The Office of the Future – Smart Telecommunication Systems

Ubiquitous Computing Seminar FS2014
Student report

Carlo Beltrame
ETH Zürich
Rämistrasse 101
Zürich, Switzerland

ABSTRACT

We present the historic and recent scientific progress in the field of adaptive telecommunication systems. The general inspirational vision in the research field is presented, along with by the major problems that need to be solved in order to create such a communication system. We also give some reasons, why this kind of communication is not yet used widely and show what still needs to be improved on to enable the broad usage by non-technical users.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Human Factors

Keywords: Telecommunication, ubiquitous computing, computer supported cooperative work.

1 INTRODUCTION

In the last years, great progress has been made in the field of smart telecommunication systems. This is to be expected, since due to globalization, a lot of companies have employees in different countries, timezones and even on different continents. A series of research papers focuses on creating computer and sensor systems that let users in different places of the world talk and work together, as if they were situated in the same room. Within a few years, these systems evolved from little more than a vision to fully operating, affordable hardware setups which run sophisticated high-performance software.

We examine the common goal of these research prototypes in section 2. While some examples have been commercially available for some time now, there are still some problems to be solved or improved on. In section 3, we discuss the most important problems faced by the researchers and their common solutions. We then – in section 4 – take a look at some of the most recent papers in the field and compare their setups and approaches. Finally, in section 5 we give an outlook over future work that still needs to be done in this research area.

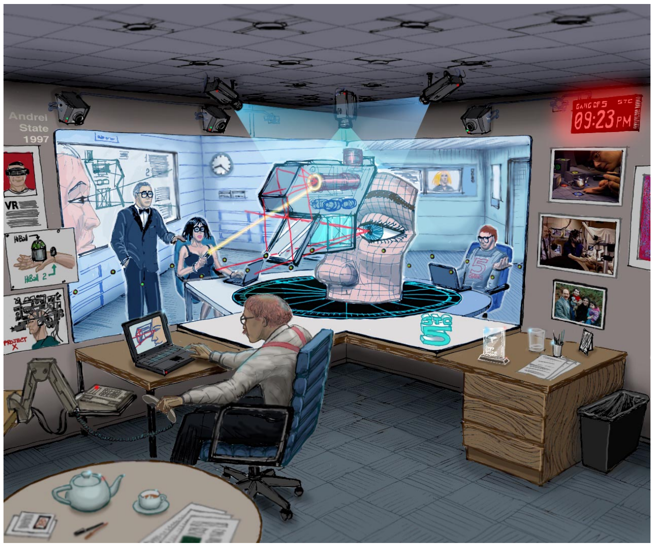

Figure 1: The vision from [13]. The walls act like virtual windows to the remote rooms, through which the users can interact verbally as if they were standing in the same room.

2 VISION

The idea of enabling collaboration across different physical locations using computer technology has been an important research topic for a long time. While differences in local time cannot be overcome in real-time interaction, physical distances can be bridged by transmitting a sufficient amount of information over a computer network. Early attempts in the field mostly concentrated on creating a virtual space which is shared among the different users and with which they can interact collaboratively [3, 5]. Another idea [4] involved a fully dedicated room where the walls are actually projection screens.

One innovative paper [13], however, envisioned a futuristic approach, which can be installed in any office. In this system large display surfaces – which can be any wall, desk or floor

