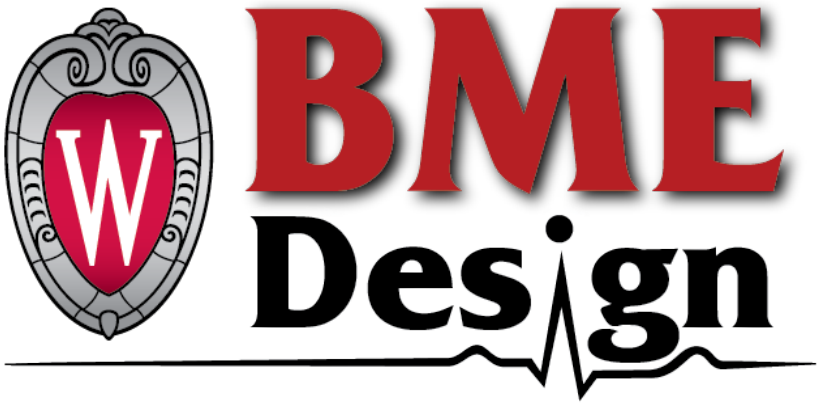

IPHONE VIRTUAL REALITY TRAINING MODEL FOR MICROSURGICAL PRACTICE

Preliminary Report | Oct 6th, 2020

Client: Dr. Ellen Shaffrey

Advisor: Dr. Willis Tompkins

Course: BME Design 400

Team Name: iPhone VR

Team Members: Jason Wang - Team Leader

Jiong Chen - BWIG

Martin Janisewski - BSAC & BPAG

Xiaoxuan Ren - Communicator

Abstract

Medical training technology has been growing as complex models and simulations have made it cheaper and easier to train medical students and professionals outside of real procedure observation and practice. The world at large can benefit from cheaper training technology as developing countries have a demand for increased medical training but lack the wealth to obtain expensive medical equipment. One area of improvement is microsurgery, which commonly requires expensive equipment for doctors to practice and perform surgery. Products on the market such as the eoMicro allow surgeons to practice at home with little equipment needed for around $100; however, issues of streaming delay and depth perception persist. Our client has plans for developing and distributing an inexpensive microsurgery training "kit" using minimal supplementary equipment and primarily consisting of commonly owned items such as smartphones and laptops. Magnification from the smartphone cameras provide a view of the microsurgery training area while the footage is streamed on a screen for comfortable viewing while practicing. What remains to be solved is a reduction in latency between video streaming and camera recording to improve operator precision, as well as a need to develop video depth perception potentially using 3D for optimal surgical performance. Several hardware configurations and video modifying software are being evaluated to overcome these issues.

I. Introduction

1.1 Motivation

The surgical microscope is the most expensive and space-occupying device used in microsurgery. In developing countries, microscopes are not always available and used in operating rooms where they are needed most [1] . Lack of resources and trained personnel in low and middle income countries lead to inaccurate diagnosis and inadequate treatment [2] . Affordable and effective surgical microscopes have the potential to significantly improve disease detection rate in undeveloped countries where diagnostic laboratories are scarce [1]. Inexpensive home microsurgical equipment allows resident surgeons to have more opportunities practicing and improving their skills.

1.2 Existing Devices

1.2.1 eoMicro simulator

Figure 1: eoMicro simulator. The eoMicro simulator in combination with a smartphone creates a home microsurgical set-up that allows microsurgical skill improvement [3]

The eoMicro simulator is a low-cost and portable home microsurgical set-up. It consists of a portable flat-pack platform which allows the trainee to use it in combination with a smartphone to practice microsurgical procedures and improve his or her microsurgical skills. But this model

fails in truly simulating a surgical microscope which magnifies the three-dimensional operation. The trainee cannot have real-time feedback and correction from their mentors about their practice.

1.2.2 3D viewing system

Figure 2: Integrated video camera in observer scope streams the real-time video to the 3D monitor Surgeons are using a 3D surgical viewing system that displays the procedure on a 3D monitor in the operating room so that the entire operation can be observed by the team. [4]

1.2.3 Client Current Model

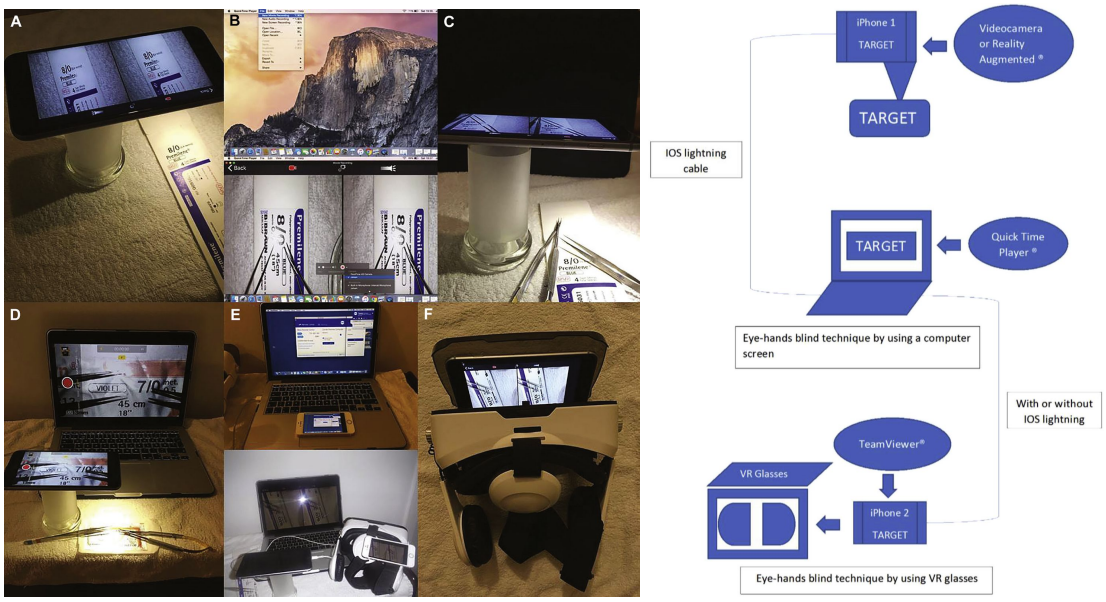

Figure 3: Setup of the training model and schematic illustration of the equipment. iPhone(#1) is connected to a MacBook and is placed on a holder to record the surgical target field. This iPhone(#1) is used as the webcam of the computer. The video is duplicated by Reality Augmented (a computer specific software) and transferred from the computer to iPhone(#2) which is connected to virtual reality glasses. [5]

This model is designed for any medical student, resident, fellow, or surgeon with almost no-extra cost, as it is based on “technological tools” that are already mostly owned by any trainee (iPhones or other smartphones, personal computers). The first iPhone is placed on a holder at a height 15 to 20 cm from the surgical target field and connected to the MacBook to work as the webcam. The video is duplicated by Reality Augmented (a computer specific software) and transferred from the computer to the second iPhone which is connected to virtual reality glasses.The use of virtual reality glasses creates a stereoscopic 3D image by angling two 2D images creating an illusion of depth to the image. This setup is able to provide comparable magnification and adequate depth perception to truly simulate a surgical microscope. But the latency of video streaming remains and the VR glasses have some limitations in offering high-resolution images.

1.3 Problem Statement

Microsurgical training is relevant across several surgical specialties, including neurosurgery, vascular surgery, otolaryngology, ophthalmology, and plastic surgery[3,6,7]. It is technically challenging to master manual skills because of the complexity of microsurgery, such as the use of fine instruments (with a tip precision of 0.025 mm) and handling delicate tissues [5]. Therefore, a continuous training process and detailed feedback on skills components are required for microsurgical residents to refine precise technical skills, develop eye-hand coordination, and achieve a high level of dexterity [8].

Effective training to develop microsurgical skills requires at least 2×–5× magnification of the surgical field, but the surgical microscopes are expensive[9] and well-equipped microsurgical labs are only available in advanced countries. As a result, many residents have little or no opportunities to learn and practice microsurgical skills due to lack of high-cost equipment.[5, 10]. Thanks to modern technology, the camera capabilities of an iPhone provide such incredible magnification that they are comparable to surgical microscopes[10, 11]. An iPhone with 8 or more megapixel camera is able to record videos from 720p HD at 30fps (frames per second) to 4K at 60 fps. Although iPhones alone are not able to simulate a surgical microscope due to its lack of stereoscopic view, the application of VR glasses displays the procedure on a 3D view and enables more accurate operation[10].

The client has attempted to create this model using a computer, two cell phones, VR glasses and lightning cable connection, However, there is a delay that is between 0.5 to 1 second long from this current mode, so a simple streamlined iPhone-VR system is proposed to create a home microsurgery simulation tool for resident surgeons to practice skills.

II. Background

2.1 Client Information

Our client is Dr. Ellen Shaffrey, a plastic surgery resident in Dr. Samuel Poore’s plastic surgery lab at the Department of Surgery in the University of Wisconsin School of Medicine and Public

Health. Dr. Shaffrey made a proposal for a series of devices to simulate a microsurgery training environment at one's home, named "iPhone virtual reality training model for microsurgical practice". Despite having virtual reality in the name, the real function of the device is to establish depth perception for its user when they perform microsurgery training. The client had also developed 3 prototypes of her own to experiment with different setups to achieve a functioning training environment. The first: an iPhone on a stand connected to MacBook. The second: a VR App on an iPhone creating two images. The third: an iPhone on a stand connected to a MacBook with Google VR cardboard glasses. Each of these had limitations, however. Respectively, the QuickTime Player, which is a multimedia player, contains a live screen mirroring feature that does not rotate from portrait view to landscape view and there was a slight time delay, poor zoom, and 0.5-1sec delay when no cable and cannot modify the iPhone when wearing the glasses. The BME team will proceed with these prototypes in mind to develop an effective system with low latency and video footage with depth perception.

