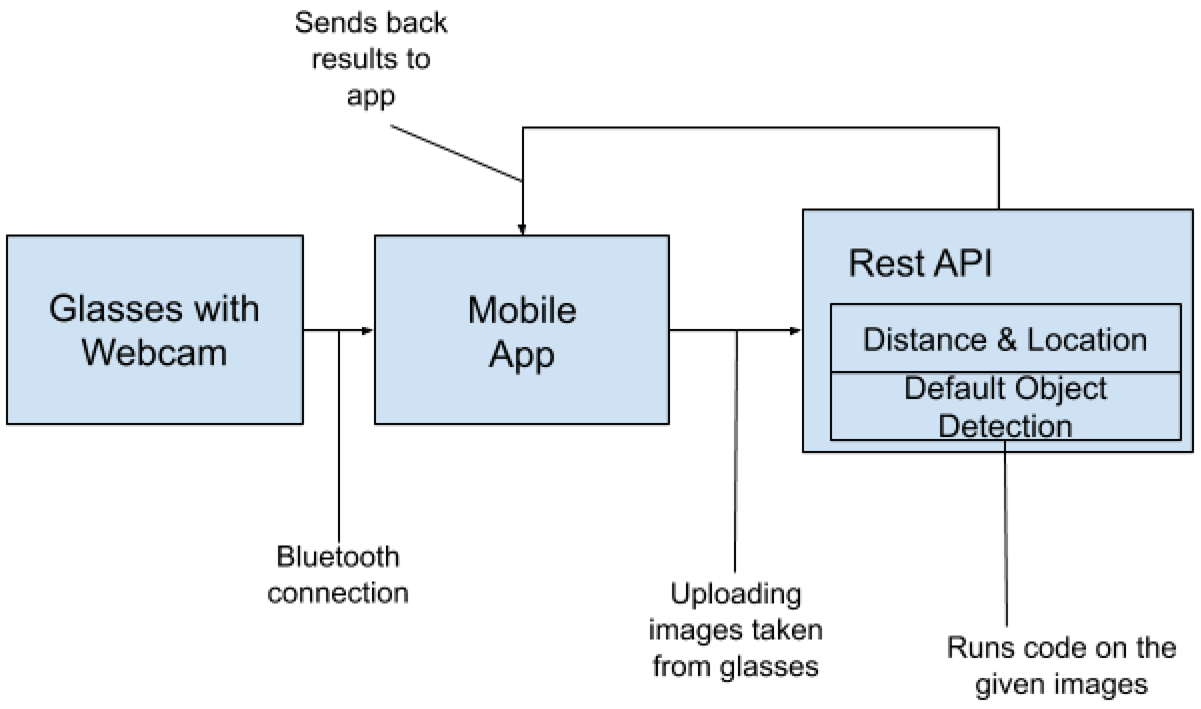
INTRODUCTION

A hackathon that I attempted to attend this year had a theme of civics. That meant that I would have to create a project that would help out with the community. I eventually came up with a project idea that would help blind people find their way on a street. The problem was, I didn't know how I was going to work it out. I had no knowledge or previous experience with computer vision, and finishing this on time seemed far out of my reach. I still tried my best, so while I wasn't able to turn in my project in the hackathon, I continued working on it because I became interested in learning more about AI and computer vision. This repository will be about my project idea. Everything included here, I have found on the web and learned on my own.

OBJECT

The main goal of my project is, like I mentioned, to help blind people find their way on a street. This means that my project should act as a third eye by scanning the surrounding and alerting the user if they are going to bump into something. The basic framework of my project would look like this:



The reason why I went with the API approach for this project is because I wanted to make the hardware easily carryable without many parts connected to it.
There were two other approaches that I could have made, but decided not to do because of how inefficient they were. One was to store everything in the glasses itself. This means that there

would be no mobile app connected to the glasses, and no use of API. Everything would be in the hardware, but the problem with this would be that the hardware spec must be really good. It also is very impractical because the glasses would contain so many things, making it heavy and uncomfortable. Another approach was to remove the API and store the code in the app itself. This is also not the best option because processing everything on the phone will consume more energy.

The webcam, attached to the glasses, will capture still images every 2 seconds. The images will be transferred from the glasses to the mobile app, which will upload the images to the rest API, where the code will run on the given images. The Default Object Detection will detect objects using a premade detection model, and the Distance & Location will find the location of the object in relation to the glasses. The outcomes will be sent back to the app, which will alert the user of what is in front of them and the possibility of them running into it.

