3-D Interactive Visualization Platform for Ray Aberration Generator

Luke Ender,[1,2] Sunglin Wang,[1] Brandon Hellman,[1] and Yuzuru Takashima[1]

[1] *College of Optical Sciences, University of Arizona, 1630 E University Blvd Tucson, AZ 85721*
[2] *Department of Physics, Whitworth University, 300 W. Hawthorne Rd., Spokane, WA 99251*

(Dated: March 24, 2017)

The Ray Aberration Generator (RAG) developed by the Takashima Lab is an excellent tool for teaching students how real-world ray aberrations manifest, enabling them to more fully understand optical systems and correctly identify and treat aberrations. However, the entire RAG is far too bulky to be mobilized, limiting its potential as a pedagogical device to only those on-site. To resolve this limitation, a remote viewing system was created which does not sacrifice either the 3-D visualization or the spatial interactivity of the RAG. The system was developed for the Google Cardboard platform, utilizing a uFactory uArm robotic arm and two HD webcams such that students anywhere in the world with access to an Android cellphone with a gyroscope and an internet connection can view a live 3-D video and spatially interact with the system.

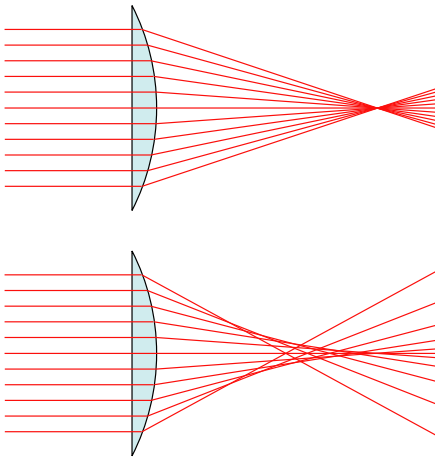

FIG. 1. A two-dimensional ray-trace of incident light passing through a planar-convex lens without aberration (top) and with spherical aberration (bottom). Such 2-D ray-traces are limited in their comparability to real-life aberration.

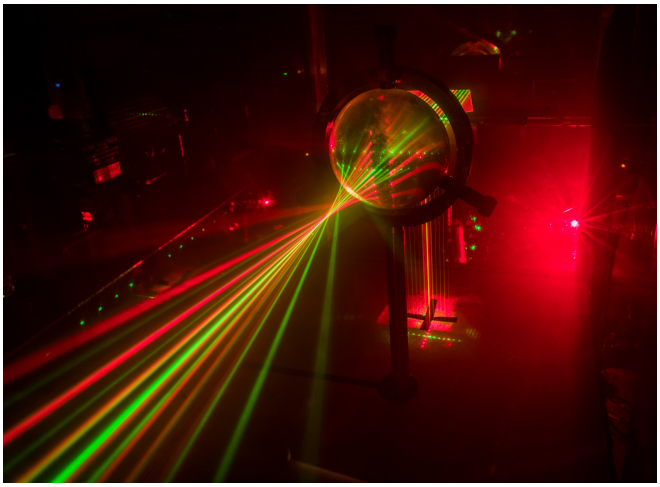

FIG. 2. The RAG 1.0 producing an example of spherical aberration in summer 2015. Photo by Chris Summitt.

I. INTRODUCTION

The term "optical aberration" designates any time the wavefront of light in an optical system deviates from the theoretical, optimal form needed for perfect imaging. Optical aberrations take many forms and are present in all real optical systems, and so the ability to recognize and treat such aberrations is critical in the field of optics. However, current teaching tools in this area are lacking and are not sufficient to learn how optical aberrations manifest themselves in real optical systems (see figure 1). In order to fill this gap, over the past years the Takashima Group at the College of Optical Sciences at the University of Arizona has developed a Ray Aberration Generator (RAG) designed as a teaching device for beginning students of Optics. In contrast to 2-D ray-traces and other commonly available teaching tools for optical aberrations, the RAG produces a real, 3-D example of a ray aberration which is much easier to connect to how ray aberrations look in real systems (see figure 2).

In summer 2015, the first working version of the RAG was completed and showed itself capable of producing rather impressive instances of ray aberrations. However, the setup was quite heavy and not easily portable, and so its potential as a pedagogical was not yet fully tapped, as it was only available those with physical access to it. In order to surmount this limitation, a remote viewing platform utilizing the Google Cardboard platform was developed which compromised neither the 3-D display of the aberration nor spacial interactivity with the system.