2.2 Magnification of Microscope and iPhone

The professional microscope that the client uses for microsurgery is the Mitaka MM51, which is capable of up to 42x magnification while using a 4K camera and monitor. The level of zoom is 8:1 [12]. The client is testing options of the iPhone 8 and iPhone XR which have 4K video recording at 24/30/60fps and 2x optical, 10x digital zoom and 4K video recording at 24/30/60fps and 3x digital zoom, respectively [13] [14]. In typical microsurgery, the magnification used by our clients ranged from 2-5x and 15-25x. In this case, the iPhone 8 meets the minimum requirements for the magnification required for some microsurgery. The iPhones' screens are not optimal for viewing, and instead a separate screen is expected to stream their video footage.

Figure 4: The entire Mitaka MM51 system

Figure 5: The iPhone 8 and iPhone XR and their cameras

2.3 Depth Perception

Depth perception is an important feature for the device as it allows the user to properly judge the distance their hands are from the camera and perform the necessary delicate actions of surgery. Depth perception can be considered seeing objects in 3D which is one of the ultimate goals for the device. Vision can be differentiated between monocular and binocular vision, where binocular vision is the feature that allows for depth perception. Binocular vision is achieved by having two similar views of the same perspective with one view being slightly offset from the other. Humans have two eyes which can achieve this, and the two images are fused into one in the brain with the ability to infer the relative distance between objects.

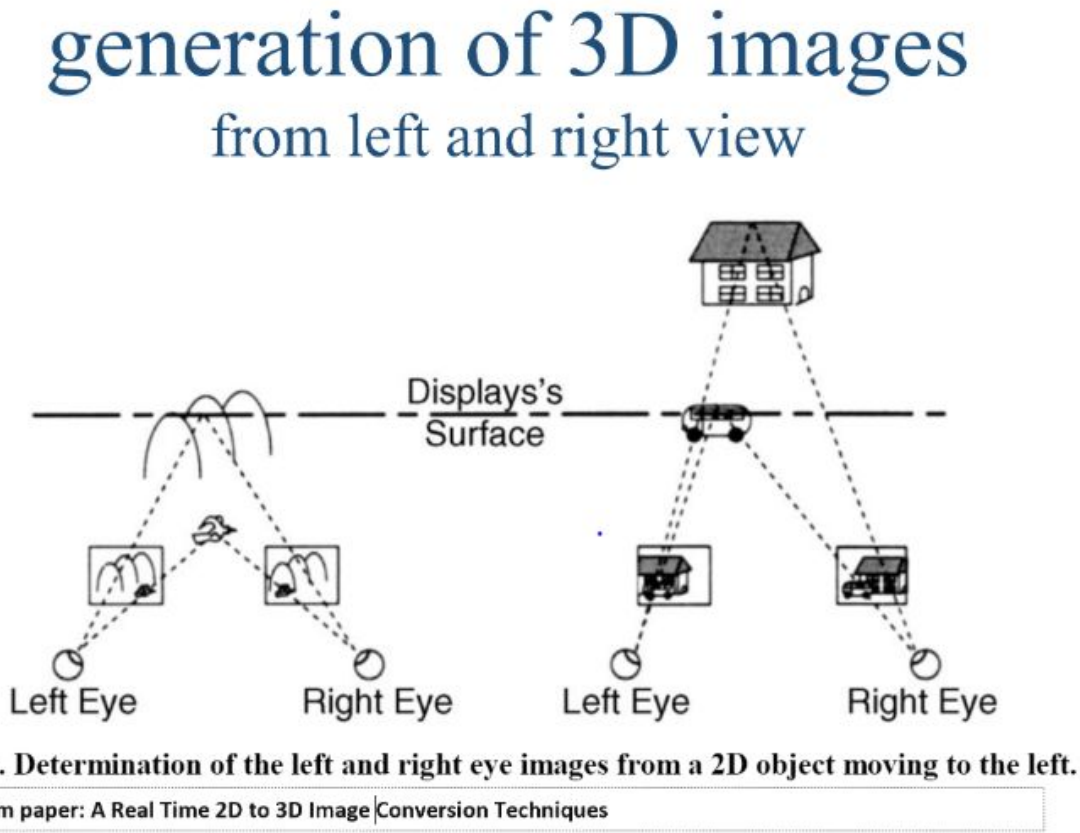

Fig.2. Determination of the left and right eye images from a 2D object moving to the left.

Cite from paper: A Real Time 2D to 3D Image Conversion Techniques

Figure 6: An example of how 3D is achieved using offset images from two different viewpoints

Cameras with a single lens can only achieve monocular vision, so there are issues with hand-eye coordination if you rely on footage from a camera to control your hands. This is the case when using a smartphone for magnification. However, there are ways to digitally create a 3D perspective when using a single lens. One strategy is through the use of Depth Image-Based Rendering (DIBR) and algorithms, which have multiple approaches. The main principle is that

the depth map of a stream is generated through values saved for each pixel that represent how far the pixel is from the camera. First, the current texture image and a stationary scene image, which is extracted from the input video, are warped to the same virtual perspective position by the DIBR method. Then, the two virtual images are merged together to reduce the hole regions and maintain the temporal consistency of these areas. Finally, an oriented exemplar-based inpainting method is utilized to eliminate the remaining holes [13].

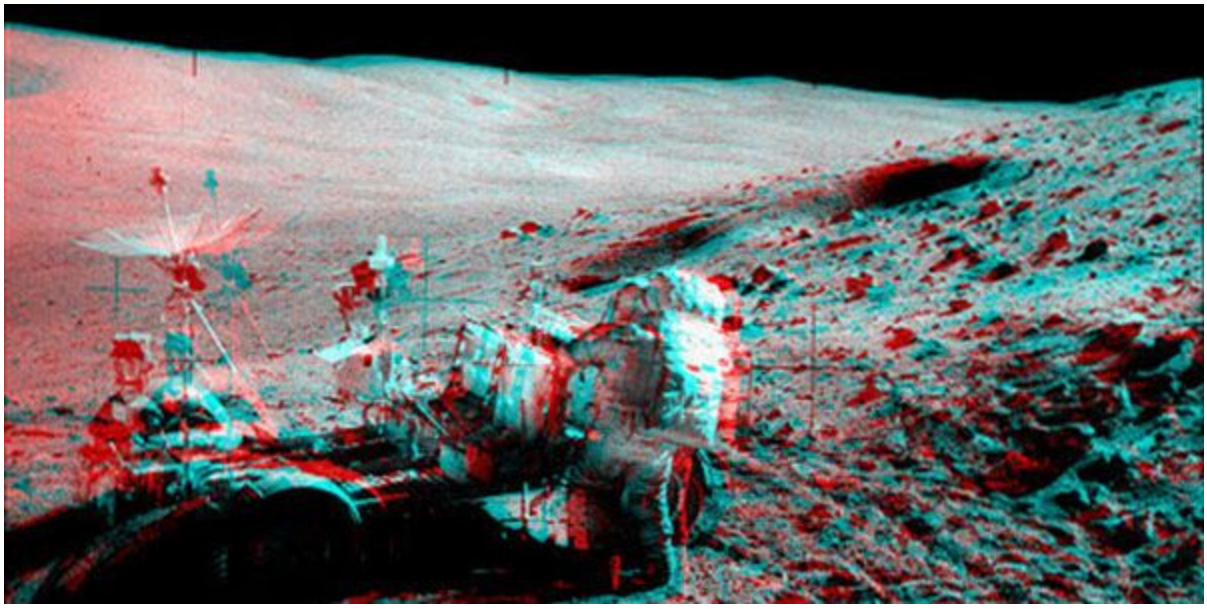

Figure 7: An anaglyph image showing the cyan and red images offset from each other

A 3D effect can also be achieved using anaglyphs, which is the technology used for blue-red 3D glasses. Anaglyph images are used to provide a stereoscopic 3D effect, when viewed with glasses where the two lenses are different (usually chromatically opposite) colors, such as red and cyan. Images are made up of two color layers, superimposed, but offset with respect to each other to produce a depth effect. Usually the main subject is in the center, while the foreground and background are shifted laterally in opposite directions. The picture contains two differently filtered colored images, one for each eye. When viewed through the 'color coded' 'anaglyph glasses', they reveal an integrated stereoscopic image. The visual cortex of the brain fuses this into perception of a three dimensional scene or composition [15]. In the end, these effects are achieved through digital filtering with minor analog components to complete them.

2.4 Design Specifications

The client has already developed prototypes of their own and has determined that for a camera 4x/5x/6x magnification and up to 15x/20x/25x magnification is necessary for viewing smaller blood vessels. This magnification needs to be adjustable and not fixed for user convenience. The range of field for the camera is not large as it is close to what it is viewing, and peripheral vision is not necessary as the focus is on a small feature. The main focus for the current project is to lower the delay time between devices and create a sense of depth perception for the viewer. A secondary feature for the device is that the setup should be rather affordable, preferably no more than $100 for the cost of items outside of smartphones and laptops, to be appealing compared to competing products on the market. The device should take into mind any motion sickness a user can experience while using it as developing an artificial sense of depth may be headache inducing.