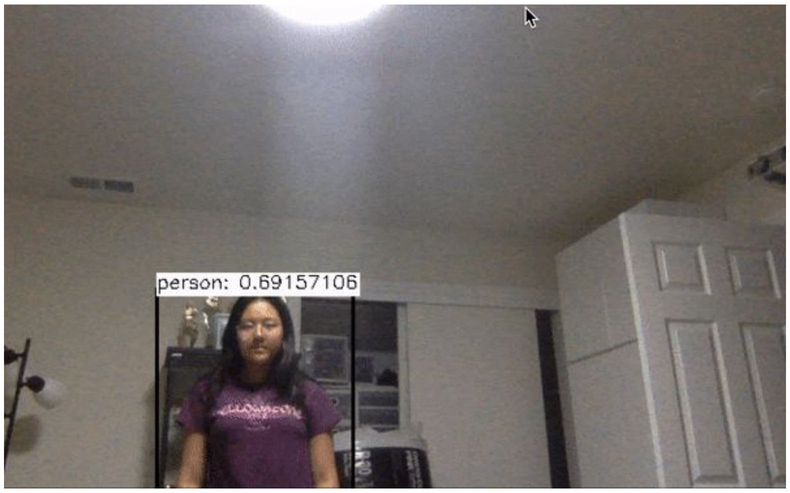
I used Anaconda, OpenCV, and Terminal on MacOS. My main language was python.

CHALLENGES

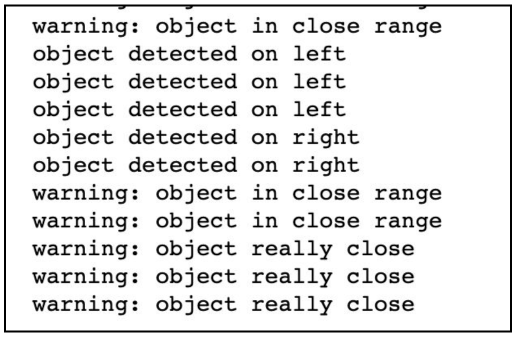
There were multiple challenges I came across while working on this project. One was setting up the environment. It took time to understand what I needed for this project and how I would set it up. I also had trouble finding the necessary resources. Another challenge was finding a way to constantly receive images from the webcam. Because the detection works on images and not live streams, I needed to find a way to capture the latest images from the live stream of the webcam. My solution was to use the imutils repository (shared by jrosebr1) and import VideoStream. My last problem was finding the distance. At first, I tried to find the distance of the object from the glasses, but in order to do that, I needed the dimensions of the object. That was a problem because the object wasn't a set thing and could differ variously. I had to scratch that idea and instead find the position of the object in the image. While it is less accurate than finding the distance from the webcam, it was the best I could come up with.

CONCLUSION

After running my code, this is what comes up:



And the code prints:



```
warning: object in close range
object detected on left
object detected on left
object detected on left
object detected on right
object detected on right
warning: object in close range
warning: object in close range
warning: object really close
warning: object really close
warning: object really close
```

WHAT'S NEXT

I want to be able to use a custom object detection model for my project, and I also want to be able to complete all the parts to this project. I'm interested in using arduino to create the glasses, so focusing on the hardware part is also on my to-do list.

REFERENCES

Git Repository:

https://github.com/jrosebr1/imutils.git
https://github.com/chuanqi305/MobileNet-SSD.git
https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10.git
https://github.com/tzutalin/labelImg.git

https://github.com/tensorflow/models.git

Blog Post:

https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/
https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606
https://gilberttanner.com/blog/installing-the-tensorflow-object-detection-api
https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/
https://towardsdatascience.com/creating-your-own-object-detector-ad69dda69c85
https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/

Application and Library:

https://docs.opencv.org/master/d9/df8/tutorial_root.html
https://www.anaconda.com/products/individual
https://stackoverflow.com/
https://www.tensorflow.org/
https://scikit-image.org/
https://www.geeksforgeeks.org/

Document:

https://pdfs.semanticscholar.org/afb1/81d8ffa99d2f381edc40a165013ad669a9b0.pdf
https://arxiv.org/pdf/1705.04114.pdf
https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html
https://www.learnopencv.com/image-recognition-and-object-detection-part1/

Online Class:

https://www.linkedin.com/learning/opencv-for-python-developers/welcome

(Coursera)

What is Data Science?
Introduction to Artificial Intelligence (AI)
API Design and Fundamentals of Google Cloud's Apigee API Platform (with Honors)