MIRRAR: Augmented Reality Product Display

Taehwan Chon, CSE, Jyerrmee Nerona, CSE, John Nguyen, CSE, and Kevin Pierre, CSE

Abstract—When buying a pair of glasses frames one of the greatest concerns of the customer is its look on the face. Despite business owners moving their stock into warehouses and their commerce online, it is impossible to recreate how glasses would look on a particular face online without precise tracking and measurements. This disparity causes glasses, and other apparel, stores to continue to see a majority of their sales occur at physical storefronts. Our solution will allow glasses storefronts the ability to have their entire stock at all times, even if they don’t have it physically. With our designed adjustable reference glasses frames, and an android app, a customer will be able to get an accurate look at how any pair of glasses will look on their head. We split up the project into four different sections. One dedicated to app development, another to hardware design, another to facial recognition, and finally one to reference frame design. Our end goal is to combine the four different sections, into one whole product.

I. INTRODUCTION

Businesses are created with one common goal, to create revenue. Everything else stems from this principle including increasing traffic, pleasing customers, advertising, or optimal consumer experience. However, the marketplace is a volatile battlefield that businesses compete on and failing to constantly adapt to their environment will result in the failure of the business. We see this everywhere. A monumental example of this was the fall of Toys R Us who lost to Amazon, the rising monopoly in retail marketing. Jeff Bezos, the CEO of Amazon, is someone who knows how to survive in the marketplace. He adapts to his environment and also looks into the future to make predictions. He is at the forefront of the shift to e-commerce and continues to lead the movement. In an age where the Internet is becoming a staple in the overwhelming majority of consumers, it is absolutely critical for businesses to transition their platform online. However, this is difficult for businesses that deal with consumers wearing or trying on their product.

Many online clothing companies offer to pay for shipping on returns allowing customers to shop with no risk, but the current solutions are inefficient and have the potential for major change. Although there are a lot of markets that involve fashion and clothing products, the niche we chose to target is eyeglass companies. With MIRRAR, consumers can accurately view different models of glasses in real time on their head using augmented reality (AR) on an Android Tablet.

Other companies are just beginning to catch onto this trend. At the beginning of this year, Amazon released a patent on an AR mirror that allows users to try on virtual clothes in different virtual locations. For example, a user could see what they would look like wearing a swimsuit at a beach. Sephora is another company progressing towards this trend through an AR mirror that simulates makeup on a user’s face in real time. This is convenient as the customer no longer has to take off the makeup every time they want to try a product.

MIRRAR makes use of a 3D printed pair of reference frames that provide accurate measurements on the user’s head shape and face. Although many companies have achieved, and almost perfected 2D AR on mobile applications, the angled movement is difficult to achieve. With the integration of our hardware implementation and Android application, we are able to create a simple, accurate experience for the user.

Designing MIRRAR began with creating a list of specifications to use as a guideline and ensure our system provides the best solutions possible. These can be seen in Table 1. The main difference with our system compared to other similar ones is the incorporation of external hardware. To guarantee this was a worthy business investment, we had to make sure the additional measurements were useful and as accurate as possible while keeping costs minimal.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of Motion</td>
<td>≤ 90°</td>
</tr>
<tr>
<td>Display Delay</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>Operational Range</td>
<td>&lt; 5 feet</td>
</tr>
<tr>
<td>Cost</td>
<td>$3D PRINTING PRICE</td>
</tr>
<tr>
<td>Battery Life</td>
<td>7.2 Hours</td>
</tr>
<tr>
<td>Weight</td>
<td>51.9 grams</td>
</tr>
</tbody>
</table>

The reference frames will provide up to 90 degrees of motion, in other words, full angled movement. We are also using 3D printing technology for both design and economic purposes. We want to ensure they are lightweight and no heavier than normal glasses. Typical sunglasses weigh up to 50 grams and our frames currently weigh 51.9 grams. With the next generation, we plan to add additional components such as the battery, additional potentiometers, and the custom PCB while maintaining this range of mass. The custom PCB
will be small, however, will still require power. Taking into account the components we are adding to the PCB, the frames will require 3.3V of power and 30mA of current to run, resulting in a runtime of 7.2 hours.