III. Preliminary Designs

3.1 3D Glasses Model

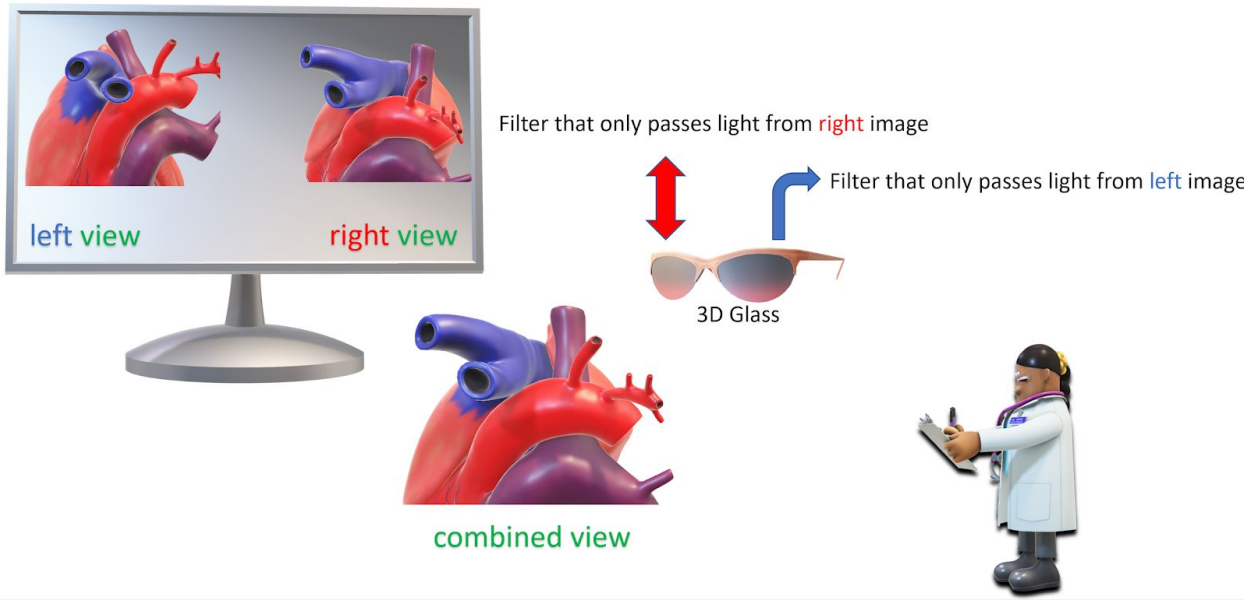

Figure 8. The Concept of 3D Glasses Model. The 3D Glasses Model consists of active 3D glasses with a big screen displaying 3D compatible images.

The first model is the 3D glasses model. As the name suggests, the 3D glasses model uses active 3D glasses with a screen casting the image from the camera. We plan to use software methods such as anaglyph or hardware methods like a 2D to 3D convertor to process the image. While

practicing surgeries, the trainee will wear 3D glasses and look at a screen in front of them. This might be a bit different from traditional surgeries where the surgeon looks down to see their hands. The 3D glasses are much lighter than the VR goggles and thus place less stress on the nasal bone. In addition, since the 3D glasses do not have a screen directly in front of the user's eyes, it is less likely to cause potential vision discomfort during usage.

3.2 VR Goggle Model

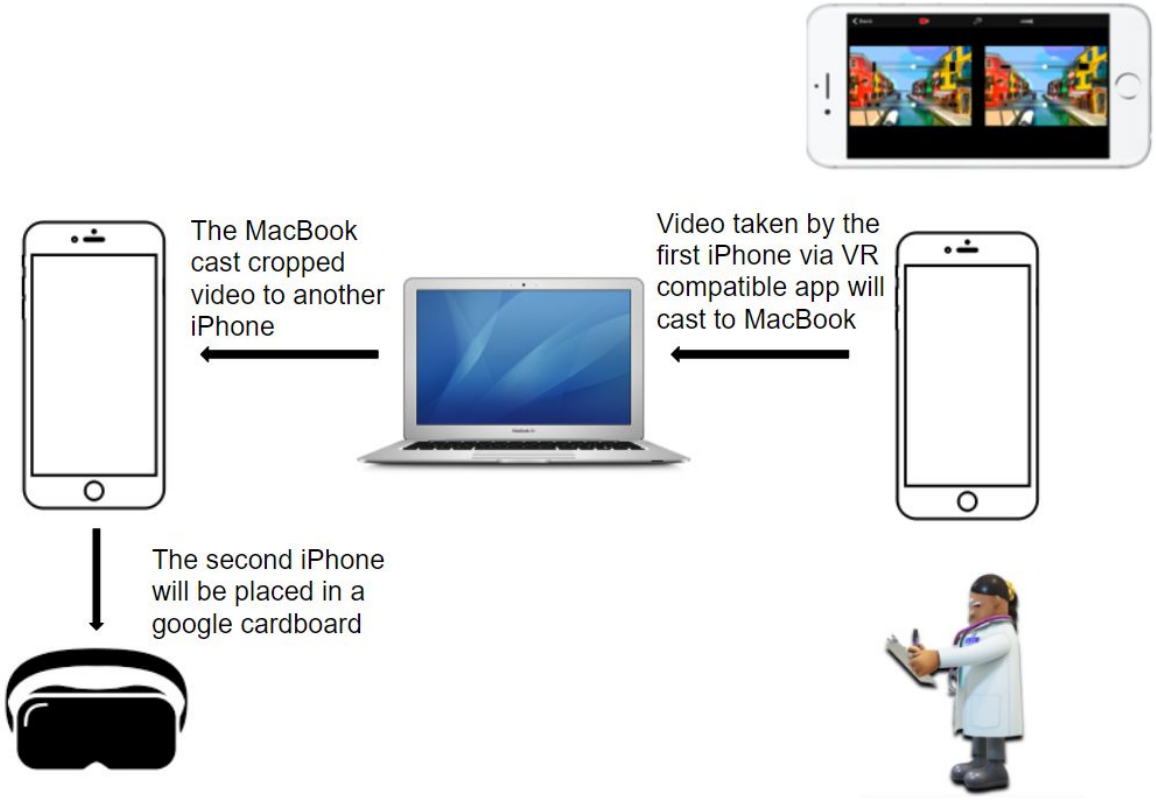

Figure 9. The Concept of VR Goggle Model. The VR Goggle Model consists of two iPhones, a MacBook, and a VR set. One of the iPhone in this model is for capturing the image of the surgery, and the other iPhone is for displaying the processed image using VR goggles.

The second model is the VR goggle model. This model uses a VR headset instead of the 3D glasses. The other parts of this model include two iPhones and one MacBook. One of the iPhones will serve as the camera for the platform. This iPhone will use an App from the Apple Store called Reality Augmented. This application will split the camera image into two so that it will be viewable with a VR headset (as shown in the upper right). Then the split image will be casted to a MacBook, and the MacBook will process the image to crop the margin of the iPhone screen.

Finally, another iPhone will receive the cropped image and display the image via the VR headset such as Google cardboard.

This setup will allow us to have better resolution using the iPhone's camera and better vision effect with the VR technology. However, the data transferring among the three devices might cause a noticeable delay for moving objects. In addition, the application Reality Augmented does not have the function of zooming so that the focal distance between the camera and the object will be fixed.

3.3 WebCam VR Model

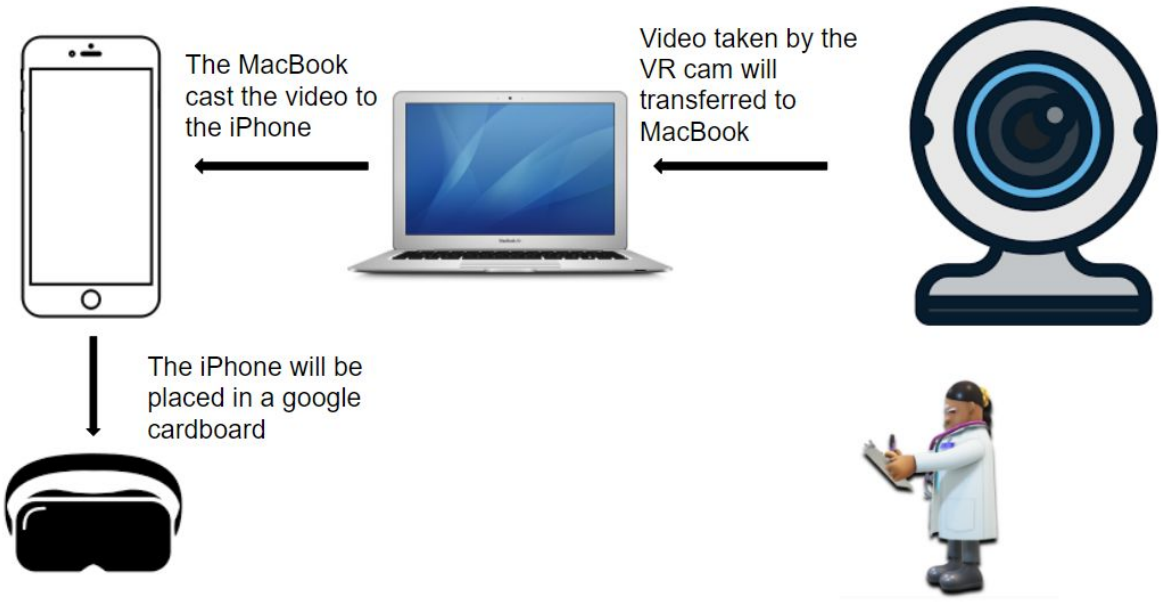

Figure 10. The Concept of WebCam VR Model. The WebCam VR Model has a very similar setup as the VR goggle model. The change we made for this model is that we replaced the first iPhone which serves as the camera with an actual VR camera.

The third model is the WebCam VR model. We used a VR webcam instead of an iPhone in this model to achieve a better VR compatible experience and zooming ability. There are already commercial VR cameras for streaming purposes. In this case, the camera will be able to display the VR compatible image directly on laptops and the delay between devices will be minimized. We anticipate that by using a webcam, it will be much easier to set up the platform and have good quality of vision as well. After the image is transferred to the MacBook, the rest of the steps will be pretty much similar to the VR goggles model; the macbook will cast the image to an

iPhone and then the iPhone will be placed in VR goggles. In this case, the user will be able to look at the object with the zoom they desire.

IV. Preliminary Design Evaluation

4.1 Design Matrix

	3D Glasses		VR Goggles		Webcam VR	
Efficiency (30)	5/5	30	2/5	12	4/5	24
Complexity (25)	4/5	20	2/5	10	2/5	10
Feasibility (20)	3/5	12	4/5	16	4/5	16
Quality (15)	3/5	9	3/5	9	5/5	15
Cost (5)	5/5	5	3/5	3	4/5	4
Safety(5)	4/5	4	3/5	3	3/5	3
Total (100)	80		53		72	

Table 1. The Design Matrix of Three Models and Evaluation Results.

4.2 Design Consideration

4.2.1 Efficiency

The first criteria we evaluated is the efficiency of the device. We analyzed the speed of the connection between each device for the three models. In other words, we estimated how fast the image transaction and conversion compared among those models, and the potential delay it may cause. The 3D glasses model got the highest score, because the other two VR models all require data transaction from a camera to a MacBook, and then from the MacBook to an iPhone. However, the 3D glasses model only needs a camera and a large screen to display the image. Therefore, we anticipate that the data transaction in the 3D glasses model will be fastest among the three models.