II. METHODS

Our setup consisted of two Logitech c920 HD webcams, a uFactory uArm robotic arm, an Arduino UNO, an Android smartphone with a gyroscope (in our case a Google Nexus 5X), a laptop computer, and a virtual private server with a static IP. The two webcams were affixed to one another using a setscrew screwed into both

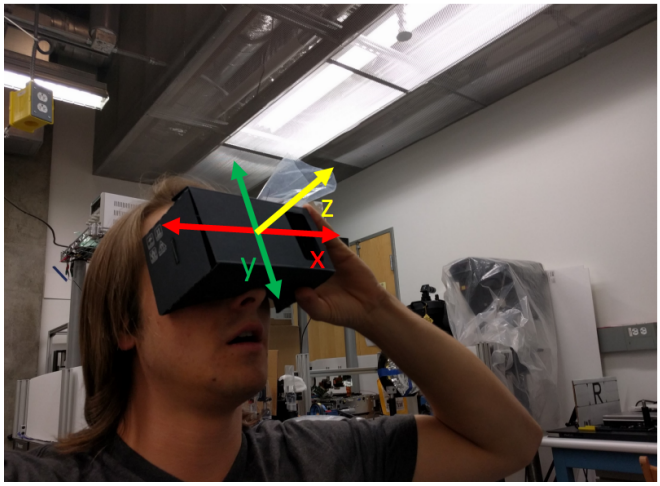

FIG. 3. A user wearing the Google Cardboard setup. An Android smartphone is placed in the cardboard holder and held up to the face such that half of the screen is visible to each eye, creating a 3-D viewing platform. Rotation around the x and y axes correspond to movement of the arm along one of its two degrees of freedom. (see figure 4)

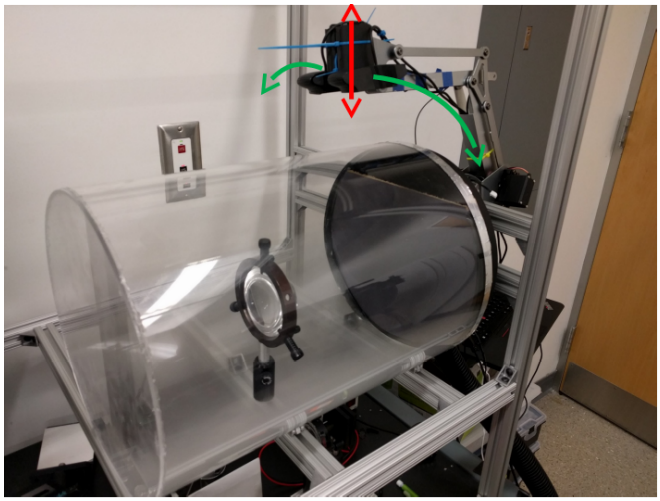

FIG. 4. The uArm and camera setup for the RAG 2.0. The arm is free to move about both of the degrees of freedom shown.

of their tripod mount screw holes, and then mounted on the uArm, which was in-term mounted on the RAG as seen in figure 4. The communications system between all the different parts is detailed in figure 5. When the user turns their head around any of the axes as shown as figure 3, the robotic arm moves along one of the degrees of freedom shown in figure 3. At the same time, the video stream from the camera mounted on the arm is streamed in real time to the Android phone in the Google Cardboard setup, creating a real spacial interaction experience.

This was achieved by first writing a very simple program for the Arduino UNO which reads the two angles via asynchronous serial which are to be written to the uArm, computes the PMW signal corresponding to that angle, and outputs that signal. This loops constantly, updating the uArm angles whenever new data is available. Then, a simple Android application was written which reads gyroscope sensor data, opens a TCP/IP port, and fires said data into that port. In order to avoid the NAT system here at the University of Arizona and any potential NAT system on the Android phones end, a VPS server with a static IP address was rented and a simple Python TCP server written, so that when run on the VPS the laptop and the Android phone can both independently connect to an open port on the VPS, and all data sent from the Android phone will be forwarded to the laptop. This form of TCP hole-punching proved quite effective, and it allows us to establish communication between the Android phone and the laptop despite the fact that neither device is directly accessible outside of the local network. Finally, a C++ program for use on the laptop was developed which read in the gyroscope data via a TCP socket, integrated the data, and produced the corresponding angle which was then sent to the Arduino UNO via serial. The use of gyroscope data was decided instead of other embedded inertial sensors such as the accelerometer due to support from both literature [1] and experiment that only the gyroscope has a low enough error to be useful.

