Localized Space Display

EE 267 Virtual Reality, Stanford University
Vincent Chen & Jason Ginsberg
{vschen, jasong2}@stanford.edu

1 Abstract

Current virtual reality systems require expensive head-mounted hardware, lead to motion sickness, and lack a type-based modality. As a result, VR adoption and use in workplace environments, where users need to type, talk, and switch between multiple computer applications, has been greatly hindered. We have developed a new method –The Localized Space Display (LSD)– to meet 3D-viewing needs in several significant use-cases. In our research, we demonstrate how such a display can be built using using only a traditional desktop monitor and two depth cameras. We also demonstrate the new functionality LSD can provide designers and engineers, working between 3D CAD software and 2D internal messaging systems, through new interactive affordances.

2 Introduction & Motivation

VR fails to provide an adequate tool for the daily user in the workplace environment, where users typically sit alone, use the computer several hours a day, and multi-task across multiple applications. In order to manage tasks and communicate among teams effectively, workplaces require a new 3D paradigm that does not fatigue, disrupt workflow, or necessitate additional hardware.

We have created the Localized Space Display (LSD), an unencumbered automultiscopic parallax-based head-tracking display that enables users to switch quickly between 3D and 2D interfaces. Though the 3D effect (motion parallax) is less pronounced than in a HMD, it is sufficient to provide benefit for designers and engineers working with 3D modeling software. With LSD, any traditional desktop monitor can be transformed into an augmented display. LSD enables users to easily switch perspectives, whether between keyboard and hand-recognition or 2D and 3D. For example, an engineer can grab a 3D model off a workbench, inspect/edit/annotate the model, and then upload the 3D model into a group chat.

3 Related Work

3.1 SpaceTop

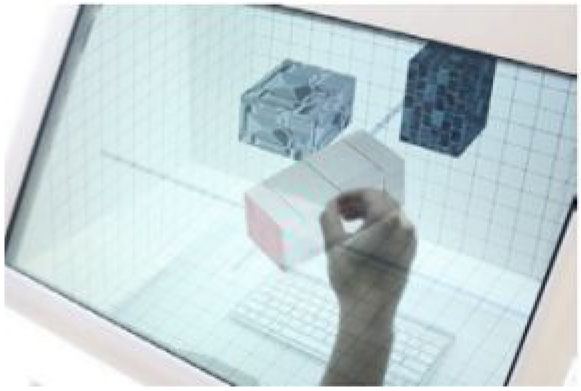


Figure 1: SpaceTop See-Through 3D Display. Image from [3]

SpaceTop is the primary inspiration for our interface. As seen in Figure 1, it features custom hardware, including a transparent screen upon which users can see floating UI elements. Space Top's goal was to introduce a means for direct 3D interaction that is 'fused' into a traditional 3D desktop interface [3].

The technology features two Kinect sensors, one for tracking the face and one for tracking hands. We wish to build our own version of this that does not rely on a see-through display but may work on any monitor. In our implementation, we enable the same interaction to work on any traditional desktop monitor rather than specifically see-through displays.

3.2 Wii Remote Head Tracking

In 2008, Johnny Lee hacked a Nintendo Wii IR controller to update a screen relative to the location of a user's head. This created a 3D parallax effect using only a traditional TV monitor, which resembles the 3D effect we seek to produce in any computer monitor. As Lee suggested, the resulting effect was like looking through your screen as a window into another world [4]. Our project goes further by allowing interactions with objects in this window through hand tracking.

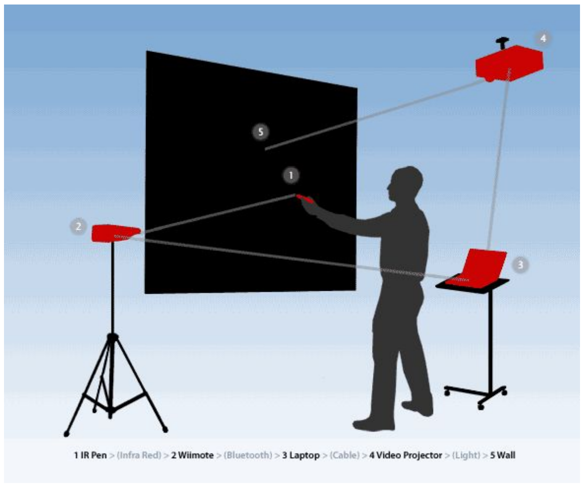

1
2
3
4
5
1 IR Pen > (Infra Red) > 2 Wiimote > (Bluetooth) > 3 Laptop > (Cable) > 4 Video Projector > (Light) > 5 Wall