4.2.2 Complexity

The complexity mainly looks for the number of devices that will be incorporated into the overall design. It is clear from the setup of the platform that the 3D glasses model has the fewest devices required, a camera and a TV or monitor. Both of the models that use a VR headset require at least three devices including an iPhone, a MacBook, and a webcam or an additional iPhone.

4.2.3 Feasibility

In order to evaluate the feasibility of the three models, we asked the question if we can make the device using the skills we have learned and the technology available to us. For the VR goggles model and the WebCam VR model, we only have to correctly connect each device and find the best application for camera and casting. Nonetheless, as for the 3D glasses model, we probably will have to code for the conversion between the 2D and 3D image. The concept of anaglyph is not hard to understand, but compared to the both VR models, the score of the 3D glasses model will be relatively lower.

4.2.4 Quality

The criteria of quality examines the quality of the image for the user. We believe that the VR webcam will have the highest quality of image because the camera is specifically designed for VR videos. Even though the iPhone also has great cameras, the quality of the image may be lost during the transaction and process through the application.

4.2.5 Cost

We calculated the cost of each model and removed the device that people normally have. We expect every surgeon will have a smartphone and a laptop, and likely will have a TV at home. In this case, the cost of an additional smartphone will be the largest, while the cost for a camera in the 3D glasses model will be the cheapest.

4.2.6 Safety

As for the safety factor, we considered motion sickness caused during usage and wearing comfortableness. Since the VR headset will have a screen close to the user's eyes, the motion of the user or object may cause some discomfort for the user such as dazzling. Because the 3D glasses are light and are designed to look at the screen far from the user, the risk of having vision discomfort will be much lower. In addition, usually the head strap for the VR headset is very tight and long term usage of the VR set may cause pain to the user. Overall, the 3D glasses model got the highest score for the safety criteria.

V. Fabrication/Development Process

5.1 Materials:

5.1.1 Software Development

Since developing the application for the iPhone is basically coding on the computer. The material we need for this part contains a MacBook with XCode environment (a 15-inch MacBook Pro 2018, 2.6 GHz 6-Core Intel Core i7, 16 GB 2400 MHz DDR4 is used in this project), an iPhone that is capable of connecting to the MacBook for installation of the application (an iPhone 12 Pro 256 GB is used in this project), and an Apple type-c to lightning cable.

5.1.2 Anaglyph

The current iteration of the anaglyph code runs on MATLAB, a programming language developed by MathWorks often used for mathematical calculations but can also be adapted to work with other programming languages. The hardware used for testing and developing this program is a 64-bit Windows 7 computer with an Intel(R) Core(™) i7-4770 CPU @ 3.40 GHz with 16.0 GB of ram.

5.2 Methods

5.2.1 Software Development

The code for displaying the camera view is written in ViewController.swift in the project. As shown in the figure x, when the app is opened and permission of the camera is granted, the method viewDidLoad() calls all the other functions.

First of all, the zoom function is added to the app by using the UIPinchGestureRecognizer [17]. By passing this parameter into the camera, the user is allowed to magnify or reduce the image in the display by pinching with fingers on the screen. After the zoom recognizer is added, we then add the camera feed to the preview layer of the user interface. Finally, the camera feed will be displayed with the self.captureSession.startRunning() command [18].

In order to process the video in real time, we used two functions, which is the addVideoOutput() and captureVideoOutput(). In this part, the camera feed is captured as RGB frames and output to the application. In this case, we will be able to combine the anaglyph algorithm into the application and process the image frame by frame to combine and generate a 3D glass compatible live video.

5.2.2 Anaglyph

The code for each function of the program was written in iterations until each one has fulfilled its purpose. Research was done on using MATLAB's built in commands and example work was analyzed to understand what was needed in order to achieve each step required for anaglyph video conversion. The process for developing the code will be explained in chronological order:

Generating an anaglyph image using software:
Due to the team's familiarity with the MATLAB scripting program, it was chosen as the basis for the program's development. However, the means to process images and video was not a familiar concept, so the first step to develop the code was the most basic one.

First, the ability to process an image into anaglyph was researched. The product of this research is "anaglyph.m", recorded in **Appendix B**. This script does three simple things: Read input images and values from the function input, apply the user's choice of filter to both images, and form both images into one image.

Second, it is necessary to be able to process singular images to have an anaglyph effect, so MATLAB commands for image processing were researched and the product of this is "one_image_anaglyph.m" recorded in **Appendix C**. This script copies a single image, rotates two versions of it in 5 degrees at opposite directions, crops the non-overlapping edges, saves

them in the short term and sends them to “anaglyph.m” with parameters for final single anaglyph image assembly.

Lastly, a script incorporating the use of both previous scripts as functions was developed from an example program for processing video files into frames. “extract_movies_into_frames.m” is the main script for the video anaglyph conversion program, and it can be seen with a lot of silent code in **Appendix D**. The code loads a video, stores some of the information it contains, and processes each individual frame into “one_image_anaglyph.m”, which calls the function “anaglyph.m”. Each frame is displayed as it is processed, displaying a video as fast as the software can process it.

This script still uses a lot of its framework for reference and it will be gutted in the next iteration of “extract_movies_into_frames.m” for a more specialized and faster working script.

The current iteration of the anaglyph conversion software is written in MATLAB and has been and will continue to be tested in MATLAB until it is determined that the software can process high quality video in real-time using appropriate software. There are ways to process MATLAB code into Xcode to be able to run on Apple devices, however, the process is complicated and will only be attempted once final testing is completed [21][22].

5.3 Final Prototype

5.3.1 Software Development

The final product of the software Camera Access meets all the expectations from our design requirement. Here we have some figures about the software. The left of figure x shows the icon of the software, along with its name “Camera Access”. The right of figure x is the user interface of the application. Once the camera permission is accessed, there will be a green dot near the battery icon on the top bar to show that the camera is working. As shown in the image, to have a clear image displayed, we only added the camera view without any other distractions, such as the tape bottom the built-in camera has. There are two white edges near the top and bottom part of the application, which can be cropped by either using a non-full screen model of iPhone or through software on the MacBook to have full view of the image. There are no notifications or scalers with the zooming function of the application, only the pure camera view will show on the display. As the user pinch on the screen, the camera view will zoom accordingly with the finger and the focus of the camera will be automatically adjusted.

Figure 11. The icon of the Camera Access application and the user interface of the application. The app shows as a regular application on the iPhone with a blank icon. The user interface contains mainly the camera view with two white edges lying on the top and bottom part of the screen.

5.3.2 Anaglyph

The most recent iteration of the anaglyph software does not yet meet the requirements specified by the client. The software currently only runs on the MATLAB scripting program, which is written in the C/C++ language, and has been tested using computers running Windows 7. Xcode, the software Apple devices use, is written in a programming language called Swift. However, Xcode supports the C/C++ language, meaning it is possible to adapt MATLAB code to run on Apple devices.

The current target video footage to be processed is 1080p 30 fps iPhone video footage. With the current software and testing hardware, this is achieved at a rate of ~2.4 fps, which is still far from being processed in real-time. The ideal footage would be processed in real-time at 60 fps at 4000k resolution, which will most likely not be feasible due to the limitations of available and accessible hardware.

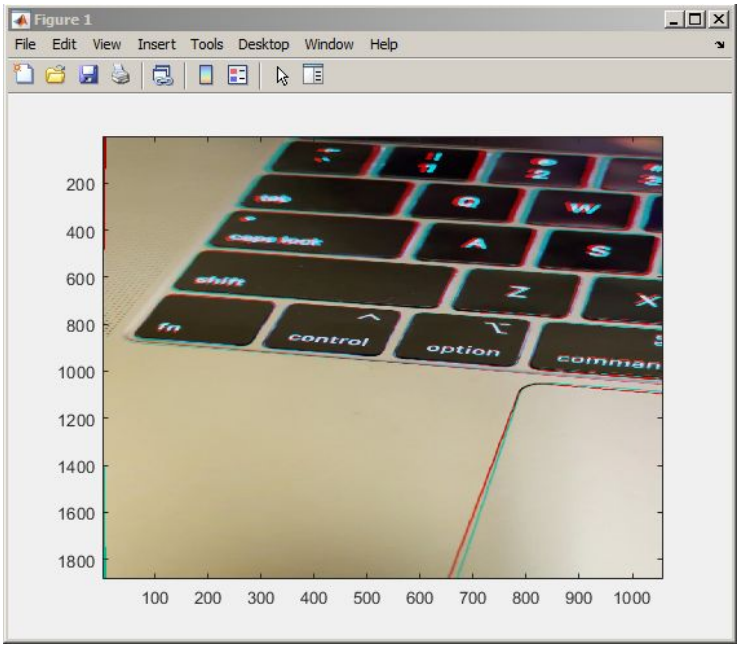

Figure 12: iPhone Camera Footage Being Processed in Matlab

While the image quality is high and the anaglyph effect is noticed, final image quality needs to be refined and video processing speed needs to be improved to match original video play speed

5.4 Testing

5.4.1 Software Development

The function of the camera feed and zooming is tested by directly installing the application on the iPhone. As shown in the figures in the final prototype section, the application achieved the expected function. In order to test if the application is actually outputting each frame while we open the application, we added a few lines of code in the captureVideoOutput() method. With the captureVideoOutput() method, once the application is running and the software is outputting the image, we will have an output message appear in the console of the XCode, and the outputting message is “did receive image frame”.

5.4.2 Anaglyph

As the anaglyph video conversion was developed, its various components were tested to ensure the basic operations that make up the script work independently of one another.

The first aspect that was tested was the script’s ability to convert a set of images into anaglyph.

Figure 13: Anaglyph Proof of Concept

The script “anaglyph.m”, as explained in the methods section of the report, being tested for functionality. A clearly anaglyph effect can be observed from the image.

Using the created script “anaglyph.m”, two images from different angles captured by a Samsung Galaxy S8 camera were merged and filtered using anaglyph colors cyan and magenta.

While this specific example isn’t of high quality, it proves that the team is capable of developing a program that makes use of anaglyph in a scripting software.

