Archives of Business Research – Vol. 13, No. 01
Publication Date: January 25, 2025
DOI:10.14738/abr.1301.18207.
Kurani, P. (2025). Vista Eyes. Archives of Business Research, 13(01). 58-70.

Vista Eyes

Prachit Kurani

ABSTRACT

This project aims at designing and developing an augmented reality application which will enable us to visualize and debug networks using an overlay of network information on the network devices. We propose an augmented reality technique to achieve this. Using 2D/3D image tracking offered by various libraries, the network devices can be identified by the system. Once the device is identified, it can be queried for various network parameters such as ping response time, number of open connections, amount of data transferred per second, number of network interfaces etc. The data can then be aggregated and analyzed. Once analysis is completed, visual representations of this data can be streamed to a VR device such as the Oculus/google cardboard. The oculus/cardboard then overlays this data on the network device itself.

Keywords: Augmented, Google cardboard, Oculus, Networks, OpenGLES2.0, 3D, Vuforia.

INTRODUCTION

Computer networks have proliferated everyday life. We have lots of network devices even in small home networks. It has become increasingly difficult to debug these problems as the networks grow. This project deals with real time augmented reality which helps in visualizing the flow of network using google cardboard.

Symbols and images are centrally wired in our brains. They compose the foundation upon which we communicate with one another. Think about it: Any word you read is associated with things you've seen or heard. We believe in "seeing is believing" and Vista Eyes App aims to bring this concept into reality.

"Have you ever faced difficulty understanding and debugging networks? "Just by looking at an issue, it becomes very hard to analyze the problem and understand how and why packets flow in a particular fashion. Imagine seeing packets flowing from your router to the computer or analyzing networking concepts in augmented reality.

From a practical point of view, I want to create a meaningful augmented reality App proving you can experience the technology to its fullest for minimal expense and without hooked up to a clunky helmet. Using 2D/3D image tracking offered by various libraries, the network devices can be identified by the system. Once the device is identified, it can be queried for various network parameters such as ping response time, number of open connections, amount of data transferred per second, number of network interfaces etc. The data can then be aggregated and analyzed. Once analysis is completed, visual representations of this data can be streamed to a VR device such as the Oculus/Google cardboard. The oculus/cardboard then overlays this data on the network device itself.

This project aims at creating an experience, a new way to visualize and debug networks using an overlay of network information on the network devices. I propose an augmented reality technique to achieve this. Your phone and tablet can be so much more than they already are. Vista Eyes will deliver a platform which can be used for educational purposes and personal use.

COMMERCIAL OPPORTUNITY

Augmented and Virtual reality has tremendous potential and is expected to reach $150 Billion in 2020 [3]. This industry has vast applications from educational purposes to medical field. This project mainly targets two areas viz. as a new educational tool and as a tool for network administrators to quickly and easily visualize complex computer networks.

According to Vartan Gregorian, President of the Carnegie Corporation of New York, investing in education is key to America’s future success [1].

Vista Eyes is designed and built for multiple purposes. But our product is focused at educational institutes. Private networking courses to university level network analysis and debugging courses require enhanced tools and software to be effective at teaching students. Vista eyes is a one stop solution for all network visualization and debugging needs. It also explores latest human computer interaction method and is highly cost effective since it utilizes the cheap Google cardboard platform armed with an Android device. Most institutes already have this infrastructure setup and can easily employ the use of Vista eyes with minimal expenditure on hardware and software.

The forecast of revenues expected for these upcoming technologies in the coming years. Generating a foothold into this niche today, guarantees that such a project can earn a good revenue as the year’s progress,

Additional functionality will be added to this system, making it easier to use and much more versatile, thereby making it a much sought after product not only by schools and universities, but also by large data centers and even home network enthusiasts. Our company can bank on increasing revenues from app downloads across platforms and may move on to develop custom solutions for paying customers. Future plans may also include production line augmented reality and even custom-made augmented reality visors for special purposes.

Project Justification and Innovation

With respect to current generation, it is very important to incorporate audio visual learning. Augment reality applications came into existence in early 2015. To simplify the need to grasp new items became easy with these applications. Many applications have stored database which is indirectly hardcoded data. This project aims to make an application which is real time.