Figure 2: Hardware Set-Up of Wii Remote Head Tracking Demonstration. Image from [4]

4 Methods

4.1 Head Tracking

We used the Kinect SDK for Windows to implement face-tracking for our system. In our implementation, we accessed the local coordinates of the based on the location of the Kinect, which we located at a position relative to the monitor of the monitor. This was an important metric for us to keep track of, because we would need to later calibrate these relative values based on the position of the Kinect (as discussed in section 5.4).

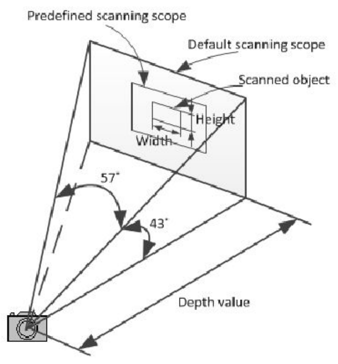


Figure 3: Camera Viewing Frustum of Microsoft Kinect 2.0. Image from [8]

In addition, we implemented a filtering mechanism to avoid tracking of multiple bodies by the Kinect. Our device only needed input based on the closest user, so we made it a priority to find the closest user, and consequently only track that user.

4.2 Gestural Recognition & Rendering

To implement gestural recognition, we used the Leap Motion SDK, which was a very straightforward way to integrate into Unity engine. The Leap Motion SDK provides basic hand-tracking based on the relative position of the physical hardware device. Consequently, we were able to import the hand models and manually calibrated their locations in relation to the real-world monitor, such that the experience felt genuine.

We ran into a few challenges with the hand-tracking of the Leap Motion. First, the Leap Motion has limitations in hand-tracking, especially related to occlusion. For example, when a user's hand was flipped upside down, the Leap Motion would lose its ability to track fingers. When a user's hands were clasped, the device would lose its ability to track discrete fingers.

In addition, we needed to carefully consider the placement of the Leap Motion, because if the Kinect was placed directly behind it, hand gestures would disrupt the Kinect's head-tracking. As a result, we ultimately offset the location of the Kinect so that an unobstructed ray could be traced from the Kinect to the user's face.

4.3 Parallax Effect

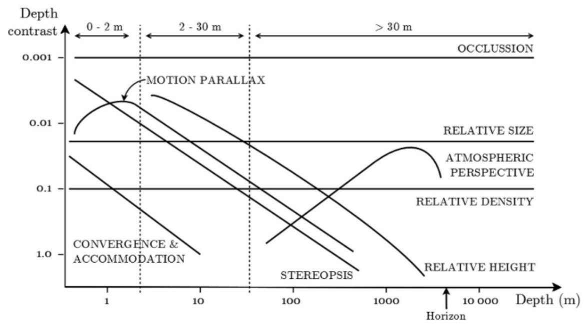


Figure 4: Depth contrast as a function of the log of distance from the observer. Image from [1]

The parallax effect was an essential component to the success and functionality of our device. This was a function of the users' ability to perceive depth based on relative size (Figure 4). Because we used a 2D monitor display, parallax was the only way to create a 3D effect, which was instrumental to the functionality and rationale of our interface, as demonstrated in Figure 5 below.

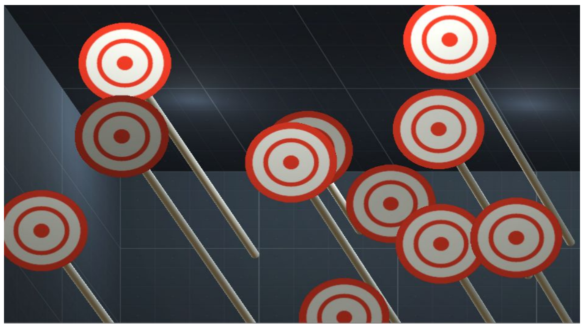


Figure 5: Parallax effect observed from the demo scene.