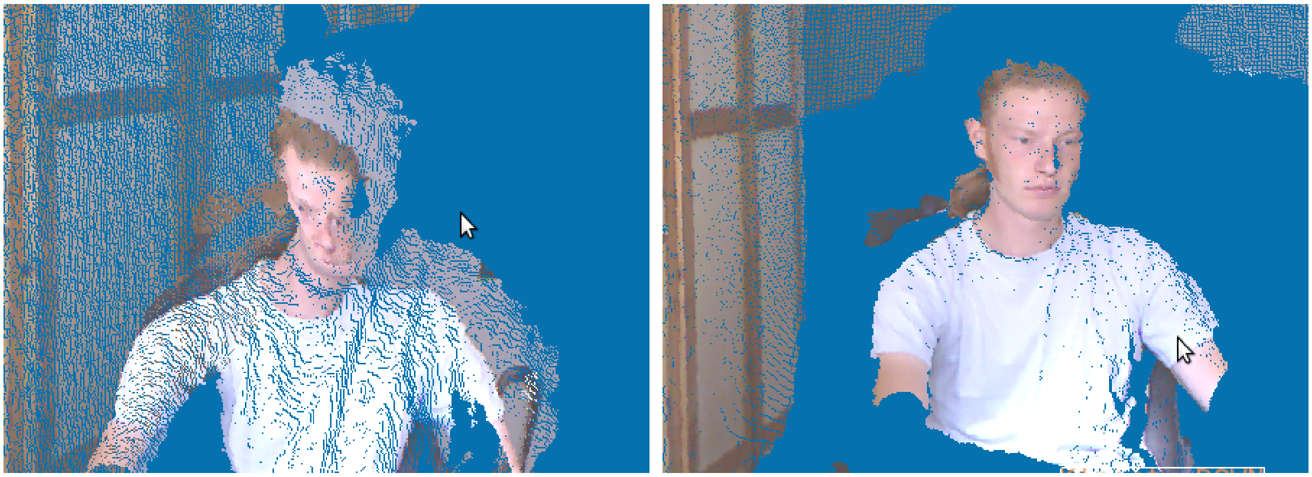

Figure 2: A simple projection of the Kinect RGB data onto the depth data will result in misalignment when the extrinsic parameters are not taken into account. Once the extrinsic parameters have been determined using a calibration, the two data streams can be lined up correctly. Image source: `http://vr.tu-freiberg.de/scivi/?page_id=12`

– serve as virtual windows to the other end of the communication pipeline. See figure 1 for a visualization of the presented vision. The relevant information in the remote room is captured using a combination of cameras and projectors to extract both RGB and depth information and is displayed on surfaces of the local room using another set of projectors. This vision inspired many researchers in the following years to develop telecommunication systems which, with the technological advancements, became more and more practical in their hardware requirements and implementations.

The concrete goals set in the mentioned paper for such a telecommunication system are listed below. First and most of all, the system should create the illusion of a virtual window, transparently showing everything the user could see if there was only a glass pane separating him from the remote room, and showing it from the current point of view of the user. For this illusion to be credible, the rendering framerate needs to be above a certain threshold [8]. When the actual framerate of the system is significantly lower than this threshold, it has been shown [10] that the quality of human interaction can suffer as a consequence.

As another goal, most of the research focuses on developing software components that can run on standard consumer PCs. While none of the papers state a concrete reason for this, the benefits in having a high-performant system for this task will become clear when the the developed technology is used more widely in practice. Being able to run the software on existing office infrastructure implies less work necessary in acquirement and setup of the whole system, ultimately making the product sell better.

One last goal already presented in the original visionary paper [13] is that the system should be able to use any surface in the room as display surface, even if it is not flat. Even more so, the system should be able to adapt to changes in the surfaces. The user looking at the virtual window should (from his point of view), ideally, not notice any change in projection.

From here on out, we will refer to the room that can be seen through the virtual window as the remote room, and to the user in the remote room as the remote user. Similarly, we will call the room where our main user is physically located the local room, and the user herein the local user.

3 IMPLEMENTATION

In this section we will show the most common problems faced by the relevant research papers and their solutions.

3.1 Overview

While there is a wide variety of hardware setups proposed in the literature, there are some common problems that are faced by most of them. First of all, the remote scene needs to be captured by some kind of sensors. As we will see, one sensor is usually not enough to capture the whole scene. Therefore, getting data from multiple sensors, the information streams need to be merged to a common model of the remote scene. Also, either the sensor data streams or the merged model need to be transferred from the remote site to the local site. Finally, from the information recieved from the remote site, the local system needs to render images for the display component (usually projectors, although there are some alternative approaches).