It is very easy to understand the concepts of network if we can visualize it in real time. It is very difficult to visualize and debug networks using overlay network information on network devices. We propose to employ an augmented reality technique to achieve this.

Preliminary research indicates that there exist multiple AR libraries viz- Vuforia, MetaIO SDK, D’fusion, ARmedia and Droid AR. Using 2D/3D image tracking offered by these libraries, the network devices can be identified by the system. Once the device is identified, it can be queried for various network parameters such as ping response time, number of open connections, amount of data transferred per second, number of network interfaces etc. The data can then be aggregated and analyzed. Once analysis is completed, visual representations of this data can be streamed to a VR device such as the google cardboard. The cardboard then overlays this data on the network device itself.

For instance, if we would like to visualize the network connection form computer A to computer B. This can be achieved by implementing the 2D/3D feature tracking module which can run on a mobile device. The next part would be to retrieve of network information from the device using various techniques followed by overlaying this information on google cardboard.

The android app works on android runtime environment or JDK. Within the linux kernel, the app might use functions like camera driver. This is one of the key features that Vista eyes will be needed as the visualization of network from one system to another with mobile camera. Another key feature used would be display driver. Communication between the systems will be prominent and can be visible when it will be reflected on to the mobile screen. Various other functions used from the kernel would be Wi-Fi driver to be the systems and mobile in same LAN network. Some other internal drives will also be used.

There are various libraries with android architecture. Some libraries that can be used explicitly are media framework. This can be used to maintain visual connection on the mobile device. Along with that many other core libraries will be used implicitly. Application framework is a part of architecture where the functioning will be established by activity manager, windows manager, view system, package manager available to make sure to have established framework before it goes to application. At application level, a set of core applications which includes maps or email well coded in Java programing.

Vista eyes will make us visible through augment reality if there is any kind of data transfer from one system to another system. We can visualize the transfer through google cardboard. Google cardboard has two 3D lens through which real time transfer can be seen. This is possible through the mobile phone's camera placed inside the google cardboard.

TECHNICAL DISCUSSION AND R&D

The Vista project is aimed at creating a new way to visualize computer network data, in real – time. I propose to develop a low – cost augmented reality visor system to view network data. This has many practical uses. Since network data and connections are an abstract phenomenon, it is difficult for students learning computer networks to grasp and imagine networks. With the help of Vista visors, it becomes easy to "see" the network data flow from system to system and thereby easily visualize and imagine computer networks. Apart from being of great help to students to initially begin network visualization, we envision that this will become an invaluable tool used by network administrators and network technicians to debug large and complex networks.

Let's break down how each aspect and see how the entire experience will work.

1. You download the Vista Eyes App on your phone/tablet.

2. Identify the network devices with the “Augmentable”.
3. Open the Vista Eyes App over the “augmentable” and experience the new reality.
4. Place your phone in google cardboard or your own VR headset.
5. Use the VR device to enter the virtual world. Experience the world like never before.

The project can be broadly classified into three parts. The first part is implementing the 2D/3D feature tracking module which can run on an android/IOS device. The second being the retrieval of network information from the device using various techniques and the third one being the overlay of this information on an oculus/google cardboard (VR devices).

State of Art:
Current tools used for network visualization are effective in showing packet data traces, but not much more. One can only say that the outputs produced by these tools are something that only people with experience can comprehend. Tools like ettercap, wireshark, traceroute, ping, nmap are examples of such tools. They are primarily command-line based with few and crude GUI systems.

Design:
The Vista visor system consists of 3 parts. The first being the target identification system, and the second one the data gathering sub-system and the third being the AR overlay system.

Target Identification System
The target identification system consists of identifying the system currently in view of the visor. This requires continuous scanning of the incoming video feed to detect text of image targets. Text detection forms the ideal target identification since textual data can be easily displayed on devices or screens. We can also use generated image targets to identify a system. There exist multiple libraries that can detect image targets. These libraries such as Qualcomm’s Vuforia and Apple’s MetaIO can be used to implement this sub-system. Each library has tools to generate image targets.

This sub-system gives the ability to detect the physical device and link it to its network address or network identity. This network address or identity can then be passed to the data and gathering sub-system.

