OculusKinect: Real Time Augmented Reality Visual Assistance System

Stephan Baulch, Nathan Fuchs, Brandon Walker

North Carolina State University

Introduction

Virtual reality devices and augmented computer vision tools have been available for quite some time now, but the Oculus Rift and Microsoft Kinect have brought the affordability that allows consumers, tinkerers and undergraduate researchers to get their hands dirty with augmented reality. With these two products, we set out to explore some new possibilities in augmented reality. We were able to create several tools for these headset wearers We set out to find interesting, useful, and innovative ways to take advantage of the combination of these new products. Some initial ideas included creating models of the world in real-time with kinect fusion and being able to render objects into the view of the Oculus Rift as if those objects were actually there in front of the user. A similar project had been done before in a microsoft sponsored research project called Kinect Fusion which is what initially inspired us to attempt this project. We thought it would be a great idea to bring the real world interaction similar to that in Kinect Fusion to be visible through the lenses of a virtual reality headset.

Hardware

Connecting the Microsoft Kinect to the Oculus Rift was a challenge. The initial design of two kinects mounted vertically in front of each eye would have been even more difficult, and was what initially led us to consider 3d printing as a solution. A friend was able to lend us the use of his personal 3d printer and we designed a system to mount the two devices together with the single-kinect design. Some of the parts were models downloaded from a website called thingiverse.com which aggregates 3d files that are good for printing. We opted to use two oculus rift brackets for lateral stability and a custom 90 degree angle bracket that acted as a tripod mount. We then attached a tripod mount to the bottom of the kinect to be compatible with the tripod bracket. This bracket underwent several prototypes.

Figure 1 - A few of our faulty bracket designs.

The final result used the clip and peg holes available in the bottom of the kinect to semi-permanently clip in. It also has a hole in the bottom that is tapped like a screw hole in the bottom of most cameras. This bracket allows for any tripod-compatible camera to be mounted including the Kinect and it's bracket. Testing this design showed the bracket system was able to withstand significant head shaking and movement, however it failed with rapid lateral head rotation (double-take). This "sliding" error was simply fixed by applying a layer of hot glue to the inside of the top/bottom of the oculus rift bracket and allowing it to dry. This created a rubber-like surface that gripped much more strongly.

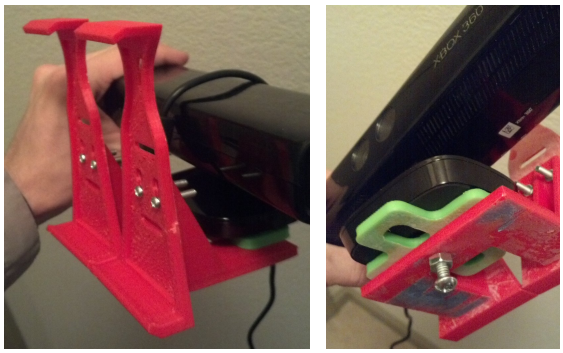


Figure 2, 3 - Back side of the Oculus rift brackets, and an underneath shot of the tripod mount.

OculusKinect

The OculusKinect SDK consists of a combination of use of the Microsoft Kinect SDK as well as JMonkey Engine with support for the Oculus Kinect. Most Kinect and Oculus Rift calls are done through C libraries called via Java Native Interface.

Color Blind Mode

After creating a system that is able to feed augmented reality video streams back to users seamlessly, we naturally had the idea to use the system in order to provide visual transforms for users. For OculusKinect, we primarily focused on providing color blind filters for dichromacy color blindness types: Protanopia, Deuteranopia, Tritanopia. To accomplish this, as the system copies image data from the data buffer to the Java BufferedImage class, we run the pixel's RGB color value through a method that determines the shift in hue for a specific type of color blindness for that pixel color. Once the change is calculated, the color is then inserted BufferedImage. The main purpose of this feature is not only to mimic the color spectrums for users, but to provide them with ways to view their visual interfaces as those with color blindness do. With this feedback they can modify their UIs to better suit these types of users. An extension of this feature would to implement an actual assistance enhancement filter for those with colorblindness in order to help them view the world better.

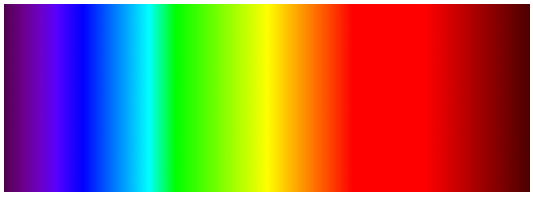


Figure 4 - Color spectrum without OculusKinect filter.

Figure 5 - Color spectrum filtered for Protanopia.

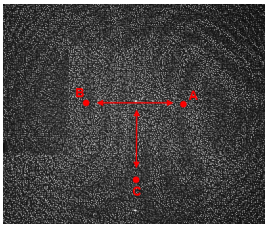
Figure 6 - Color spectrum filtered for Deuteranopia.