One key factor that we discovered to be very important to the parallax effect was the texture and depth effect of the scene. To support this finding, we created a grid texture with special lighting effect to emphasize the depth effect for parallax.

In addition, we learned that by simply tracking the scene camera to the location of a user's head, the parallax effect was not achieved. As a result, we implemented the an asymmetric view frustum, highlighted in section 4.5.

4.4 Asymmetric View Frustum

In order to create the parallax effect, we dynamically changed the FOV to match the virtual camera and viewer head. We also dynamically changed the shape of the frustum into an asymmetric shape. This constrains the display to single user support and requires manual input of the screen size and camera position relative to the center of the screen, as seen in Figure 6.

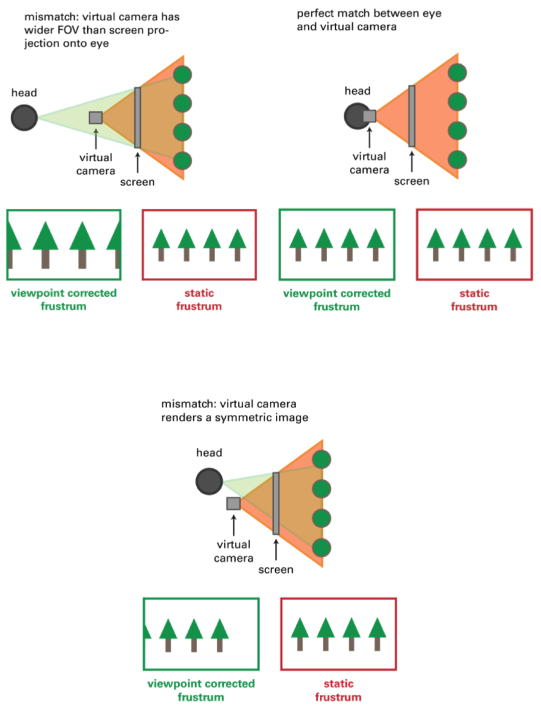
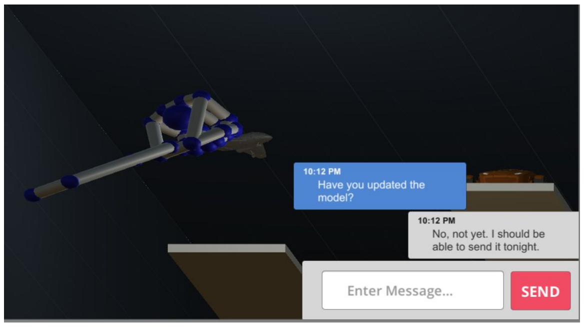

mismatch: virtual camera has wider FOV than screen projection onto eye
head
virtual camera
screen
viewpoint corrected frustrum
static frustrum
perfect match between eye and virtual camera
head
virtual camera
screen
viewpoint corrected frustrum
static frustrum
mismatch: virtual camera renders a symmetric image
head
virtual camera
screen
viewpoint corrected frustrum
static frustrum

Figure 6: Diagrams explaining the asymmetric viewing frustum. Our implementation was based on the bottom image.

4.6 User Interface Design

Our primary demo features a scenario that highlights the utility of the Localized Space Display. We feature what appears to be a "work shed," where multiple shelves are presented (in parallax) to the user. Upon the shelves are 3D models, which the user has the ability to manipulate. Overlayed on the user's screen, is a messenger application, where they can still interact and type like they would in a conventional work setting.

The scene featured two primary components, the GUI and the 3D interaction model. Both were implemented from scratch in Unity.


10:12 PM
Have you updated the model?

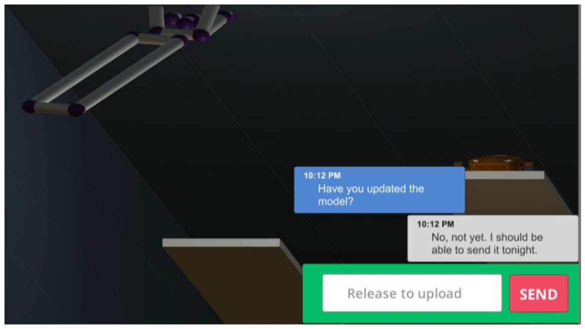
10:12 PM
No, not yet. I should be able to send it tonight.