The Data Gathering System
The data-gathering subsystem consists of a network monitoring software which can be queried for the statistics of the given device. This can be accomplished using the traditional network monitoring and visualization tools, such as ettercap, wireshark, traceroute, ping etc. We can also add query interfaces for software such as PRTG and MRTG or systems can be queried directly via SNMP, if SNMP services are running on these devices. Once this data is gathered, it can be analyzed for anomalies and a conclusive summary can be generated. This summary is then passed to the overlay sub-system.

The AR Overlay Sub-system
The AR overlay subsystem consists of the Google Cardboard. The google cardboard is a cheap VR device which utilizes an Android mobile device to generate VR scenes. The Vista eyes project

utilizes this VR device to overlay the generated network summary onto the tracked device from the target identification sub-system, thereby giving the user the ability to "see" network data as it flows through the network.

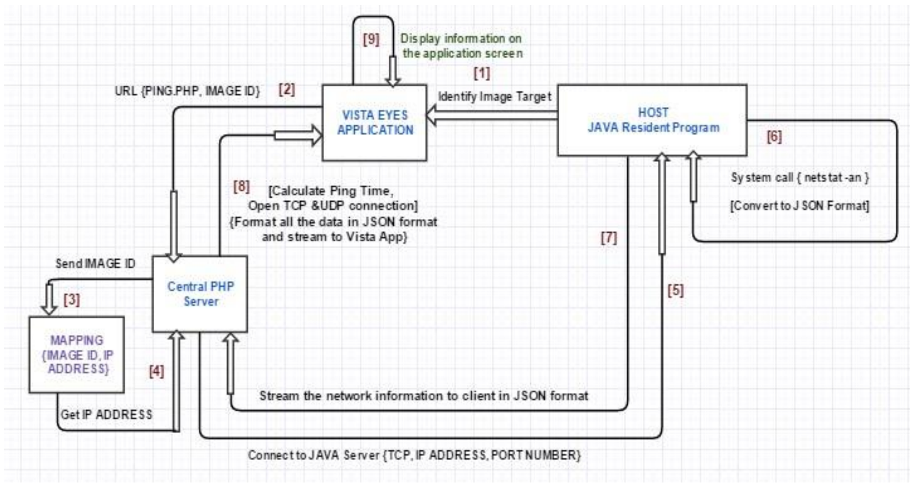

[9]
Display information on the application screen
[1]
Identify Image Target
URL {PING.PHP, IMAGE ID}
[2]
VISTA EYES APPLICATION
HOST JAVA Resident Program
[6]
System call { netstat -an }
[Convert to JSON Format]
[8]
[Calculate Ping Time, Open TCP &UDP connection] {Format all the data in JSON format and stream to Vista App}
[7]
Send IMAGE ID
[3]
Central PHP Server
[5]
MAPPING {IMAGE ID, IP ADDRESS}
[4]
Get IP ADDRESS
Stream the network information to client in JSON format
Connect to JAVA Server {TCP, IP ADDRESS, PORT NUMBER}

Figure 1: System design and simple flow diagram

In this we can see three main parts viz. Vista eyes android application, central PHP server and Java resident program. Three sections will be explained in detail.

Overall project can be explained in 9 steps which are mentioned in the figure 1 and are explained as below.

1. The user using the Vista eyes application can look at any host in the network, which has an image on it. Each host is associated with a pre-determined image. The application identifies the image in step 1.
2. Vista Eyes application will send the request or ping the central PHP server with parameters like URL address i.e. the IP address of the PHP server and image ID identified at step 1.
3. Central PHP server gets an image ID from the Vista Eyes app. This ID is searched in mapping.dat file. This file contains Image ID mapped with associated IP address of the host. This is stored in JSON format.
   Example:

```
{
        "blue": "172.20.10.7",
        "green": "172.20.10.5",
        "red": "172.20.10.8",
        "gray":"172.20.10.3"
}
```

Figure 2: Mapping.dat file

4. PHP server gets IP address from the mapping, data file.
5. The Java Server program runs on port 8888. PHP server open socket using TCP connection and tries to connect to host on port 8888. If the connection is established, then host is ready to service the client.
6. JAVA resident program executes the shell command "netstat -an" to get open network connections and listening ports on the host.
7. This information is packed in JSON format and sent back to the PHP server.