3.2 Scene Capture

The primary scene capture device used in the recent research papers is the Microsoft Kinect [8, 16]. The Kinect contains an RGB camera, a depth sensor and a 4-channel microphone array. Compared to other sensors with similar capacities, the Kinect is relatively cheap, ranging around 150 US$.

The depth sensor is an active sensor consisting of an infrared (IR) projector and an IR camera. In order to extract depth information from the scene, the depth sensor projects a fixed pseudo-random dot pattern onto the scene using the IR projector. Given the nature of infrared light, the dot pattern is

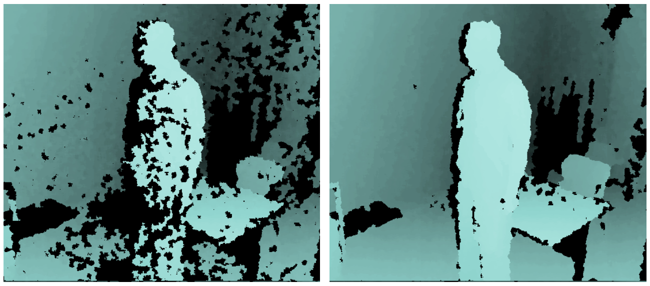

Figure 3: When capturing depth data of the same scene using multiple active depth sensors, the structured light patterns can interfere, resulting in erroneous depth data which needs to be corrected [9].

imperceptible to the human eye but can be picked up by the IR camera of the depth sensor. From the deformation of the known pattern, the scene depth can then be estimated.

In [13], before the Kinect was available, the scene is captured using just RGB cameras. However, depth information is also captured using the same principle of structured light that the Kinect uses. For this purpose, the authors project the structured light pattern onto the scene for only a fraction of a second at a time, followed by the negative image of the pattern. By using a high-framerate projector, they can reduce the amount of time per second the pattern is projected to a minimum, such that the human eye cannot percieve the pattern anymore. Only a camera synchronized to the projector can pick up the pattern. Scene depth is then again estimated from the deformation due to scene structure.

A newer version of the Kinect features a different kind of depth sensor, which has a wider angle of view and uses the time-of-flight principle. Again, an infrared pattern is projected onto the scene and picked up by an IR camera. However, the depth is estimated from the time it takes the infrared rays to get from the projector to the scene and back to the camera.

One important issue occurs when using multiple active depth sensors to capture the same room. Because every sensor projects its own structured light pattern onto the scene, the sensors tend to interfere with each other, resulting in holes in the depth images (see figure 3a). These artefacts are usually corrected by using depth data from multiple sensors [11] or by using multiple successive depth images from the same depth sensor [6].

Since the RGB camera on a Kinect is not in the exact same location as the depth sensor, the images acquired from the two sensors do not line up exactly without further effort, as can be seen in figure 2a. Therefore, both sensor types need to be calibrated, in order to determine their respective intrinsic and extrinsic parameters. Using these parameters, the misalignment can then be corrected.

3.3 Registration

Note that as long as our sensor has a fixed position in the room, due to occlusion by other objects, all objects in the room generally can't be captured by a single Kinect. Particularly when trying to simulate a window, where the local user is free to move in the local room, we need multiple Kinect units to capture everything the local user could be able to see through the communication system. Recall that ideally we want to get one unified room model from our sensor data stream. Any process that merges multiple sensor data stream is called registration.

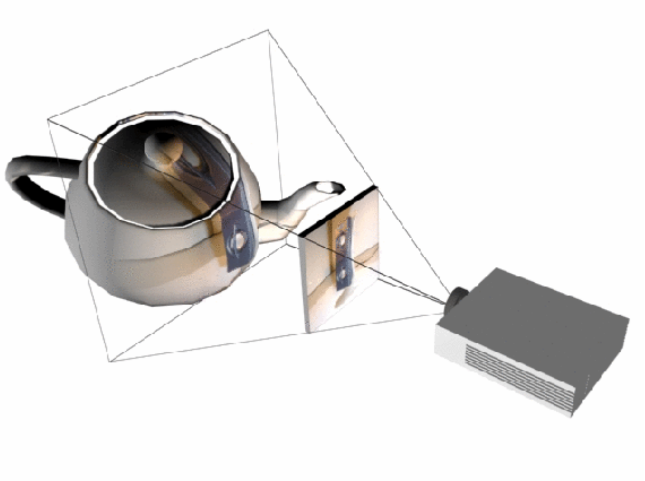

Figure 4: The RGB images from the Kinect units are used as projective textures for the 3D model generated from the depth data. Image source: `http://cgg-journal.com/2004-2/03/`