On the video side, first the two video streams are captured, cropped, rotated, combined, and streamed to the VPS using the open-source program FFmpeg. The open-source webserver nginx was then run on the VPS configured such that any client that connects to the VPS will receive the RTMP stream streamed being streamed to the VPS. Finally, the video is then played using any Android-platform video player.

III. RESULTS AND DISCUSSIONS

The system works quite satisfactorily, creating a one-to-one rotation motion experience. However, there lies one problem which remains to be solved in the system – latency between the user moving their head and the video feed responding to that movement. The program FFmpeg was used to capture, crop, rotate, and stream the video streams from the webcams, and it has been found to introduce around 1-1.5 seconds of lag. This is an acceptable amount of lag, as it still provides a decent user experience, however, there is yet another source of lag in our system which drives this number out of the acceptable threshold – the buffering of the video play on the Android side. The Android system has a hard-coded method for buffering and playing video streamed via RTMP, and this includes an around 5 to 6 second lag before beginning to decode the video. This is, in theory, an easy problem to overcome – simply change the size of the video buffer. However, this has proved to be much more difficult in practice. For a more experienced

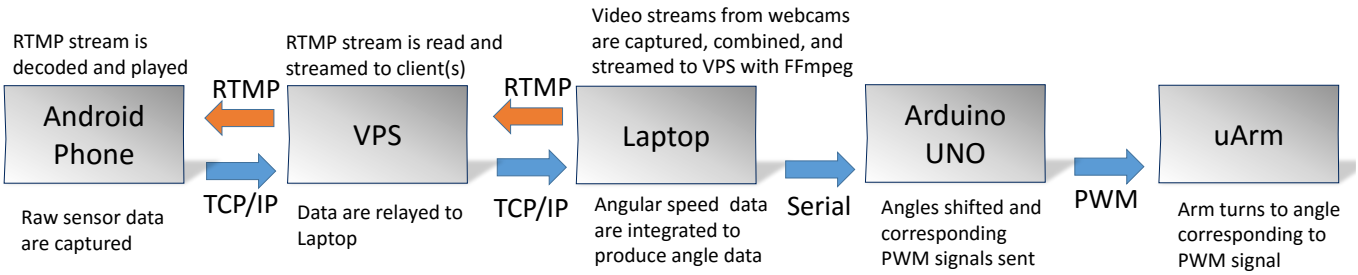

FIG. 5. A diagram of the communication system used in our setup. Blue arrows denote sensor data stream flow; orange video stream data flow. Acronym key: RTMP Real – Time Messaging Protocol, TCP/IP – Transmission Control Protocol / Internet Protocol, VPS – Virtual Private Server, PMW – Pulse Width Modulation

Android developer, this problem should be easily surmountable, but it is yet to be fixed as of the end of this program.

IV. CONCLUSIONS

While the system we developed is fundamentally working, latency remains an issue, and it appears this latency can only be reduced under 1-1.5 seconds by modifying the program used to capture the camera feeds, FFmpeg, as this latency has been shown to originate almost exclusively from this initial stage. Additional latency is introduced due to buffering when playing the video on the Android side, but this should be easily surmountable by an experienced Android developer.

Beyond the use case described in this paper, this system is highly expandable. In the future, any variety of vision device could be placed on the robotic arm, such as a LIDAR system or similar. There are many uses for such a head-rotation tracking user experience.

V. ACKNOWLEDGMENTS

The authors would like to thank the Nation Science Foundation for providing the grant money that allowed this project project to take place, NSF grant #EEC-1004331. In addition, the University of Arizona, College of Optical Sciences deserves recognition for providing the facilities and organizing the Research in Optics program. I, Luke Ender, would further like to thank Sunglin Wang and Brandon Hellman for their continued support, Dr. Yuzuru Takashima for his guidance on the project, and finally Melissa Sarmiento Ayala and Dr. John Koshel for making this program possible.

[1] Woodman, O. J. (2007). An introduction to inertial navigation. Technical Report UCAM-CL-TR-696, University of Cambridge, Computer Laboratory.