8. Once the information is received from the host, TCP connection to the Java resident server is closed. PHP server processes this raw data and calculates the network statistics like ping time, and number of open TCP and UDP connections. This information is wrapped in JSON format and streamed to vista eyes mobile application.
9. This information is then processed by the Vista Eyes application and it is finally rendered over the detected system.

Technical Innovation

The introduction of the Google Cardboard platform brings about affordable and easy to use VR for everyone with an Android device. Extending upon this cheap and effective platform, we introduce an augmented reality application which is the amalgamation of current network visualization techniques with upcoming computer vision technologies. This project also explores a new human-computer interaction technique, which holds much promise in the near future.

Required Development Tools
1. Android SDK and NDK
2. Unity – Game engine
3. Cardboard SDK for Unity
4. Cardboard SDK for Android
5. Android Studio
6. The system will be programmed in Java.

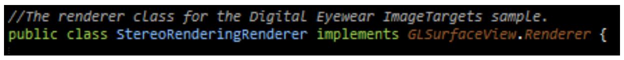
The original idea stemmed from the need to make complex computer networks easy to visualize and debug. The team got together to brainstorm and came up with the idea of overlaying network information over the devices themselves.

CHALLENGES

Modifying the StereoRendering Sample to Support Google Cardboard Handset

Since there are so many interacting components, a decent understanding of each of the module is required before we could modify the complex system.

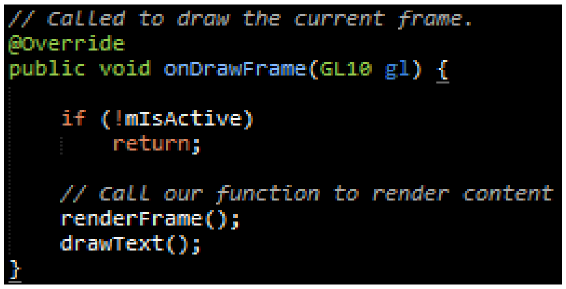
The vuforia renderer is implemented as a GLSurfaceView.Renderer to the android opengl es 2.0



```
//The renderer class for the Digital Eyewear ImageTargets sample.
public class StereoRenderingRenderer implements GLSurfaceView.Renderer {
```

Figure 3: GLSurface View

The setup of the rendering subsystem is as follows -
1. System calls the - onSurfaceCreated(GL10 gl, EGLConfig config) handler. This is where a new rendering (GL Surface is created and passed.) This is the entry point for initialization of the rendering.
2. The next method is void onSurfaceChanged(GL10 gl, int width, int height). This is called when the rendering canvas changes size and the new parameters are passed to the program. Once this is done, we are ready to render.
3. Beyond this point the void onDrawFrame(GL10 gl) is called for every frame to be rendered. Here we call into the vuforia frame rendering function - renderFrame()



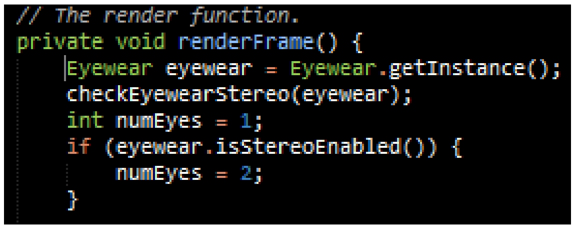
```
// Called to draw the current frame.
@Override
public void onDrawFrame(GL10 gl) {

    if (!mIsActive)
        return;

    // Call our function to render content
    renderFrame();
    drawText();
}
```