The usual registration approach taken in the literature is to generate a 3D model of the remote scene from the depth data and then use the RGB images as projective textures (see figure 4). In order to reconstruct a 3D model, a flat triangle mesh with one vertex per depth pixel is overlaid with the depth image. Then the triangle mesh is deformed by setting the distance of each vertex to the depth camera according to the corresponding depth data value. Effectively, vertices corresponding to deeper depth values (darker areas in typical depth images) are "pushed in" further than vertices with closer depth values. Additionally, a thresholding method can be used to separate objects when the depth jump is too steep (which occurs at the edge of foreground objects). This operation on the triangle mesh can be performed as a vertex shader for fast performance [6, 8].

Using the techniques described above, the depth data from every Kinect unit can be used to create a partial 3D model of the room, and the RGB images can be mapped onto the partial models as projective textures. Since with the extrinsic parameters, we have already found the sensor to world transformation during calibration, applying a geometric merger algorithm such as ICP [14] is one possible way to proceed from this point on [6]. However, some authors [8] note that geometric merger algorithms take a lot of computational power and thus defer the merging until after the first pass of the rendering stage (see section 3.5).

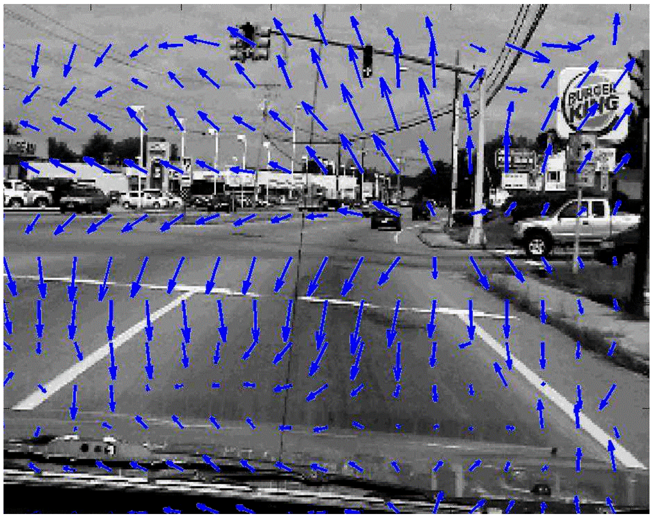

Figure 5: In optical flow, a video compression technique, the inter-frame motion for every pixel or image region is computed and encoded instead of the whole frame. Image source: `http://vislab.bu.edu/projects/vistars/`

3.4 Transmission

Unfortunately, none of the papers considered for this report handle the transmission of the model data from the remote site to the local site. The developed solutions just being prototypes, the authors usually kept the 3D model data in the graphics card buffer of one computer running the system on both ends. However, naïvely transmitting the whole model in every frame over the network is likely to be too slow for this application, since we need to sustain a relatively high framerate to enable natural human interaction. Therefore, in this section, we present a novel proposed approach to optimize the data transfer.

The general idea of our optimization scheme comes from the observation that in an office setting, which the discussed systems are generally developed for, most of the scene stays stationary most of the time. This allows us to apply to the 3D models a transform similar to the optical flow transform as it is used on video data.

Optical flow [7], in short, assumes the total brightness of video frames to stay constant between successive frames. This constraint is then used to calculate the most likely motion of every pixel or patch of pixels between two frames. This way, after encoding the initial frame as a whole, only incremental motion and previously obstructed parts of the image need to be encoded for later images. In practical video compression algorithms [1], frames are divided into different categories following a periodically repeating pattern. The so-called I-frames (short for index frames) are encoded in their entirety, while other categories can be reduced to only incremental changes according to the optical flow. Using this scheme, especially for video streams containing little motion, high compression rates can be acheived.

