EE267 Proposal: Virtual Occlusions in Video See-Through Immersive Augmented Reality

Rahul Prabala
rprabala@stanford.edu

Alex Bertrand
abert13@stanford.edu

May 25, 2017

1 Motivation

Augmented Reality (AR) allows the viewer to interact with the world at large in new and exciting ways, using the backdrop of reality as a canvas. With advances in modern computing technology, it is now possible to construct immersive augmented reality devices with very small constituent parts. In an effort to improve realism when viewing augmented scenes, it is possible to combine the depth maps of the virtual scene with that of reality captured using a depth camera in order to introduce occlusions between virtual and real objects. The scene with occlusions can then be rendered onto a display, facilitating see-through video augmented reality.

2 Related Work

[1] introduces an immersive augmented reality system with many sensors to provide and improve user experiences. [2] provides excellent taxonomies of augmented reality devices, and lists several examples of each including use of video see-through AR and optical see-through AR. [3] uses video see-through AR to enhance user experiences reading books.

3 Project Overview

In this project, we aim to improve upon existing video see-through augmented reality devices by demonstrating that it is possible to construct such a system using off-the-shelf devices into a small form-factor. While ostensibly a prototype, our final product has the potential to greatly reduce the area and power consumption of video see-through AR while retaining scene enhancement.

4 Hardware Design

We will use three devices in the construction of our small-form prototype. The NVIDIA Jetson TX1 acts as the "brain" of the system, containing both a CPU and GPU. The Intel RealSense camera generates an RGB view of the scene as well as a depth map to be integrated with the virtual depth. Lastly, we will use the head-mounted display to visualize the scene.

5 Software Components

The two main software components (not including the Linux kernel and driver modifications to integrate the RealSense) are the virtual scene and camera data capture. These two are written in entirely different languages, and thus communication between the two will need to be constructed in order to render onto the HMD.

6 Demonstration and Application

To demonstrate the occlusions, we will construct a simple virtual scene consisting of a single planar object located at a fixed depth. Then, using the RGB and depth data, we will occlude the virtual object for those pixels that are closer to the viewer than the object. We will also add the ability for the viewer to change the depth of the virtual object using the keyboard. A visualization of this demo is shown in Figure ??.

7 Milestones, Timeline, and Goals

The rough outline of the project is as follows. First, the camera needs to be integrated with the Jetson TX1. Rahul worked on this in EE367, and so does

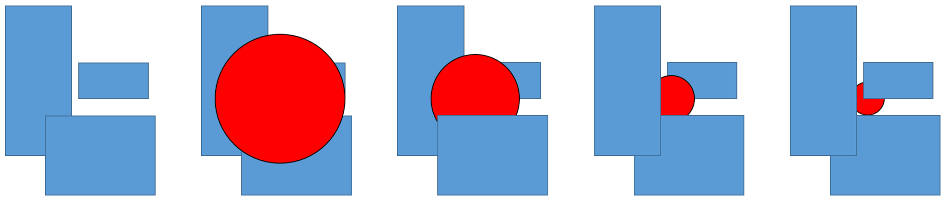
Figure 1: Demo visualization, where the red ball represents the virtual object.

not anticipate this taking much time. Then, the virtual scene needs to be created. Once this is done, the last step is to pass data from the camera to the javascript-based virtual scene. The rendering will be performed as in the homework, and the occlusion will take place in the fragment shader to achieve improved throughput.

Though the TX1 is a very powerful GPU-based system, achieving real-time performance is very challenging due to its embedded nature. While the RealSense can generate streams of data, moving those into the browser is ultimately the bottleneck due to the differences in paradigms. To get around this, we propose to communicate between the two using the file system. This is inherently slow, as we pay the I/O cost of reading and writing a full frame for every frame of output. However, we believe that this limitation will be sufficient to demonstrate proof of concept, and if we have time, will explore optimizations to improve the framerate.

A list of goals and associated timeline is below:

7.1 Integrate RealSense with Jetson TX1 (end of week 1)

1. Recompile kernel with modifications
2. Build depth and rgb streaming application

7.2 Build Virtual Scene with Keyboard Hook (end of week 1)

1. Modify existing scene to include planar object
2. Add keyboard functionality to move object within the virtual scene

7.3 Depth Map Integration (end of week 2)

1. Investigate alternatives to file based communication
2. Use files to capture a single frame of RGB and depth data
3. Read file into browser and store within framebuffer
4. Pass the two buffers (RGB and virtual scene) to fragment shader to handle occlusion

References

[1] T. Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway, "Exploring mars: developing indoor and outdoor user interfaces to a mobile augmented reality system," *Computers & Graphics*, vol. 23, no. 6, pp. 779–785, 1999.

[2] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented reality: A class of displays on the reality-virtuality continuum," in *Photonics for industrial applications*, pp. 282–292, International Society for Optics and Photonics, 1995.

[3] M. Billinghurst, H. Kato, and I. Poupyrev, "The magicbook-moving seamlessly between reality and virtuality," *IEEE Computer Graphics and applications*, vol. 21, no. 3, pp. 6–8, 2001.