Figure 4: Render Frame

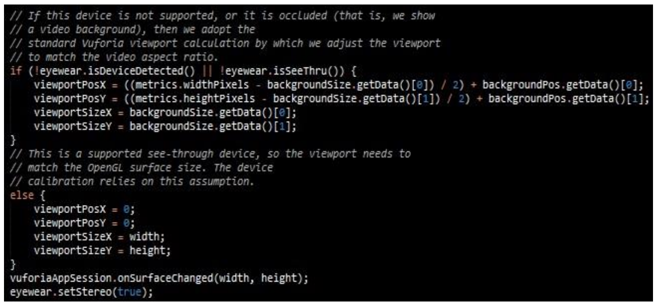
4. The renderFrame() is further divided into multiple sub- sections.
5. The first check done is to call into the checkEyewearStereo() which returns if the current headset is stereo capable or not. This is our first point of attack.



```
// The render function.
private void renderFrame() {
    Eyewear eyewear = Eyewear.getInstance();
    checkEyewearStereo(eyewear);
    int numEyes = 1;
    if (eyewear.isStereoEnabled()) {
        numEyes = 2;
    }
```

Figure 5: CheckEyewearStereo

6. Since the device is not supported by the library, it returns false since no device is detected.
7. We override this by manually setting eyewear to stereo in the onSurfaceChanged() callback.

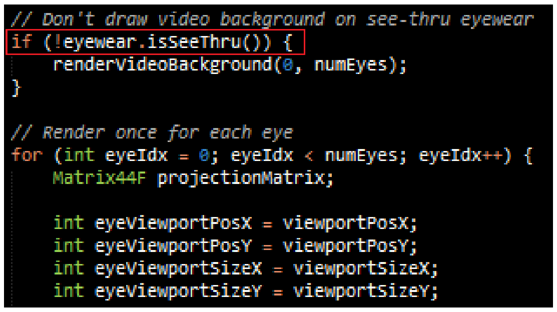


```
// If this device is not supported, or it is occluded (that is, we show
// a video background), then we adopt the
// standard Vuforia viewport calculation by which we adjust the viewport
// to match the video aspect ratio.
if (!eyewear.isDeviceDetected() || !eyewear.isSeeThru()) {
    viewportPosX = ((metrics.widthPixels - backgroundSize.getData()[0]) / 2) + backgroundPos.getData()[0];
    viewportPosY = ((metrics.heightPixels - backgroundSize.getData()[1]) / 2) + backgroundPos.getData()[1];
    viewportSizeX = backgroundSize.getData()[0];
    viewportSizeY = backgroundSize.getData()[1];
}
// This is a supported see-through device, so the viewport needs to
// match the OpenGL surface size. The device
// calibration relies on this assumption.
else {
    viewportPosX = 0;
    viewportPosY = 0;
    viewportSizeX = width;
    viewportSizeY = height;
}
vuforiaAppSession.onSurfaceChanged(width, height);
eyewear.setStereo(true);
```

Figure 6: onSurfaceChanged

The last line in the above figure shows this change. Then we override the video background rendering since the google cardboard is not a see-through device, but vuforia library cannot detect this.

This change is highlighted in the image.



```
// Don't draw video background on see-thru eyewear
if (!eyewear.isSeeThru()) {
    renderVideoBackground(0, numEyes);
}

// Render once for each eye
for (int eyeIdx = 0; eyeIdx < numEyes; eyeIdx++) {
    Matrix44F projectionMatrix;

    int eyeViewportPosX = viewportPosX;
    int eyeViewportPosY = viewportPosY;
    int eyeViewportSizeX = viewportSizeX;
    int eyeViewportSizeY = viewportSizeY;
```

Figure 7: Eyewear is DeviceDetected

This change is highlighted in the code fragment.

Finally, we now have support for the google cardboard on the vuforia stereo rendering program with image target tracking.

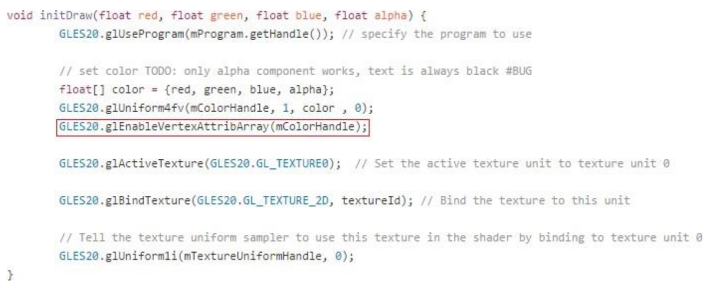
Rendering Text in OpenGL ES 2.0

"It is very likely that at one point you will want to draw text with OpenGL. It may seem like a very basic functionality of any drawing API, but you will not find any function in OpenGL that deals with text. There are good reasons for that: text is much more complicated than you think". This quote sums up, our initial take at rendering text, but we learnt that text - rendering on a programmable pipeline like OpenGL ES 2.0 is really hard.