In our case, as stated above, we can assume that there is not much motion in the scene. As an equivalent of the brightness constancy assumption, we can assume that the total volume of the objects in the room stays the same. Extending existing optical flow algorithms to handle voxels, points in a point cloud or vertex groups (segmented based on connectivity and past motion) should be straightforward. Since the Kinect SDK already has an API for human skeleton detection built in, we could also try to match vertices to skeleton joints or bones and then just encode the incremental changes in skeleton pose.

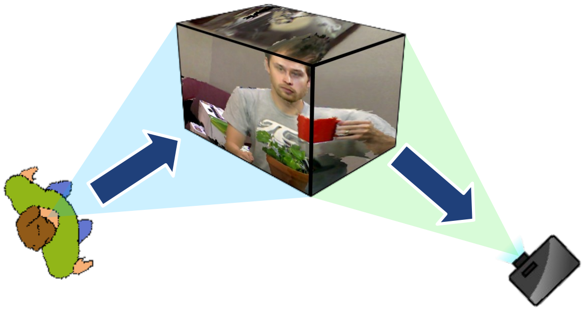

Figure 6: The second pass of the two-pass rendering scheme uses the image the local user should see (from the first rendering pass) as a projective texture for a model of the local scene structure, and renders this virtual local scene from the viewpoint of the projector. Based on an image from [8].

As an alternative to sending the 3D model data over the network, we could also apply optical flow compression to the depth image stream and defer the 3D model construction and registration to be executed on the local site. In any case, we can apply standard optical flow to the texture data, since the data streams from the Kinect RGB cameras are normal video streams.

3.5 Adaptive Rendering

Finally, having a textured 3D model of the remote scene on a PC in the local room, the model needs to be rendered down to a 2D image that can be sent to a display medium (usually a projector [2, 12, 13, 16]). Due to our goals of simulating a virtual window and being able to adapt to the projection surface, this rendering stage usually consists of two passes. In the first pass, the model is rendered from the point of view of the local user to generate the view the user should get when he is looking at the window. In the second pass, the actual image to be projected is determined. To do this, the image from the first pass is used as projective texture for the *local* scene. This textured local scene can then be rendered from the point of view of the projector (see figure 6). The resulting image is ready to be sent to the projector.

In order to perform these two steps, both the position of the local user and the structure of the local scene need to be known. Fortunately, since the whole telecommunication system is assumed to be bi-directional, there will likely be some similar scene capture setup in the local room as there is in the remote room. It is then straightforward to use some of

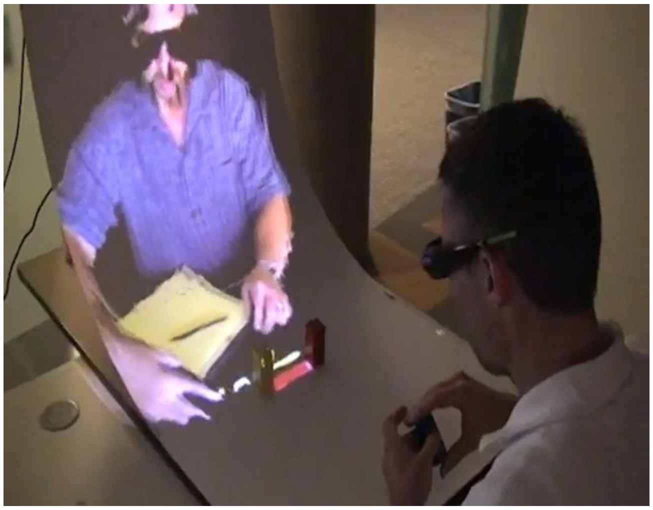

Figure 7: MirageTable [2]. A projector and a Kinect are located on top of the curved surface, looking down on the user. Image from http://www.engadget.com/2012/05/12/microsoft-researchs-miragetable-brings-some-augmented-reality-t/.

the well-developed computer vision techniques for facial and eye recognition to recognize the current position of the local user's eyes. Once the user is localized, it is also possible to restrict the search space on the next frame initially, in order to keep tracking the same user even when more users enter the room.