Another important consideration while programming MirrAR was to ensure real-time display to provide a seamless user experience. This meant keeping the delay between the image display and user movement less than one second while keeping operation range within five feet. We also provide the ability to scale the resolution to account for a user varying the distance between them and the camera. For example, a user might want a closer look at the screen and will lean in. Through this, we hope to help eyeglass companies move forward to the future and beat their competition.

II. Design

A. Overview

There are two major components of approaching this system. One is the Android Application which will act as the user interface, and the other is the reference frames as shown in Figure 1.

B. Android Application

MirrAR is programmed in Android Studio using Java. This platform provides all of the tools necessary to approach Android application development. It provides both SDK Platforms, which is the operating system the application will be compatible with, and SDK Tools which are resources provided by Android Studio to enable specific development and features. The minimum SDK version required is Android 6.0 Marshmallow and we are targeting the latest OS version which is Android 9.0 Pie.

For our project, we make use of the following SDK Tools:

- **Android SDK Build-Tools:** Used to debug, build, run, and test an Android application.
- **Android Emulator:** Allows us to emulate an Android device to test our program without actually having to upload an APK file onto an Android device
- **Android SDK Platform-Tools:** Used to support the features for the current android platform
- **Android SDK Tools:** Downloadable component for the Android SDK that includes the complete set of development and debugging tools for the Android SDK
- **Intel x86 Emulator Accelerator:** As its name suggests, it’s used to speed up Android application emulation on a host machine.

The tools and platforms were chosen to take into consideration the tablet we would use to interact with the application. We selected the ASUS Zenpad 3s 10 while taking into consideration our budget and project goals, deciding that this was the best mid-range tablet for our purposes. There were a few features that were essential in our selection:

- **IPS (In-Plane Switching) Display:** This is one of three major screen technologies used in LCD panels which involves manipulating the molecular orientation between glass substrates. IPS screen technology allows for:
  - Wide viewing angle: The tablet provides a 178-degree viewing angle.
  - Better color reproduction/representation
  - Better response times
- **Hexacore Processor:** This tablet makes use of the following processors that handle different tasks for optimization
  - Quad Core: Handles graphics
  - Dual Core: Handles all other processes
- **The Camera:** This tablet provides an 8MP rear webcam and 5MP front webcam which are not far behind from leading competitors and fully satisfy industry standards.

Taking into consideration all of the specifications, the ASUS Zenpad 3s 10 was the best tablet for us at the best price. However, we need to have the application communicate with the hardware. Luckily the Android OS provides many libraries and tools to achieve this. We require this communication so that the reference frames can send the necessary measurements to the application and from there MirrAR will provide model recommendations and an accurate display of those models.

While programming MirrAR, a major issue we ran into was memory usage and would receive the OOM Exception (Out of Memory). As our application provides an inventory of all available models, there must be a database that stores the images of all these models. In order to handle this, we created a class in Android Studio called ImageResize which uses bitmaps to obtain the dimensions of the images and reduces them to the appropriate resolution for the device the application is downloaded onto. This class can handle any device, thus this error will not occur no matter the size of the device.

However, all current inventory and images are in 2D and do not have complete facial recognition software implemented. In other words, we must find out how to create and upload 3D models as well as applying the facial recognition software to them. There are a plethora of programs that support 3D modeling in Android, however, we will begin with DesertRain Studios and continue to build the application in Android Studio.

C. Facial Recognition Software Design
Our facial recognition software is able to detect and identify facial features such as the eyes, nose, and mouth by placing a distinct colored rectangle around them in real time.

```java
Class detectFace {
  - Activate device camera
  - Detect face in live feed and store in buffer
  - Implement classifiers into code
  - Once facial features detect, draw rectangle around face, place in memory

  Above, is general pseudocode for how the facial recognition software works. To start, the device’s camera needs to be activated. Then, the classifiers are able to determine if a face is present in an image that is provided (or in this case, a live image). The way this works is first, the image is run through a buffer and saved. The classifiers then scan the image (or live feed) and start analyzing the pixel values. The face and its features all have their own separate values. In the recognition software, the live feed is analyzed first to see if there is a face present. If there is no face, the software does not continue to detect anything face related. If there is a face present, the software will detect and box the face and proceed to the next steps.
```

![Image](image.png)

**Figure 4: Example of how pixel values are analyzed.**

The next step is to detect individual features, such as the eyes, nose, and mouth. Figure 4 is an example of how these features are distinguished on a person’s face. An eye has a different pixel value than the forehead and cheeks (which have their own values). Each feature is checked by comparing them to their surroundings and once they are determined to be a feature, a box is placed around them. Once all the features are detected, the image/live feed will be restored in the memory and constantly updated by pulling the most recent image/live feed from the buffer to analyze. The process then repeats.