This quote sums up, our initial take at rendering text, but we learnt that text - rendering on a programmable pipeline like OpenGL ES 2.0 is really hard. For this we used the procedure as shown in This blog by factitious. However, the article deals with rendering the text using OpenGL ES 1.0 which is a fixed function pipeline, whereas the Vuforia rendering example utilizes OpenGL ES 2.0 which is a programmable function pipeline.

Therefore, we used the code from a source called Texample2. However, the example 2 source code is focused on developing games, which generally have access to more resources than what is available on a mobile GPU, and would not function properly on the mobile. It also had a few bugs which we needed to fix. This caused us to spend some time making this work on top of the Vuforia renderer correctly.

The changes required to make it work with the vuforia renderer is as follows:
Disabling unnecessary glEnableVertexAttributeArray and glDisableVertexAttributeArray calls. These calls are a bug in the code and return gl errors which corrupt the drawing.



```
void initDraw(float red, float green, float blue, float alpha) {
        GLES20.glUseProgram(mProgram.getHandle()); // specify the program to use

        // set color TODO: only alpha component works, text is always black #BUG
        float[] color = {red, green, blue, alpha};
        GLES20.glUniform4fv(mColorHandle, 1, color , 0);
        GLES20.glEnableVertexAttribArray(mColorHandle);

        GLES20.glActiveTexture(GLES20.GL_TEXTURE0);  // Set the active texture unit to texture unit 0

        GLES20.glBindTexture(GLES20.GL_TEXTURE_2D, textureId); // Bind the texture to this unit

        // Tell the texture uniform sampler to use this texture in the shader by binding to texture unit 0
        GLES20.glUniform1i(mTextureUniformHandle, 0);
}
```

Figure 8: glDisableVertexAttribute

Here, mColorHandle is not an attribute array as per the shader program and does not need to be enabled or disabled.

Since there was not known solution to this problem, I had to post a question to stack overflow to look for an answer. But could not get one, and had to debug the entire module myself. The link to the question is here. (http://stackoverflow.com/questions/18495612/opengl-2-0- es-1281-error-on-glenablevertexattribarray)

Finally, there was an attribute value out of bounds, since the shader program was taking and array of 24 matrices, each being a 4*4 matrix of floats. The mobile GPU did not have this capability, which is why we needed to remove this array and use just 1 matrix.

Using a single matrix, solved the issues and we could render text with the vuforia renderer and example 2.

Drawing Lines

Drawing lines between image targets was not very difficult, but 2D lines do not look like flow lines, which is why we needed to remove them. The below image shows an example of 2D line between image targets.

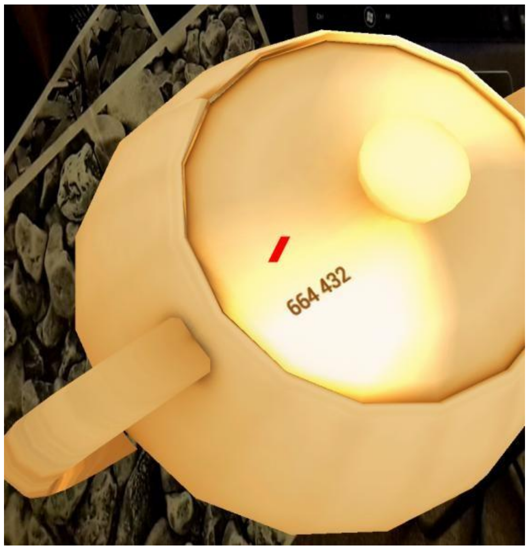


Figure 9: 2Dline

Secondly, to draw 3d lines the algorithms are very complex. Some of the methods are drawing 3D Bezier curves, Path controlled extrusion.Examination of these methods led to the conclusion that the algorithms are very complex to implementation for OpenGL ES 2.0 in the limited amount of time.