Obviously, the system can only project the appropriate image for one local user at a time. However, when another user needs to get the attention of the system, the Kinect's microphone array can be made use of. Through the Kinect SDK, we have the ability to get the horizontal angle between the facing direction of the Kinect and the source of the audio signal reaching the microphones. We could therefore implement some speech command to control which user has the focus of the system: Whoever says the codeword will be triangulated by the Kinect units in the room. The eye detection system can then adjust the search space where it starts looking for the eyes in the next frame.

4 HARDWARE SETUPS

In this section we will look at some exemplary hardware configurations from the literature.

MirageTable [2] uses a curved surface in front of the user with a 3D projector and a Kinect unit mounted on top, looking down on both the surface and the user (see figure 7). This setup does not incorporate a very big viewing window, but its components are rather easy to acquire, install and calibrate when compared to some other implementations. The single projector makes it easy to enable 3D imagery, compared to multi-projector setups, even though in that case, the users are forced to wear 3D glasses which, due to missing eye contact, might have an impact on natural interaction.

The Beamatron [16] also consists of one projector and one Kinect, but mounted on a controllable pan and tilt platform built for stage lighting. While this system still only uses a single projector and Kinect, it is a lot harder to calibrate, because the extrinsic parameters in this system depend on the current orientation of the platform. This difficulty is even worsened by the fact that the values for pan and tilt given by the platform do not always match the ground truth accurately. The authors solved this dilemma applying some hardware modifications to the platform. The upside of this implementation on the other hand is clear: The Beamatron is much more flexible in projection placement and choosing the right part of the room to capture.

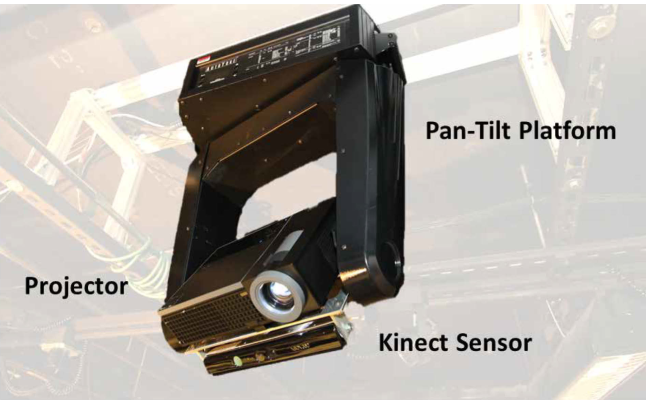

Figure 8: The Beamatron [16]. A projector and a Kinect are mounted on a pan and tilt platform, enabling capture and projection all over the room.

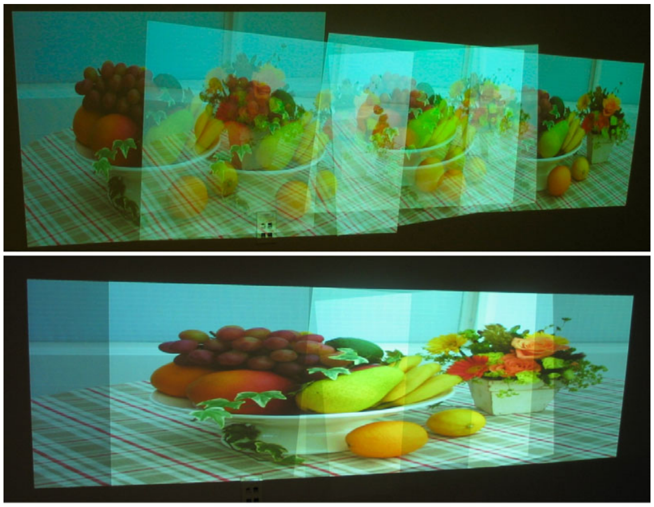

Figure 9: Overlapping projection areas can easily be handled by the described setup. Image from [13].