The next goal for this software is detecting the side profile of faces to achieve angled display of our 3D models. This entails adding extra measurements to detect the ears and make estimations for the sides of the user’s face. This extra measurement will help us in resizing our 3D models. In the end, this recognition code will be implemented into the Android Application and be responsible for detecting the facial features so that the models can be projected accurately on the user’s face.

We will also use techniques from Computer Systems 1 and 2 by manipulating the camera and filtering the image. In these classes, an FPGA was used and the coding was in C, but the general process is very similar. We will be able to activate the camera in a similar way. Programming in C is comparable to programming in Java which allows us to incorporate methods into the code and connect the software to the hardware. General coding techniques from Computer Science 121 and 187 will be used to develop the code. For example, it is easier to know how to create methods and their desired actions, which loops (“for” or “while”) to use and when to use them, and how to terminate a software with an “Error” message when an undesired action has occurred or the runtime for a process is taking too long.

This recognition software will be coded in Java and will be taking advantage of the OpenCV library. An emulator on the laptop will be used to test the software before exporting to the Android Application. To be able to execute the creation of this facial recognition code, we will need further research on the OpenCV library and the types of methods it has and how to incorporate them. We also need to learn how to convert the Java code into a file that will be compatible with Android Studio. This conversion will allow the code to run on our Android Application via tablet instead of the laptop. Finally, the last thing we need to learn is how to place the 3D models of the glasses onto a person’s face by utilizing the landmarks that the facial recognition detects. The landmarks will need to be used since we want the placement to look as accurate and realistic on a person’s face as possible. Once this is all done, the final thing we would like to learn is how to derive measurements from the detection software, such as how far landmarks are from each other or possibly how far a person is from the camera. The reason distance measurement can be useful to learn is because we can compare the measurement received from the hardware to the measurement in the software. By doing this we can check for deviation in hardware and software measurements.

The experiments that are performed on the facial recognition software prior to uploading to the tablet is done through testing on an emulator. The emulating software, “Netbeans”. allows the use of one’s webcam. By using the “gui” package in “Netbeans”, the emulator allows for the creation of a user interface on the laptop screen which includes but is not limited to a frame where the webcam image can be seen and a start and pause button. The image from the webcam will be displayed on the laptop and if the
software can detect the facial features, the boxes around them will be drawn. This will conclude the test and the results will be taken as a success.

The results of each test will be analyzed by seeing how accurate they are. If a facial feature is detected incorrectly, the method will need to be adjusted and the code will need to be retested. This will keep repeating until each detecting method works perfectly. Eventually, we will only have the most important detecting methods implemented to not clutter the software and slow down our processing.

D. Hardware Design

Our hardware design consists of reference frames that house multiple components. Thus, our main focus for the design was to make sure the components were necessary, light, compact, and comfortable. The average weight we measured for a typical pair of sunglasses is 50 grams which was our target weight. One major difficulty we experienced during design was determining the component for facial measurements. We went through three different phases.

The first phase was incorporating a caliper. After some research, we could tell they would not be an efficient implementation. They are relatively heavy and would add a significant amount to the cost. Each caliper is expensive and has its own processing board and battery supply. This would make the frames uncomfortable to wear and difficult to balance.

The second phase was incorporating a linear encoder. From our understanding, they were lighter and did not require a separate power supply, initially making it a viable alternative. However, we found they had no absolute start and end point which would require the incorporation of a regularly occurring calibration system. This is an inconvenience to the customer we wanted to avoid.

The third phase began with a change in perspective. Instead of directly measuring distance, we tried relating another measurable value to distance such as voltage. We then incorporated a potentiometer which can change its resistance. They are light, easy to move, have absolute start and end points, and don’t require their own power supply. Thus, we structured our hardware design around potentiometers.