PROTOTYPE

In this section we have implemented the experiment where an image is tested to see if they are connected to same network. Along with that it shows what is the ping time and the connection ports.

Experiment Setup

To achieve the result of the product, we have modified the vuforia sample code. The image below shows through google cardboard when we render over an image, we see a 3D image of tea pot.

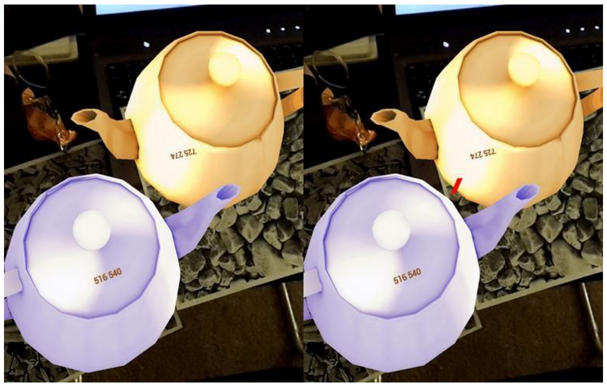


Figure 10: Initial Setup

The numbers denoted over the teapot are the coordinates which we have used in the modified code. Another image shows the similar image, with different coordinates.

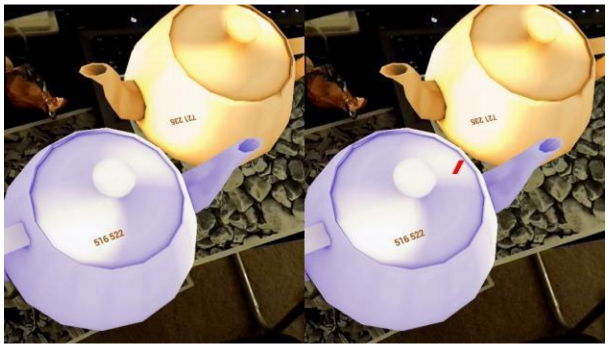


Figure 11: Initial set up with different prototype.

Lab Experiment

To execute the project aim, we have developed a lab experiment with 4 laptops. In which, 2 laptops are connected to same network and other 2 are not. When we hover the google cardboard with Vista Eyes application, we see the ping time and the port numbers connected to it if the laptop is on same network.

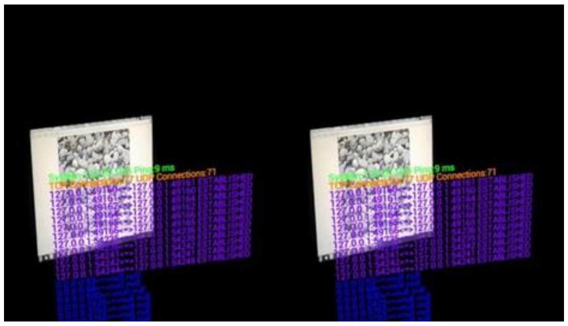


Figure 12: output over laptop1

We see the ping time and along with that we see the number of TCP and UDP connections. We see that the connection is established with port numbers established. If the system is not connected with same network, we see the error message.

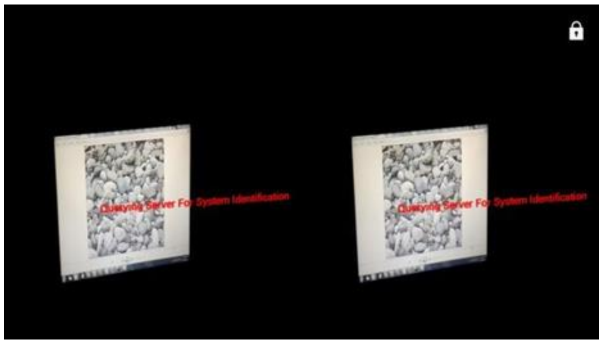

Querying Server For System Identification
Querying Server For System Identification