In both [13] and [12], the authors chose to install multiple fixed projectors on the ceiling in their test room, along with multiple scene capture devices. Obviously, this setup costs more than MirageTable or the Beamatron due to the higher number of projectors. It also brings with it the problem of overlapping projection areas. However, using the local scene model and the position of the projectors, this problem can be solved, and indeed the authors can then choose to set up the projectors for increased brightness, resolution or display area [13].

Finally, [8] uses multiple Kinects but just one autostereoscopic (3D-)display. Like MirageTable, the field of view is relatively limited in this implementation, and autostereoscopic displays are still relatively expensive today. However, the image quality of the display is unmatched when compared to 3D-projectors. The display is also the only implementation that can create 3D imagery without forcing the user to wear any additional devices. To display 3D images, the autostereoscopic display only needs to know the eye position of the user, which as we recall is already calculated in the rendering stage.

5 CONCLUSION

There has been much work and improvement in the field of multimedia telecommunication systems in the last years. Most of the involved problems have thus far been solved, and the solutions are ready to be run on standard consumer PCs. However, the last big open problem relates to the transmission of large model data over the network. For video only, the existing solutions used in video conference systems have proven efficient enough, and using the approach we presented, the last hurdle could possibly also be overcome soon. Once this issue is resolved, it is only a matter of time until the envisioned office of the future will be the office of the present.

References

1. J. G. Apostolopoulos. Video compression: Principles, practice, and standards. `http://ewh.ieee.org/r6/scv/ce/meetings/video_coding_overview_IEEESantaClara_Sept05.pdf`, 2005. Accessed: 2014-03-23.

2. H. Benko, R. Jota, and A. Wilson. Miragetable: Freehand interaction on a projected augmented reality tabletop. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 199–208. ACM, 2012.

3. C. Codella, R. Jalili, L. Koved, J. B. Lewis, D. T. Ling, J. S. Lipscomb, D. A. Rabenhorst, C. P. Wang, A. Norton, P. Sweeney, and G. Turk. Interactive simulation in a multi-person virtual world. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 329–334. ACM, 1992.

4. C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 135–142. ACM, 1993.

5. H. Fuchs and G. Bishop. Research directions in virtual environments. Technical report, University of North Carolina at Chapel Hill, 1992.

6. S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568. ACM, 2011.

7. B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679. Morgan Kaufmann Publishers Inc., 1981.

8. A. Maimone, J. Bidwell, K. Peng, and H. Fuchs. Augmented reality: Enhanced personal autostereoscopic telepresence system using commodity depth cameras. *Comput. Graph.*, 36(7):791–807, November 2012.

9. A. Maimone and H. Fuchs. Reducing interference between multiple structured light depth sensors using motion. In *Proceedings of the 2012 IEEE Virtual Reality*, VR '12, pages 51–54. IEEE Computer Society, 2012.

10. M. J. Meehan. *Physiological Reaction As an Objective Measure of Presence in Virtual Environments*. PhD thesis, The University of North Carolina at Chapel Hill, 2001.

11. P. Merrell, A. Akbarzadeh, L. Wang, J.-M. Frahm, R. Yang, and D. Nistr. Real-time visibility-based fusion of depth maps. In *Proceedings of the 2007 International Conference on Computer Vision*, ICCV '07. IEEE Computer Society, 2007.

12. R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. ilamps: Geometrically aware and self-configuring projectors. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06. ACM, 2006.

13. 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 the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 179–188. ACM, 1998.

14. S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

15. J. Underkoffler, B. Ullmer, and H. Ishii. Emancipated pixels: Real-world graphics in the luminous room. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 385–392. ACM Press/Addison-Wesley Publishing Co., 1999.

16. A. Wilson, H. Benko, S. Izadi, and O. Hilliges. Steerable augmented reality with the beamatron. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 413–422. ACM, 2012.