![Figure 2: Block Diagram](image)

Figure 2 shows two blocks, one representing hardware and the other representing software. The right side is dedicated to hardware and its different components. Starting from the top, we have a block for the potentiometer. Our design includes two linear potentiometers which are wired to the microcontroller. Their purpose is to capture the dimensions of the user’s head. Potentiometers are devices that have a user-controlled resistance across the device. Most have a rotary knob, but ours has an indicator that slides from one side to another, changing the resistance. This subsequently changes the measured voltage. Our potentiometers work by having three pins. The first pin connects to the voltage source, the second pin is where voltage measurements occur, and the third pin connects to the rest of the circuit. In our case, we connect the first pin to the power supply, the second pin to the analog input on the microcontroller, and the third pin to common ground.

The second block is the microcontroller. Our design is currently using the AtTiny85. We chose this because it has enough pins for the potentiometers and the Bluetooth module, but also works within our voltage supply. It has the following objectives:

- Runs an algorithm to convert the voltage across the potentiometers to distance traveled in centimeters.
- Provides voltage nodes for a common V(cc) and common ground.
- Programs the Bluetooth component to organize the data to be sent and under which conditions the data should be sent.

The Bluetooth component we initially chose for this project is the HC-06 Bluetooth Module. We choose this component because it provides the best range at its power consumption levels when compared to other Bluetooth modules. However, based on our research we discovered that the marketed voltage on batteries is not indicative of the voltage output in real life.
As shown in Fig 3, at the best case scenario, with a miniature current draw of 10 mA, the peak voltage still does not reach the marketed value of 3.3 V. It also does not reach the marketed capacity of 240 mAh. With our design, we are estimating about 25 mA of current draw in normal use. If we round up to 30mA, the peak voltage is about 2.6 V then almost immediately drops to about 2.5 V. It then decreases slowly until it runs out of capacity. The Attiny85 is rated to work at 1.8 V but our Bluetooth module, the HC-06, requires a minimum voltage of 2.8 V.

Moving forward, we are looking at multiple different Bluetooth modules, that will fit within our voltage specifications. We do not want to change the battery due to weight considerations. A couple modules to test are the HM-10, which can run at 2.5 V, and the AMS001, which can run at 1.8 V. These low power receivers compromise with signal strength, but the typical use case scenario for the frames should not pose an issue.

Two things we need to test moving forward:
1. The power requirements of the PCB and all the connected devices, and calculating how long it should last on battery power. We can test these by connecting all of the devices on a breadboard and running power consumption tests.
2. The workable distance between the Bluetooth module and the connected device. This can be tested by varying the distance between the connected Bluetooth module, and the connected device.

E. Reference Frames Design
Our main target for our wearable apparatus is that they are safe, comfortable, durable and cheap to manufacture. The main purposes of the reference frames are to precisely measure the clients head breadth to the millimeter and provide landmarks for our facial recognition software to enhance the positioning of the augmented reality display.

F. Materials
Durability and price are the key points we considered when deciding what kind of materials are the best way to create our prototype. It has to be durable so that it can be used multiple times without breaking and without negatively affecting the experience of the users. Our first choice was to have the frames made in aluminum since that is what most of the glasses are being made out of today. Aluminum is cheap, light and much easier to work with than any other metals. However, metals are prone to bend which could lead to inaccurate head breadth measurements and would be difficult to return to its original shape if they were bent too far. It would also need multiple steps to shape it to the correct design during manufacturing.

After further research and consideration of the materials accessible to us, we decided to use plastic and have the frames 3D printed. The software we used to create our designs is Autodesk Fusion 360. It has more than enough tools that we can use to create a practical, comfortable and ergonomic pair of reference frames.

G. Design
The reference frames are 3D printed in four different parts. One piece for the right temple and bridge, another piece for the left and two adjustable ear hooks. They are held together with friction and are tight enough that they will hold their measurements when being worn and loose enough that adjustments can be made with ease. They will contain two linear potentiometers on the right temple which measure the head breadth length and the temple length. The PCB is located on the left temple and the batteries on the other side to distribute weight and increase user comfort.