After this development, the criteria for the requirement of single image anaglyph conversion was addressed. Using commands to duplicate and modify the same image in MATLAB, it was easier to manipulate the extent of the anaglyph effect in an image.

Figure 14: Anaglyph Effect on Single Image

Proper depth perception is achieved using two different views of the same object. However, with appropriate image editing, an anaglyph effect can be achieved using a single view and image of an object.

It was observed in testing that this achievement came at a small cost: for a 434x342 pixel image, 5.83% of the image needed to be cropped out to focus only on the part of the image that had contained both traits of the shaded images. The process of rotating images and merging them meant that the edges of the final image did not have proper overlap and would not provide depth perception when using appropriate 3D glasses. The script was further modified to include a cropping of the image exclusive to the resolution and aspect ratio to produce a slightly smaller but consistent anaglyph image.

Figure 15: The 434x342 Image Unedited

The basic single image anaglyph conversion uses an image rotation that results in regions of no overlap at the borders of the final image. Because there is no overlap, the anaglyph effect breaks in these regions and produces an overall lower quality anaglyph effect for the image.

Figure 16: Close up of Figure X's Region of No Overlap (above)

Figure 17: The 434x342 Pixel Image Cropped to 421x332 Pixels

The image is kept at the same aspect ratio but made smaller to remove the borders where the one image anaglyph conversion is imperfect. A slightly smaller image also results in a slight increase in image and video processing rate proportional to the image size decrease

While the degree of depth perception to viewers remains untested, the means to modify this effect very easily was achieved. The final testing for the program in this state is the perpetual optimization of video processing script “extract_movie_into_frames.m” which calls upon the other two scripts to produce anaglyph video.

The main indicator for real-time processing video is program processing framerate identical to the original video’s frame rate. While it is possible for there to be an initial delay between real-time and the video playback, that form of test will be conducted when the video anaglyph program is in a position to process live video. Various video files of different resolutions and sizes were processed, and a comparison of three are made here using a timer and the program’s ability to provide each video’s details:

The first video is a 2 minute and 6 second video at 320x230 resolution played at 15 fps. While the video was long, it was recorded in low quality and could be processed more easily.

```
General Properties:
         Name: 'Rabbit.avi'
         Path: 'C:\Users\User1\Documents\MATLAB'
     Duration: 125.9607
  CurrentTime: 0
    NumFrames: <Calculating...> learn more

Video Properties:
        Width: 320
       Height: 240
    FrameRate: 15.0000
 BitsPerPixel: 24
  VideoFormat: 'RGB24'
```

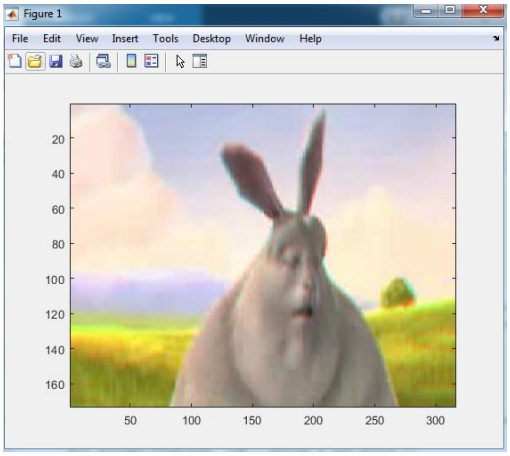

Figure 18: "Rabbit.avi" Test Video Data

The program has an issue displaying NumFrames for videos with many frames but this value can be calculated using Duration*FrameRate. NumFrames for this video is 1890.

It took 40.75s, 40.57s, 40.37s for three trials of this video to process, resulting in an average processing time of 40.56s. Relative to the video's frame rate, this is a processing speed of 46.58 fps, which is 310.5 % of the original framerate. The fact that the program processed the video at a higher framerate than the native video's frame rate means that the script had no problem processing the video into anaglyph and would most likely be able to convert this video in real-time if it needed to.

The second video is a 21.2 second video at 1280x720 resolution played at 25 fps. The video is similar to the target video in resolution and fps, but is of lower quality in general.

```
General Properties:
         Name: 'star_trails.avi'
         Path: 'C:\Users\User1\Documents\MATLAB'
     Duration: 21.2000
  CurrentTime: 0
    NumFrames: <Calculating...> learn more

Video Properties:
        Width: 1280
       Height: 720
    FrameRate: 25
 BitsPerPixel: 24
  VideoFormat: 'RGB24'
```

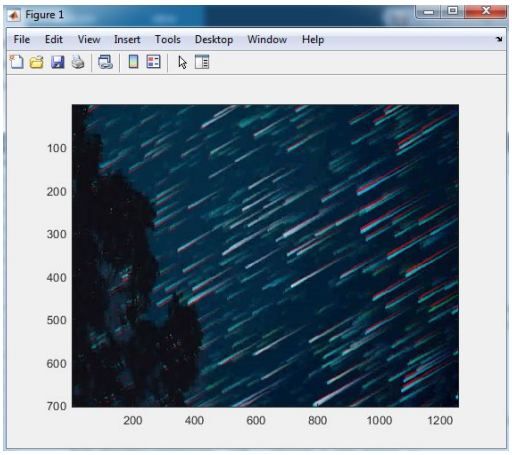

Figure 19: “star_trails.avi” Test Video Data

This video’s total number of frames is 530 frames.

For three trials of the software running this video, it took 47.89s, 48.76s, 48.37s for an average of 48.34s of processing. Relative to the original video’s frame rate, this is a processing speed of 10.964 fps, at 43.856 % the speed. It would lag behind processing this video in real-time, meaning the video anaglyph program hasn’t been optimized well enough to run at a fast enough speed or the hardware of the test computer isn’t powerful enough to process the video as fast as necessary.

The third video is a 2.1595 second video taken from an iPhone 12. It represents the target footage the analyph video conversion program is meant to process, although its video footage is in 1080p at 30 fps. The most powerful quality it can provide is footage at 4k resolution at 60 fps, which would be ideal for microsurgery, but is mainly available to the more expensive recent models and would be very demanding to process.

```
VideoReader with properties:

 General Properties:
         Name: 'Jiong.avi'
         Path: 'C:\Users\User1\Documents\MATLAB'
     Duration: 2.1595
  CurrentTime: 0
    NumFrames: 63

 Video Properties:
        Width: 1080
       Height: 1920
    FrameRate: 29.9700
  BitsPerPixel: 24
  VideoFormat: 'RGB24'
```

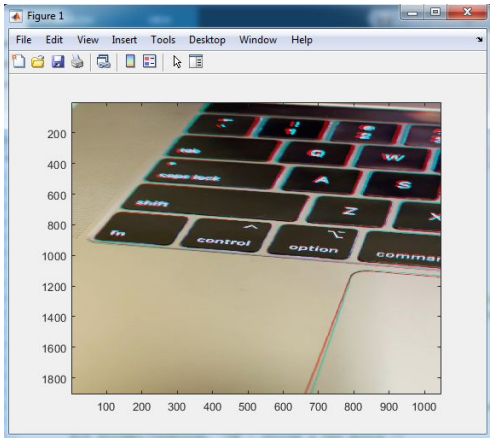

Figure 20: “Jiong” Test Video Data

This is the lowest target for the program to successfully process to meet the minimum conditions for the client’s purposes

The recorded times for three trials are 12.28s, 12.22s, 12.28s, with an average of 12.26s of processing. Relative to the video’s frame rate, this is a processing speed of 5.14 fps, which is only 17.146% of the original framerate. Due to the relatively high frame rate and high quality video resolution, the program and hardware struggled to process the video.

VI. Results

6.1 Software Development

The results of software development is basically the application itself. In the previous section, we showed how the application Camera Access and the zoom function performs. Therefore, here in the results section we will show the results of the testing code in the method captureVideoOutput(). The results are shown in figure x. With the application installed on the iPhone and a stable connection between the iPhone and the Macbook is present, there is no delay in between the start running of the application and the pop up of the testing message. In addition, the built in function of AVFoundation has no delay in displaying the moving object and zooming the camera view.

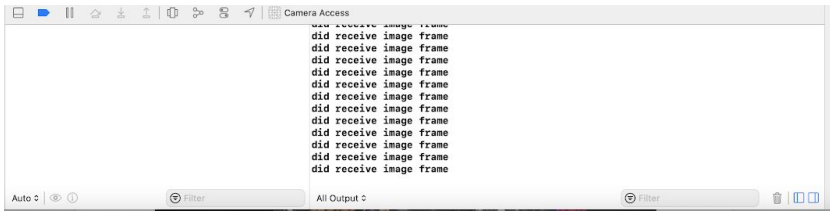

Figure 21. The testing results of outputting the frame. The message pop up at the same time the application is opened on the iPhone, which means the delay for getting the frame is minimal

6.2 Anaglyph

The results of the effectiveness of the anaglyph code is determined by the speed at which it can process video of varying quality. During testing, it was determined that the current iteration of the video anaglyph program is not strong enough to process standard iPhone video quality that is 1080p at 30fps.

Comparing the data from the results,

	Rabbit.avi	star_trails.avi	Jiong.avi
Original Video Frame Rate	15	25	29.97
Program Processing Speed in Frames per Second	46.58	10.964	5.14
Video Quality in Pixel Resolution	320x240	1280x720	1080x1920

Table 2: Testing Results from Processing Different Quality Videos

We can see that the hardware and software can run lower framerate and lower quality at faster speeds, but in the current state, only low quality videos can be processed in real-time. The base target processing of 1080p at 30 fps has yet to be reached, meaning the code still requires further development and more hardware tests need to be done as well.

VII. Discussion

7.1 Implication of Results

Current anaglyph programs fail to do real-time 3D conversion for standard iPhone video quality which is 1080p at 30fps. According to our research, no existing IOS app on the market is able to achieve this performance, and most are designed to add 3D effect to recorded videos and not live videos. A model based on a system with a multi-core CPU and a GPU was claimed to be able to achieve real-time 2D-3D conversion at 1080p, but not applied in commercial use yet.[23] Other proposed solution includes: A processor based method, comprising: employing an image sensing device to receive a first set of two dimensional (2D) images; employing one or more processors to perform actions. Another method transfers depth information from frames in the 3D reference database to the target frame while respecting object boundaries. It computes depth maps from the depth gradients, and outputs a stereoscopic video.