Figure 7: User's left hand is "active". They can now grab the model airplane in the scene.

4.6.1 Interaction Model

By integrating the parallax effect and gestural control (discussed in previous sections), we successfully implemented an interaction model that allowed users to translate, rotate, and scale virtual objects. This model was based on the `AbstractHoldDetector` class from Leap Motion. In essence, the Leap Motion API provides us with the ability to track "hold strength", based on the configuration of a user's fingers and hands. We defined a specific threshold for "hold strength" that would indicate an *active* state for the both the left and right hands. When active with one hand, users could grab objects, as seen in Figure 7. When both hands were active, users could scale and rotate objects.

As a part of this interaction model, we implemented a mechanism to allow users to interact with the 2D GUI in the scene. By dragging the model up and out of the frame of the scene, users trigger a prompt that indicates the ability to "up"-load any objects to the messenger interface, as seen in Figure 8. Upon releasing the object, the model appears to be virtually beamed through the messaging platform, not unlike other familiar processes of uploading files.


10:12 PM
Have you updated the model?

10:12 PM
No, not yet. I should be able to send it tonight.

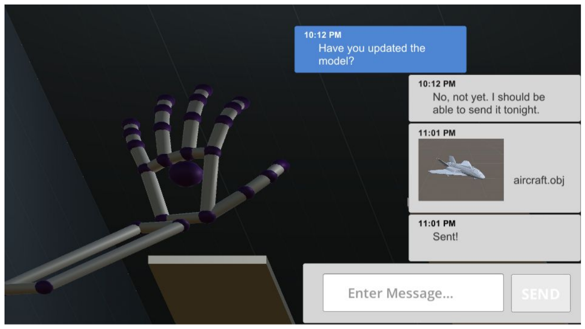
Figure 8: Users have the ability to send files (3D models) and messages through the UI.

This user experience follows the "drag-and-drop" paradigms that individuals may have used on desktop or web computing devices.

4.6.1 2D GUI

At any point in time, users have the ability to type in the messenger interface pictured in Figure 8. This 2D GUI is fixed to the user's screen, like any 2D interface would be on a desktop display. This GUI simulates any popular messaging app that might be useful for work settings (i.e. Slack), as shown in Figure 9.

This feature was also constructed from scratch using Unity GUI elements.


10:12 PM
Have you updated the model?

10:12 PM
No, not yet. I should be able to send it tonight.

11:01 PM
aircraft.obj

11:01 PM
Sent!

Figure 9: Users have the ability to send files (3D models) and messages through the UI.

5 Evaluation

5.1 Motion Parallax Effect

Though our head-tracking system produced motion parallax, stereoscopic vision cues greatly reduced the effect. Nawrot et al. [2014] performed a matching experiment in which they compared perceived depth as induced by motion parallax to equivalent depth resulting from binocular disparity. In this study, Nawrot et al. observed a near-tenfold depth foreshortening. We corroborated this perceived 3D depth discrepancy by using the display with monoscopic vision (either by viewing the display through a camera or by blocking one eye).

5.2 Gestural-Control

The Leap Motion Hardware posed several constraints for our interaction model. Specifically, the Leap Motion failed to render the hands in the case of self-occlusion; had a small range for which it could reliably detect hands; and encountered numerous errors when tracking the motions of the middle and ring fingers. The introduction of multiple hands into a scene merely exacerbated these latent issues.

5.3 Background Noise

We filtered the Kinect depth camera to only track the nearest person. Nevertheless, large, crowded, and open spaces led to interruptions in the tracking, which reoriented the view frustum and disoriented the user.

5.4 Manual Camera Calibration

In order to properly create the view frustum for the scene, we needed to take account the distance and orientation of the user's head relative to both the Kinect camera and desktop monitor. Though generally static, in the few situations where the Kinect camera was moved, rotation and position parameters required manual calibration to maintain the centering of the interface and 3D effect.