Figure 13: Querying for System Identification

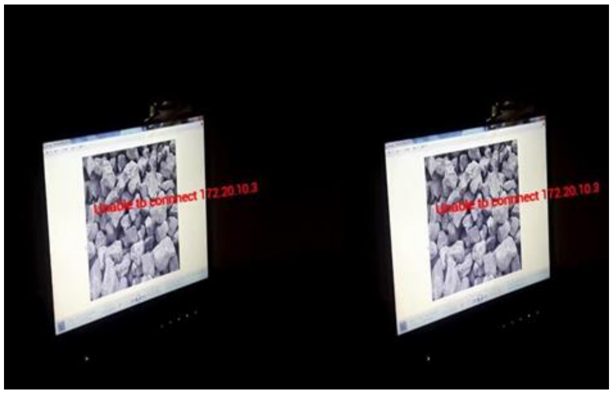


Figure 14: Error Message

The above figure shows if the system is not connected to same network we get error message. The 4th system is connected to same network, and we see the output.

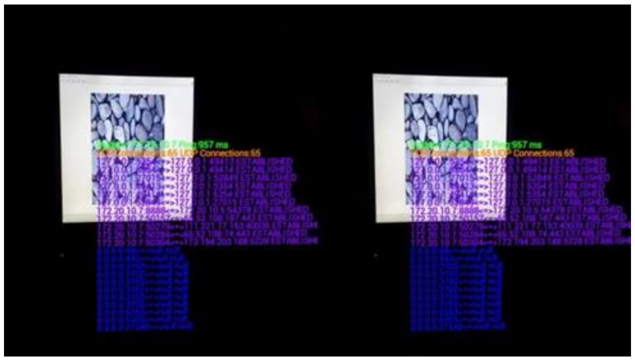


Figure 15: Connection with same network.

CONCLUSION

This project aims for talking a small step into new method of network debugging and visualization. This reduces the efforts to have peer review to run a software and analyze it. Vista Eyes gives modern and mobile way to see a network establishment.

References

[1]. Installing Android studio, Android Developers, Tools, 2015

[2]. "Create and Connect with Unity 5", Unity, 2015

[3]. "Cardboard SDK for Unity", Google Developers, May 28, 2015

[4]. "Cardboard SDK for Android", Google Developers, May 26, 2015.

[5]. "Virtual Reality", Wikipedia, December 4, 2015 [6]. "Google Cardboard", Google

[7]. "Hands-On with Google Cardboard Virtual Reality Kit", Google, June 30, 2014

[8]. "Augmented Reality", Wikipedia, December 4, 2015. [9]. "How to Use the Trackable Base Class ", Vuforia Developer Library, 2011

[10]. "PHP Socket Programming, done the right way", Christoph Hochstrasser, July 24, 2012

[11]. "Writing the Server Side of a Socket", Java Tutorials, 2015.

[12]. "Reading from and Writing to a Socket", Java Tutorials, 2015.

[13]. "Netstat", TechNet, Library, Microsoft, 2015.

[14]. "Command line reference A-Z, Library, Microsoft, 2015.

[15]. "google/gson", GitHub, Google Inc., 2008.

[16]. "Configure a PHP Website on IIS", IIS, Keith Newman and Robert McMurray, April 14, 2013.

[17]. "OpenGL Programming/Modern OpenGL Tutorial Text Rendering 01", Wikibooks, January 31, 2015.

Glossary

[1] Augmented Reality is direct or indirect view of a physical, real-world environment whose elements are augmented or supplemented by computer generated sensors input such as sound, video and graphics or GPS data.

[2] Virtual Reality can be referred as computer- simulated reality, replicates an environment that simulates a physical presence in places in the real world or imaginary world, allowing users to interact in that world. VR creates artificially creates sensory experiences, which can include sight, hearing, touch and smell.

[3] Vuforia is an Augmented Reality Software Development Kit (SDK) for mobile devices that enables the creation of Augmented Reality applications.

[4] Ping response is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network and to measure the round-trip time for messages sent from the originating host to a destination computer and back.

[5] SNMP Simple Network Management Protocol is an Internet Standard Protocol to manage devices on IP Networks.

[6] Network Socket is an endpoint of an inter-process communication across a computer network.

[7] Routing is selecting the best path in a network.

[8] Google Cardboard is a Virtual Reality platform developed by google for use with a fold-out cardboard mount for a mobile device.

[9] SDK is typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development.