7.2 Ethical Considerations

7.2.1 Microsurgery in Developing Countries

The progress of microsurgery in low-income countries is much slower due to lack of access to the expensive microsurgical instruments. Limited resources impacts the training and development of skills of microsurgical surgeons. The wide gap between the need and availability prevents acquisition of medical treatment. With low-cost smartphone-based microsurgical scopes, the residents are able to receive regular operation training, which would facilitate the organizing of microsurgical teams. This meets the necessities for application of microsurgical techniques for treatment of patients.

7.2.2 HIPAA

The app could be used offline (not connected to the internet), which decreases the potential for unethical use of medical images. Issues of breach of patient confidentiality, privacy and lack of protection of images and patient data on the device may pose ethical and legal dilemmas, because of the ease with which data may be shared from smartphones or stolen by hackers [19].

Therefore, the IOS app should not save any data or record any videos, unless authorized by the patients. According to a study, most patients (207/280) preferred a doctor to be the person photographing them. Other healthcare personnel such as medical students, interns, nurses or medical photographers are less desired [20]. But medical photographs expose the surgeons to medico-legal risks, they must obey the laws to protect patients' privacy.

VIII. Conclusions

8.1 Design Summary

This project aims to design a simple iPhone-VR system to create a home microsurgery simulation tool that could be used as a resource for resident surgeons to practice microsurgery. To meet microsurgy requirements, this system needs to reach magnification up to 5x, with a latency less than 100ms. It's expected to have an anaglyph effect at a 4k resolution, or the highest resolution capable of being captured by commonly used smartphones. Now, A streamlined iPhone - 3D glasses model is proposed, which consists of one iphone, active 3D glasses, and a projection screen. The main task to accomplish this model is develop an IOS app. Currently, the code has met the purpose of getting access to the iPhone camera and allows the users to zoom in or zoom out to have a preferred view. The anaglyph algorithm is able to achieve the stereoscopic 3D effect by encoding each eye's image using filters of different (usually chromatically opposite) colors. The video processing speed is currently 5.14 frames per second at 1080p resolution. However, due to access to appropriate hardware, the team hasn't completed the app yet. Apple's IDE (Integrated Development Environment) for both Mac and iOS apps is XCode, which is the only go-to graphical interface people use to write iOS apps. It's available only to run on the Mac Operating System. Only one group member has MAC, so the anaglyph algorithm is developed and tested in MATLAB until it has successfully achieved the client's requirements.

8.2 Future Work

8.2.1 Software

The MATLAB script is expected to be converted to Xcode after it has been improved to satisfy the conditions of processing 1080p videos at 30fps.. The functionality should be tested on the iPhone and MacBook, and the degree of depth perception due to the anaglyph effect needs to be tested. If possible, optical zoom should be considered to develop, as the digital zoom in smartphones causes pixelation and reduction of picture quality.With the ability to maintain 4K resolution, the focus and brightness is expected to be adjusted from the screen. Finally, the code efficiency should be enhanced to realize real-time microsurgery operation.

8.2.2 Hardware

Once the coding is completed, the generated 3D video is expected to be projected onto a Macbook screen (or a monitor screen) and the resolution of 3D videos should be examined. Furthermore, evaluation of depth perception should be performed to ensure the microsurgery could be operated. While the intended target device for the client are last generation smartphones, the client has offered to advance testing to higher end-latest generation devices to test if low processing speeds can be overcome with better hardware.

IX. References

[1] M. Cenzato, A. Fratianni, and R. Stefini, "Using a Smartphone as an Exoscope Where an Operating Microscope is not Available," *World Neurosurg.*, vol. 132, pp. 114–117, Dec. 2019, doi: 10.1016/j.wneu.2019.08.137.

[2] W. Zhu, C. Gong, N. Kulkarni, C. D. Nguyen, and D. Kang, "Chapter 9 - Smartphone-based microscopes," in *Smartphone Based Medical Diagnostics*, J.-Y. Yoon, Ed. Academic Press, 2020, pp. 159–175.

[3] G. Pafitanis *et al.*, "The use of mobile computing devices in microsurgery," *Arch. Plast. Surg.*, vol. 46, no. 2, pp. 102–107, Mar. 2019, doi: 10.5999/aps.2018.00150.

[4] B. M. Mendez, M. V. Chiodo, D. Vandevender, and P. A. Patel, "Heads-up 3D Microscopy: An Ergonomic and Educational Approach to Microsurgery," *Plast. Reconstr. Surg. Glob. Open*, vol. 4, no. 5, May 2016, doi: 10.1097/GOX.0000000000000727.

[5] J. Choque-Velasquez, R. Colasanti, J. Collan, R. Kinnunen, B. Rezai Jahromi, and J. Hernesniemi, "Virtual Reality Glasses and 'Eye-Hands Blind Technique' for Microsurgical Training in Neurosurgery," *World Neurosurg.*, vol. 112, pp. 126–130, Apr. 2018, doi: 10.1016/j.wneu.2018.01.067.

[6] G. Pafitanis *et al.*, "Evolution of an evidence-based supermicrosurgery simulation training curriculum: A systematic review," *J. Plast. Reconstr. Aesthetic Surg. JPRAS*, vol. 71, no. 7, pp. 976–988, 2018, doi: 10.1016/j.bjps.2018.04.005.

[7] I. Badash, D. J. Gould, and K. M. Patel, "Supermicrosurgery: History, Applications, Training and the Future," *Front. Surg.*, vol. 5, Mar. 2018, doi: 10.3389/fsurg.2018.00023.

[8] S. Capkin, A. Cavit, and T. Kaleli, "Microsurgery training with smartphone," *Handchir. · Mikrochir. · Plast. Chir.*, vol. 50, no. 6, pp. 443–445, Dec. 2018, doi: 10.1055/a-0661-6015.

[9] A. Huotarinen, M. Niemelä, and B. R. Jahromi, "Easy, Efficient, and Mobile Way to Train Microsurgical Skills During Busy Life of Neurosurgical Residency in Resource-Challenged Environment," *World Neurosurg.*, vol. 107, pp. 358–361, Nov. 2017, doi: 10.1016/j.wneu.2017.08.024.

[10] I. G. Margulies, H. Xu, and P. W. Henderson, "Microsurgery Training in the Digital Era: A Systematic Review of Accessible Digital Resources," *Ann. Plast. Surg.*, vol. 85, no. 4, pp. 337–343, Oct. 2020, doi: 10.1097/SAP.0000000000002214.

[11] A. Naqvi, N. Manglik, E. Dudrey, C. Perry, Z. D. Mulla, and J. L. Cervantes, "Evaluating the performance of a low-cost mobile phone attachable microscope in cervical cytology," *BMC Womens Health*, vol. 20, no. 1, p. 60, Mar. 2020, doi: 10.1186/s12905-020-00902-0.

[12] "Highest Resolution Microsurgery Microscope | MM51," *Mitaka USA*. https://mitakausa.com/mm51/ (accessed Sep. 18, 2020).

[13] M. Xi, L.-H. Wang, Q.-Q. Yang, D.-X. Li, and M. Zhang, "Depth-image-based rendering with spatial and temporal texture synthesis for 3DTV," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, p. 9, Feb. 2013, doi: 10.1186/1687-5281-2013-9.

[14] "Anaglyphs : About : Stereogranimator." http://stereo.nypl.org/about/anaglyph (accessed Oct. 07, 2020).

[15] "iPhone 8 Plus - Technical Specifications." https://support.apple.com/kb/sp768?locale=en_US (accessed Oct. 07, 2020).

[16] "iPhone XR - Technical Specifications," *Apple*. https://www.apple.com/iphone-xr/specs/ (accessed Oct. 07, 2020).

[17] John Doe , Ritvik Upadhya, Hanny, Thyselius, Sherwin Zadeh, Chathurka, and Chan Jing Hong, "Pinch to zoom camera," Stack Overflow, Nov-2015. [Online]. Available: https://stackoverflow.com/questions/33180564/pinch-to-zoom-camera.

[18] A. Ajwani, “How to process images real-time from the iOS camera,” Medium, 11-May-2020. [Online]. Available: https://medium.com/@anuragajwani/how-to-process-images-real-time-from-the-ios-camera-9c416c531749.

[19] Nair, A.G., Potdar, N.A., Dadia, S. et al. Patient perceptions regarding the use of smart devices for medical photography: results of a patient-based survey. Int Ophthalmol 39, 783–789 (2019). https://doi.org/10.1007/s10792-018-0878-2

[20] Dumestre, D. O., & Fraulin, F. (2017). Balancing the Need for Clinical Photography With Patient Privacy Issues: The Search for a Secure SmartPhone Application to Take and Store Clinical Photographs. Plastic surgery (Oakville, Ont.), 25(4), 255–260. https://doi.org/10.1177/2292550317731761

[21] “How to Interface MATLAB with XCode,” *Stack Overflow*. https://stackoverflow.com/questions/17938706/how-to-interface-matlab-with-xcode (accessed Dec. 09, 2020).

[22] “How to deploy a Matlab app to iOS and Android? - MATLAB Answers - MATLAB Central.” https://www.mathworks.com/matlabcentral/answers/71539-how-to-deploy-a-matlab-app-to-ios-and-android (accessed Dec. 09, 2020).

[23] S. Tsai, C. Cheng, C. Li and L. Chen, "A real-time 1080p 2D-to-3D video conversion system," in *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 915-922, May 2011, doi: 10.1109/TCE.2011.5955240.

X. Appendix A

Preliminary Product Design Specifications

Function:

The clients' current model is combining virtual reality glasses with a computer, two cell phones, and a lightning cable connection to perform microsurgery. However, there is too much delay and the team is expected to design a simple streamlined iPhone-VR system to create a home microsurgery simulation tool that could be used as a resource for resident surgeons to practice microsurgery.

