007.

[30] R. Yang, G. Welch, and G. Bishop. Real-time consensus-based scene reconstruction using commodity graphics hardware. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, 2002.

[31] C. Zhang, D. Florencio, and Z. Zhang. Improving immersive experiences in telecommunication with motion parallax. *IEEE Signal Processing Magazine*, 28(1), 2011.

[32] S. Zhang and P. S. Huang. High-resolution, real-time three-dimensional shape measurement. *Optical Engineering*, 45(12), 2006.

[33] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(11), 2000.

[34] Z. Zhang, Q. Cai, and J. Stokes. Multichannel acoustic echo cancelation in multiparty spatial audio conferencing with constrained kalman filtering. In *International Workshop on Acoustic Echo and Noise Control (IWAENC)*, 2008.