5.5 Sensor Edge Cases

The Kinect depth camera had a minimum range of 1 meter. When paired with the Leap motions maximum range of several inches, the relative locations among the two depth cameras and monitor were greatly constrained. This posed greater problems when the user would sit near the computer –as we intend for the workplace environment.

5.6 Common UI Problems

The affordances of interactive vs noninteractive objects proved ambiguous when we tested our demonstration with users. In particular, the motion of moving a 3D object to the 2D application, required additional instructions.

6 Discussion

Localized Space Display is a novel approach to combining 2D and 3D interfaces. Through our experiments, we've found that there is certainly an advantage to enabling spatial memory on a natural interface that is still connected to modern, familiar desktop applications.

Modern, commercial VR systems provide access to compelling spatial interaction models. While these systems are advantageous in terms of tracking and precision, Localized Space Display eliminates the need of physical hand controllers. In effect, it provides a remarkably low-friction approach to accessing 3D interactions.

While the system is not as immersive as a technology like VR, immersion to excess is counterproductive, especially in office settings or high-collaboration environments.

Ultimately, by bridging the gap between 3D and 2D, you gain new perspectives for viewing, transforming, and communicating additional dimensions of information, without losing widespread computing conventions present in modern mobile and desktop devices.

Localized Space Display introduces a unique "upload" interaction that combines a 3D interaction model with familiar messaging interfaces. This is just the start. There are numerous equally exciting interfaces that take advantage of space as well as well-known UX paradigms.

Future Work

Besides improving the demonstration environment, three primary interventions could have greatly augmented the experiences.

First, automatic camera calibration. This would have adjusted the Kinect to adjust the asymmetric frustum dependent on its relative to position and orientation to the user.

Second, we came to realize that depth tracking added little to the 3D effects compared to 2D head tracking. A web camera, utilizing OpenCV to mark facial position, could potentially produce a similar 3D effect with even less required hardware. The web camera would also eliminate the needs for calibration or minimum user range.

Finally, tilting the UI environment so that it is perpendicular to the ground regardless of the monitors tilt angle would greatly improve the 3D effect. In this case, the added tilt would contradict the normal viewing cues of 2D displays, and further differentiate between the 2D and 3D interfaces.

Acknowledgements

We would like to thank the fantastic teaching team of EE 267. Thank you to Gordon Wetzstein, Robert Konrad, Keenan Molner, and Hayato Ikoma for providing guidance and (of course) hardware, throughout our project and the entirety of EE 267.

References

[1] CUTTING & VISHTON," Perceiving layout and knowing distances: The interaction, relative potency, and contextual use of different information about depth", Epstein and Rogers (Eds.), Perception of space and motion, 1995

[2] DIEGO MARTINEZ PLASENCIA, EDWARD JOYCE, AND SRIRAM SUBRAMANIAN. 2014. MisTable: reach-through personal screens for tabletops. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM.

[3] JINHA LEE, A. OLWAL, H. ISHII, AND C. BOULANGER. SpaceTop: integrating 2D and spatial 3D interactions in a see-through desktop environment. In Proc. of CHI 2013, pages 189–192

[4] J. C. LEE, "Hacking the Nintendo Wii Remote," in IEEE Pervasive Computing, vol. 7, no. 3, pp. 39-45, July-Sept. 2008.

[5] J. C. LEE. (2007, Dec 21). Head Tracking for Desktop VR Displays using the WiiRemote [Video File]. Retrieved from https://www.youtube.com/watch?v=Jd3-eiid-Uw

[6] NAWROT, M., AND STROYAN, K. 2009. The motion/pursuit law for visual depth perception from motion parallax. Vision Research 49, 15, 1969–1978.

[7] N. BOGDAN, T. GROSSMAN AND G. FITZMAURICE, "HybridSpace: Integrating 3D freehand input and stereo viewing into traditional desktop applications," *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, Minneapolis, MN, 2014, pp. 51-58.

[8] Z. ZHANG, M. ZHANG, Y. CHANG, C. CHASSAPIS, Real-Time 3D Model Reconstruction and Interaction Using Kinect for a Game-Based Virtual Laboratory