Client Requirements:

- Magnification: 1:12.
- Ensures an immersive experience for the entire surgical team due to its big-screen 3D imaging.
- Minimize the delay of real-time 3D imaging (currently around 1 ms)
- Make the model more streamlined with fewer devices. The expectation is to use a single iPhone for recording and casting to the Macbook screen so that surgeons could use VR glasses to directly look at the screen.

Design Requirements:

- *Performance Requirements:*
    - The device should be as fast as possible to improve the VR viewing experience
        - Minimize display lag between the smartphones and the MacBook

- *Safety:*
    - The device should minimize unnecessary visuals to the user's eyes
        - Not overly bright, no sudden flashes, reduce motion blur
- *Accuracy and Reliability:*
    - The ratio of the original camera image to the one seen should be maintained
- *Life in Service:*
    - The device should be able to be used multiple times for as long as the shelf life
- *Shelf Life:*
    - The device should ideally last as long as the smartphone/MacBook in use remains functional
- *Operating Environment*
    - The device will be exposed to normal conditions, such as room temperature, and will be used by resident surgeons
- *Ergonomics:*
    - The device should be fairly portable as it will likely be worn by the user
- *Size:*
    - The device should minimize the number of devices, such as smartphones and MacBooks, used.
- *Weight:*
    - The device currently consists of an iPhone XR (194g [1]), an iPhone 8(148g [2]), 1 Macbook 13" with retina display computer (1551g [3]), lightning cables (negligible), and an articulating arm phone mount stand (440g with a maximum load of 500g [4]). Therefore, the total amount of weight is estimated to be 2333g.
- *Materials:*
    - The device does not have certain materials that should be used for fabrication. However, the material used in the VR set should be considered. The current model is using google cardboard as our VR headset, and the material from the

google cardboard such as the head strap may cause some uncomfortableness according to some customer reviews.

- *Aesthetics, Appearance, and Finish:*

    - The device aims to perform a mock surgery at home, so it should use as many materials available at home as possible. In this case, this platform will result in an iPhone clamped on the mount and connected to a MacBook Pro, and another iPhone will wirelessly be connected to the MacBook and placed in the google cardboard.

Production Characteristics:

- *Quantity:*

    - Nowadays, in developing countries, the demand for microsurgery, especially in the field of plastic surgery, has increased tremendously. It is estimated that in Zimbabwe there are only 0.03 plastic surgeons in every 100,000 people, which is a very low number compared to 1.98 plastic surgeons per 100,000 people in the United States [5]. Additionally, the lack of facilities for surgeons to practice will also cause high demand for experienced surgeons. In this case, the development of a VR platform that mimics the surgical situation with depth perception will help surgeons in developing countries.

- *Target Product Cost:*

    - The current model requires two smartphones, one computer, a connection cable, and a smartphone stand mount, but in the future, we may reduce the smartphones or computers used in the product. We would assume that the user already has at least one smartphone and a cable. Thus, although the total cost of the product may vary, the maximum cost for this platform is to buy another smartphone, a computer, and a smartphone stand mount.

Miscellaneous:

- *Standards and Specifications:*

    - The device, similar to its current professional version used by surgeons, is a class 1 device and is exempt from requiring FDA Premarket Approval. To receive a CE mark for EU approval for a class I medical device, the device needs a Declaration of Conformity registered with a Competent Authority. ISO 10936-1:2017

specifies requirements and refers to test methods for operation microscopes used for observation during surgical operation and treatment of patients, but it does not apply to accessories, e.g. photographic cameras. [6]

- *Customer:*
    - The customer has a preference for the components of the device to be cheap and easy to obtain. There is potential to market the device in underdeveloped countries where there are limitations in money and technology available. The device should be able to hook up to external monitors for the potential viewing of students in a teaching environment. For better ease of use, the operator of the device should have the ability to adjust the zoom and move the optics of the device as needed without any trouble. The device needs a depth of field such that the operator is able to effectively wield instruments while using the device - however, the distance required is short, less than 30 centimeters in practice, and no peripheral vision is required. The client is considering between two iterations of the device, one iteration being stand-mounted, and the other being a form of headset the user wears, and both routes for the design still need to be evaluated for their practicality and performance.
- *Patient-Related Concerns:*
    - Whether it is used for training or for a real operation, the device comes into contact with its user and must be cleaned after every use. It should not, however, come into contact with a potential patient. The device does take footage of the operation, which requires the user's consent, and if it were to be used to operate on a patient, it would need the patient's consent.
- *Competition:*
    - The competition to this device is not an already used professional device such as what the client uses (MM51 YOH Surgical Microscope System, MSRP $310,000), but any proposed cheapened alternative that offers great magnification and resolution for microsurgical practices using as few components as possible. [7]
        - eoSurgical has microsurgery simulations using your own devices such as phones, tablets, and tv screens, but their cheapest product, the eoMicro, costs £82.50 ($106.84 USD). It has its own stand, which is rather bulky and cannot be used outside of training. [8]
        - Pocket Suture has a Pocket Microsurgery Trainer for $145.00. It is too simplistic for the client's needs by only requiring a phone and does not

solve the client's problem with their current proposed device (the lag between devices). It is also only for individual training. [9]

Literature Cited:

[1] "iPhone XR - Technical Specifications," *Apple*. [Online]. Available: https://www.apple.com/iphone-xr/specs/. [Accessed: 18-Sep-2020].

[2] "iPhone 8 - Technical Specifications," *Official Apple Support*. [Online]. Available: https://support.apple.com/kb/SP767?viewlocale=en_US&locale=en_US. [Accessed: 18-Sep-2020].

[3] "MacBook Pro 13-inch - Technical Specifications," *Apple*. [Online]. Available: https://www.apple.com/macbook-pro-13/specs/. [Accessed: 18-Sep-2020].

[4] "Articulating Arm Phone Mount Stand for Baking Crafting Demo Videos/Live Streaming - Acetaken" *Amazon.* [Online]. Available: https://www.amazon.com/Articulating-Baking-Crafting-Videos-Streaming/dp/B07B93ZJYM. [Accessed: 18-Sep-2020].

[5] S. M. Inchauste, P. L. Deptula, J. T. Zelones, R. S. Nazerali, D. H. Nguyen, and G. K. Lee, "Global Health Microsurgery Training With Cell Phones," *Annals of Plastic Surgery*, vol. 84, 2020.

[6]International Organization for Standardization. (2017). *Optics and photonics — Operation microscopes — Part 1: Requirements and test methods* R(ISO/DIS Standard No. 10936-1:2017). Retrieved from https://www.iso.org/obp/ui/#iso:std:iso:10936:-1:dis:ed-2:v1:en

[7] "Highest Resolution Microsurgery Microscope | MM51," *Mitaka USA*. https://mitakausa.com/mm51/ (accessed Sep. 18, 2020).

[8]"eoSim Core + SurgTrac," *eoSurgical*. https://www.eosurgical.com/products/surgtrac-core (accessed Sep. 18, 2020).

[9] "Pocket Microsurgery Trainer™," *Pocket Suture*. https://www.pocketsuture.com/products/pocket-microsurgical-trainer (accessed Sep. 18, 2020).

Appendix B

"anaglyph.m" function

```
function [resultImage] = anaglyph(image1, image2, filter1, filter2)

%% Basic function - Creates stereoscopic image


% Saving function inputs to workspace.

assignin('base','image1',image1);

assignin('base','image2',image2);

assignin('base','fil1',filter1);

assignin('base','fil2',filter2);


% Reading the images.

img1 = imread(image1);

img2 = imread(image2);


% Adjusting the colors in both images, based on input filters.

switch filter1
```

```
        case 'red'
        img1(:,:, 2:3) = 0;
        case 'green'
        img1(:,:, 1:2:3) = 0;
        case 'blue'
        img1(:,:, 1:2) = 0;
        case 'cyan'
        img1(:,:, 1) = 0;
        case 'magenta'
        img1(:,:, 2) = 0;
end
switch filter2
        case 'red'
        img2(:,:, 2:3) = 0;
        case 'green'
        img2(:,:, 1:2:3) = 0;
        case 'blue'
        img2(:,:, 1:2) = 0;
        case 'cyan'
        img2(:,:, 1) = 0;
```

```
        case 'magenta'
        img2(:,:, 2) = 0;
end

resultImage = img1 + img2;
```

Appendix C

"one_image_anaglyph.m" function

```
function [x] = one_image_anaglyph(image)

assignin('base','frame',image);

parallaxAngle = 0.5 ;           % You can change this to suit yourself.

%leftImage = imread(image);
leftImage = image;

J = imrotate(leftImage,(parallaxAngle));
```

```
% rightImage = imread(image);

rightImage = image;



K = imrotate(rightImage, (-parallaxAngle));



%Here's how this works:

% [J1,rect] = imcrop(J) activates a custom crop menu where I must crop the

% image J, and it will be saved as J1. This must be exclusive per image.

% The parameters for the crop window are saved in rect.

% We can apply imcrop with rect as below with K1



rect =
[9.510000000000000,21.510000000000000,1.057980000000000e+03,1.883980000000000e+03]
;

%rect = [100, 200, 400 300];

J1 = imcrop(J, rect);

K1 = imcrop(K, rect);



%subplot(1,2,1), imshow(J1);

%subplot(1,2,2), imshow(K1);
```

```
imwrite(J1, 'leftImage.jpg');

imwrite(K1, 'rightImage.jpg');


% x1 = 'leftImage.jpg';

% y1 = 'rightImage.jpg';


[x] = anaglyph('leftImage.jpg', 'rightImage.jpg', 'red', 'cyan');
```