Figure 7 - Color spectrum filtered for Tritanopia.

Depth Data

The Microsoft Kinect uses a unique pattern of projected infrared lasers to calculate the depth of each pixel in the image stream. This data is then presented to the user to assist in estimating distances. The system is able to accurately provide data anywhere from 1.5 to 13 feet, accurate to less than 1 centimeter and provide this information in a HUD through the Oculus Rift. Since the depth sensor relies on infrared beams emitted from the Microsoft Kinect, it also works in the dark.

Along with displaying simple depth data, this data can be used to calculate the normal vector of surfaces. Three points are chosen near the center of the image, approximately 30-50 pixels apart. If the points are chosen further apart the accuracy is increased, but smaller surfaces cannot be calculated.



The three points form a triangle. The distance from the Microsoft Kinect sensor to each of the three points is known. The field of view is also known. The horizontal field of view is 58° and vertical is 48°. Since the distance to points A and B are known, and the angle between them can be calculated from the pixel distance and the field of view, the law of cosines can be used to calculate the distance between A and B (this will increase as the target gets further away).

$$\overline{AB} = depth(A) + depth(B) - 2 * depth(A) * depth(B) * cos(viewAngle)$$

Once the distance between A and B is found, the law of sines can be used to calculate the angle between the users viewpoint and the plane of the target object.

$$angle = sin^{-1}(depth(A) * sin(viewAngle) / \overline{AB})$$

To demonstrate this feature, we shine a virtual laser on the target object and project the reflection off of the surface.

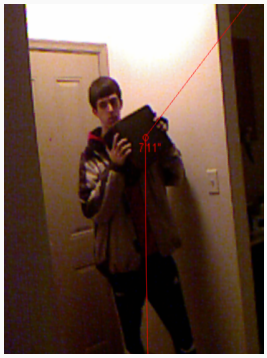


Figure 8 - The virtual laser is being reflected to the left of the target object

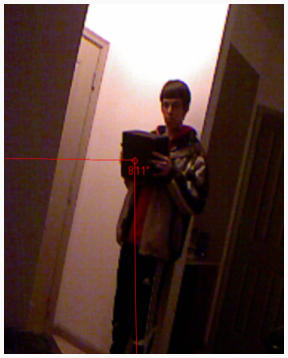


Figure 9 - the virtual laser is being reflected to the right of the target object

Night Vision Mode

One of most interesting features of the Kinect's use of it's infrared camera is the fact that it can be used for night vision. With the OculusKinect we can harness this feature and pull the infrared video into the Oculus Rift in order to give the user the effect of night vision goggles.

Lessons Learned and Future

The planned use for the Kinect throughout the course of the project had drastically changed from the first couple of iterations to the final product. For example, the original concept for OculusKinect involved using two kinects in order to pull separate video streams into each eye. This designed has several benefits, most notably a doubling of the frame rate, larger field of vision, and depth perception for users. Unfortunately, even with the assistance of the servos with the Kinect, it was too impractical to have the separate Kinects converge on objects for the users to view them correctly. Without the correct convergence, users would see the world as if their eyes were "crisscrossed" and would be unable to focus on any object inside or outside the range. Due to this restriction, we scaled the concept down to a single kinect. The project was still functional, sans some depth perception, however optimally separate video streams for each eye would be an incredible feature to add in future research. Furthermore, as mentioned previously in our color blind feature segment, the ability to help assist those with dichromatic color blindness to see troublesome scenery could be a key area for future research. Finally, one of the last features that would be key to future research would be the use of the Kinect Fusion library to input the environment via the kinect into the JMonkey Engine, using point cloud meshes in order to use real physics on in game objects. All of these areas also expose the problem with the form factor of the OculusKinect, which requires a central computer as well as a power source for both the Oculus Rift, Kinect, and Computer in order to function properly. Any advancement in either the reduction of the size or enhancement of the speed of the central computer component would be necessary to make the OculusKinect practical for any consumer environment.

Figure 10,11 - One Kinect per eye

Conclusions

We were able to accomplish three primary feature goals with our OculusKinect product. Infrared vision was enabled with the Kinect's infrared emitter and camera. We created a filter that simulates color blindness by scaling rgb values of each pixel based on different masks. Finally, we implemented a demonstration of interaction with the real world by simulating balls bouncing off of real world objects as well as lasers reflecting off of surfaces. Further research with this product would involve color correction for assisting colorblind people, more realistic environment rendering control with real-time modeling, and progress towards separate image feeds for each eye for true 3d vision.

Sources

http://www.microsoft.com/en-us/kinectforwindows/

http://jmonkeyengine.org/

https://code.google.com/p/jmonkeyengine-oculus-rift/

http://research.microsoft.com/pubs/155416/kinectfusion

-uist-comp.pdf