---

### Table 2: Design Expenses for a Single Product

<table>
<thead>
<tr>
<th>Part</th>
<th>Value ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plastic Skeleton(PLA Filament)</td>
<td>-0.50</td>
</tr>
<tr>
<td>Printed Circuit Board</td>
<td>2.00</td>
</tr>
<tr>
<td>Bluetooth Module</td>
<td>10.00</td>
</tr>
<tr>
<td>2x Linear Potentiometer</td>
<td>3.00</td>
</tr>
<tr>
<td>Battery (with battery case)</td>
<td>5.00</td>
</tr>
</tbody>
</table>
H. Measurements

The reference frames are capable of measuring the human head breadth length and the distance between the back of the ear to the front of the face. These measurements are important in making the perfect fit spectacles. It is gender neutral and made to fit the majority of people’s heads. This prototype is made specifically for adults and has a head breadth length compatibility range from 144mm to 164mm. The 20mm limit is due to the maximum range that the linear potentiometers provide. Based on research, the average human head breadth is 148mm and ranges between 140mm and 165mm. That means our design will support most of the people in the world. Most of the eyeglasses today have a temple length of 120mm and our prototype can be adjusted from 115mm to 135mm.

![Fig. 6: Initial reference frames dimensions.](image)

III. PROJECT MANAGEMENT

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Value (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android Application</td>
<td>100</td>
</tr>
<tr>
<td>Facial Recognition and Tracking</td>
<td>100</td>
</tr>
<tr>
<td>Application Communication</td>
<td>90</td>
</tr>
<tr>
<td>Hardware Design</td>
<td>100</td>
</tr>
<tr>
<td>Circuit Design</td>
<td>100</td>
</tr>
</tbody>
</table>

After PDR, our group needed to reassess the product to make it more viable for businesses as the original implementation along with the reference frames were not attractive enough to invest in. However, through great team synergy, we were able to convert the advice given by the evaluators into the current system and make impressive progress even with the initial setbacks.

In order to do so, our team required great communication and the capitalization on each member’s skill set to optimize output. Although all of us are computer systems engineers, we have our own interests and backgrounds that made the selection of our tasks much easier. Taehwan has experience in Android application development and is the front end developer for MirrAR. This includes programming the full interface along with designing the models that will be shown in real time. John is also interested in software development and is the back end developer for MirrAR. He mainly handles the facial recognition software and algorithms. Kevin’s specialties lie in embedded systems so he is handling the custom PCB and hardware design. Lastly, Jyerrmee also specializes in hardware thus handling the reference frames designing.

We have accomplished in delivering almost all of our MDR goals excluding the application communication due to an error in the ordering process which we are currently solving with Franz. We have developed an Android application that has a fully functioning user interface which we are able to test with an emulator. We also accomplished facial recognition and tracking software that identifies a user’s facial features. We’ve completed and printed our first generation of reference frames which uses a potentiometer to display measurement values. For MDR, we used an Arduino as our PCB, however, we have finished the design of our custom PCB and are ready to send it out to have ready for next semester.

IV. CONCLUSION

MirrAR is progressing as planned through the utilization of a Gantt Chart, however, we also know there is much work to be done. We have scheduled our project in a way that allowed MDR to be the point where all of our groundwork was completed. This means circuit design with all of the necessary components, a pair of reference frames to test the functionality of the hardware, an application that displays all functionality, and facial recognition software ready to be imported into the application code. Moving forward, we plan on working towards an actual functioning prototype that can be advertised on the market. In terms of software, this includes fully functioning Bluetooth communication that allows the reference frames to send its measurement values to the application. The 3D models will also display on the application in real time along with additional landmarking/tracking to provide a more accurate projection. For hardware, the second generation of reference frames will include all the components including the battery, potentiometers, and custom PCB while meeting the requirements we have laid out. This means they will be safe, lightweight, and compact.

We believe the main difficulties will lie in creating accurate 3D models while taking into consideration memory limitations as well as achieving angled movement as well. However, assuming the CDR deliverables have been met with remaining time, we hope to focus on styling and different functionalities for the application along with attempting to
make the reference frames wireless.

**ACKNOWLEDGMENT**

Our team would like to take this time to thank our advisor, Professor Tessier, for all the advice, insight, and guidance he has provided for our project. We would also like to thank our evaluators, Professor Pishro-Nik and Professor Jackson, for their advice as well.

**REFERENCES**