Appendix D

"extract_movie_into_frames.m" Master script

```
clc;      % Clear the command window.

close all;  % Close all figures (except those of imtool.)

imtool close all;  % Close all imtool figures.

clear;  % Erase all existing variables.

workspace;  % Make sure the workspace panel is showing.

fontSize = 22;


% First get the folder that it lives in.

folder = fileparts(which('Jiong.avi')); % Determine where demo folder is (works with all
versions).

% Pick one of the two demo movies shipped with the Image Processing Toolbox.
```

```
% Comment out the other one.

movieFullFileName = fullfile(folder, 'Jiong.avi');

% movieFullFileName = fullfile(folder, 'traffic.avi');

% Check to see that it exists.

if ~exist(movieFullFileName, 'file')

   strErrorMessage = sprintf('File not found:\n%s\nYou can choose a new one, or cancel',
movieFullFileName);

   response = questdlg(strErrorMessage, 'File not found', 'OK - choose a new movie.', 'Cancel',
'OK - choose a new movie.');

   if strcmpi(response, 'OK - choose a new movie.')

         [baseFileName, folderName, FilterIndex] = uigetfile('*.avi');

         if ~isequal(baseFileName, 0)

               movieFullFileName = fullfile(folderName, baseFileName);

         else

               return;

         end

   else

         return;

   end

end
```

```
try
    videoObject = VideoReader(movieFullFileName)
    % Determine how many frames there are.
    numberOfFrames = videoObject.NumFrames;
    vidHeight = videoObject.Height;
    vidWidth = videoObject.Width;


    numberOfFramesWritten = 0;
    % Prepare a figure to show the images in the upper half of the screen.
    %figure;
    %     screenSize = get(0, 'ScreenSize');
    % Enlarge figure to full screen.
    %set(gcf, 'units','normalized','outerposition',[0 0 1 1]);


    % Ask user if they want to write the individual frames out to disk.
    %promptMessage = sprintf('Do you want to save the individual frames out to individual disk
files?');
    %button = questdlg(promptMessage, 'Save individual frames?', 'Yes', 'No', 'Yes');
    %if strcmp(button, 'Yes')
            %writeToDisk = true;
```

```
	% Extract out the various parts of the filename.
	%[folder, baseFileName, extentions] = fileparts(movieFullFileName);
	% Make up a special new output subfolder for all the separate
	% movie frames that we're going to extract and save to disk.
	% (Don't worry - windows can handle forward slashes in the folder name.)
	%folder = pwd;   % Make it a subfolder of the folder where this m-file lives.
	%outputFolder = sprintf('%s/Movie Frames from %s', folder, baseFileName);
	% Create the folder if it doesn't exist already.
	%if ~exist(outputFolder, 'dir')
		%mkdir(outputFolder);
	%end
%else
	%writeToDisk = false;
%end


% Loop through the movie, writing all frames out.
% Each frame will be in a separate file with unique name.
%meanGrayLevels = zeros(numberOfFrames, 1);
%meanRedLevels = zeros(numberOfFrames, 1);
%meanGreenLevels = zeros(numberOfFrames, 1);
```

```
%meanBlueLevels = zeros(numberOfFrames, 1);
for frame = 1 : numberOfFrames
    % Extract the frame from the movie structure.
    thisFrame = read(videoObject, frame);

    % Display it
    %hImage = subplot(1, 1, 1);
    %image(thisFrame);
    %caption = sprintf('Frame %4d of %d.', frame, numberOfFrames);
    %title(caption, 'FontSize', fontSize);
    %drawnow; % Force it to refresh the window.

    % Anaglyph it and display it
    aImage = subplot(1,1,1);


    image(one_image_anaglyph(thisFrame));

    % [x] = one_image_anaglyph(thisFrame);
    % image(anaglyph(x, y, 'red', 'cyan'));
```

```
%caption = sprintf('Frame %4d of %d.', frame, numberOfFrames);
%title(caption, 'FontSize', fontSize);
 drawnow; % Force it to refresh the window.


 % Write the image array to the output file, if requested.
 %if writeToDisk
        % Construct an output image file name.
        %outputBaseFileName = sprintf('Frame %4.4d.png', frame);
        %outputFullFileName = fullfile(outputFolder, outputBaseFileName);


        % Stamp the name and frame number onto the image.
        % At this point it's just going into the overlay,
        % not actually getting written into the pixel values.
        %text(5, 15, outputBaseFileName, 'FontSize', 20);


        % Extract the image with the text "burned into" it.
        %frameWithText = getframe(gca);
        % frameWithText.cdata is the image with the text
```

```
        % actually written into the pixel values.
        % Write it out to disk.
        %imwrite(frameWithText.cdata, outputFullFileName, 'png');
%end

% Calculate the mean gray level.
%grayImage = rgb2gray(thisFrame);
%meanGrayLevels(frame) = mean(grayImage(:));

% Calculate the mean R, G, and B levels.
%meanRedLevels(frame) = mean(mean(thisFrame(:, :, 1)));
%meanGreenLevels(frame) = mean(mean(thisFrame(:, :, 2)));
%meanBlueLevels(frame) = mean(mean(thisFrame(:, :, 3)));

% Plot the mean gray levels.
%hPlot = subplot(2, 2, 2);
%hold off;
%plot(meanGrayLevels, 'k-', 'LineWidth', 3);
%hold on;
%plot(meanRedLevels, 'r-', 'LineWidth', 2);
```

```
        %plot(meanGreenLevels, 'g-', 'LineWidth', 2);

        %plot(meanBlueLevels, 'b-', 'LineWidth', 2);

        %grid on;


        % Put title back because plot() erases the existing title.

        %title('Mean Gray Levels', 'FontSize', fontSize);

        %if frame == 1

        %    xlabel('Frame Number');

        %    ylabel('Gray Level');

        %    % Get size data later for preallocation if we read

        %    % the movie back in from disk.

        %    [rows, columns, numberOfColorChannels] = size(thisFrame);

        %end


        % Update user with the progress.  Display in the command window.

        %if writeToDisk

                %progressIndication = sprintf('Wrote frame %4d of %d.', frame,
numberOfFrames);

        %else

                %progressIndication = sprintf('Processed frame %4d of %d.', frame,
numberOfFrames);
```

```
%end

%disp(progressIndication);

% Increment frame count (should eventually = numberOfFrames

% unless an error happens).

%numberOfFramesWritten = numberOfFramesWritten + 1;



% Now let's do the differencing

%alpha = 0.5;

%if frame == 1

        %Background = thisFrame;

%else

        % Change background slightly at each frame

        %                Background(t+1)=(1-alpha)*I+alpha*Background

        %Background = (1-alpha)* thisFrame + alpha * Background;

%end

% Display the changing/adapting background.

%subplot(2, 2, 3);

%imshow(Background);

%title('Adaptive Background', 'FontSize', fontSize);

% Calculate a difference between this frame and the background.
```

```
        %differenceImage = thisFrame - uint8(Background);

        % Threshold with Otsu method.

        %grayImage = rgb2gray(differenceImage); % Convert to gray level

        %thresholdLevel = graythresh(grayImage); % Get threshold.

        %binaryImage = im2bw( grayImage, thresholdLevel); % Do the binarization

        % Plot the binary image.

        %subplot(2, 2, 4);

        %imshow(binaryImage);

        %title('Binarized Difference Image', 'FontSize', fontSize);

    end



    % Alert user that we're done.

    %if writeToDisk

        %finishedMessage = sprintf('Done!  It wrote %d frames to folder\n"%s"',
numberOfFramesWritten, outputFolder);

    %else

        %finishedMessage = sprintf('Done!  It processed %d frames of\n"%s"',
numberOfFramesWritten, movieFullFileName);

    %end

    %disp(finishedMessage); % Write to command window.

    %uiwait(msgbox(finishedMessage)); % Also pop up a message box.
```

```
% Exit if they didn't write any individual frames out to disk.
%if ~writeToDisk
	%return;
%end

% Ask user if they want to read the individual frames from the disk,
% that they just wrote out, back into a movie and display it.
%promptMessage = sprintf('Do you want to recall the individual frames\nback from disk into a movie?\n(This will take several seconds.)');
%button = questdlg(promptMessage, 'Recall Movie?', 'Yes', 'No', 'Yes');
%if strcmp(button, 'No')
	%return;
%end

% Create a VideoWriter object to write the video out to a new, different file.
%writerObj = VideoWriter('NewGlobe.avi');
%open(writerObj);

% Read the frames back in from disk, and convert them to a movie.
% Preallocate recalledMovie, which will be an array of structures.
```

```
% First get a cell array with all the frames.
%allTheFrames = cell(numberOfFrames,1);
%allTheFrames(:) = {zeros(vidHeight, vidWidth, 3, 'uint8')};
% Next get a cell array with all the colormaps.
%allTheColorMaps = cell(numberOfFrames,1);
%allTheColorMaps(:) = {zeros(256, 3)};
% Now combine these to make the array of structures.
%recalledMovie = struct('cdata', allTheFrames, 'colormap', allTheColorMaps)
%for frame = 1 : numberOfFrames
    % Construct an output image file name.
    %outputBaseFileName = sprintf('Frame %4.4d.png', frame);
    %outputFullFileName = fullfile(outputFolder, outputBaseFileName);
    % Read the image in from disk.
    %thisFrame = imread(outputFullFileName);
    % Convert the image into a "movie frame" structure.
    %recalledMovie(frame) = im2frame(thisFrame);
    % Write this frame out to a new video file.
    %writeVideo(writerObj, thisFrame);
%end
%close(writerObj);
```

```
    % Get rid of old image and plot.
    %delete(hImage);
        %delete(aImage);
    %delete(hPlot);
    % Create new axes for our movie.
    %subplot(1, 3, 2);
    %axis off;  % Turn off axes numbers.
    %title('Movie recalled from disk', 'FontSize', fontSize);
    % Play the movie in the axes.
    %movie(recalledMovie);
    % Note: if you want to display graphics or text in the overlay
    % as the movie plays back then you need to do it like I did at first
    % (at the top of this file where you extract and imshow a frame at a time.)
    %msgbox('Done with this demo!');

catch ME
    % Some error happened if you get here.
    strErrorMessage = sprintf('Error extracting movie frames from:\n\n%s\n\nError: %s\n\n)',
movieFullFileName, ME.message);
    uiwait(msgbox(strErrorMessage));
end
```

Appendix E

Expenses

Item	Description	Manufacturer	Part Number	Date	QTY	Cost Each	Total	Link
Component 1								
Component 2								
Component 3								
TOTAL:								**$0.00**

Due to the client's design specifications and the development of the project to be software based, no purchases have been made so far. The client has their own prototypes and parts that they have been testing with, but the team has not had any access to these items due to safe adherence in accordance with social distancing.