Ethan Owen
2/4/2024
EE113DW Status Report 3
Gesture Control for Home Devices Project - (solo project)
https://github.com/Ehanlion/HGR_project

Main Intention:
This week was spent improving upon the faster detection program that I implemented last week. While that program saw a lot of improvement and learning from the prior weeks, it still lacked speed and the code that drove it was poorly written, messy, and disorganized. This week, I rebuilt that code, experimented with new methods of detection, and ultimately rewrote that 'fast' code into a smoother, more operation program. This is a critical step because I need this project to be both effective and robust, with an emphasis on reliability. My code before was not reliable so this week was spent fixing that. I spent time researching better ways of performing the detection, ultimately to no avail, but the process was important since it allowed me to confirm that the current approach I am taking seems like the best. This is an excellent time in the development process to perform this sanity check because it is not worth developing a project that is not being built in an efficient manner.

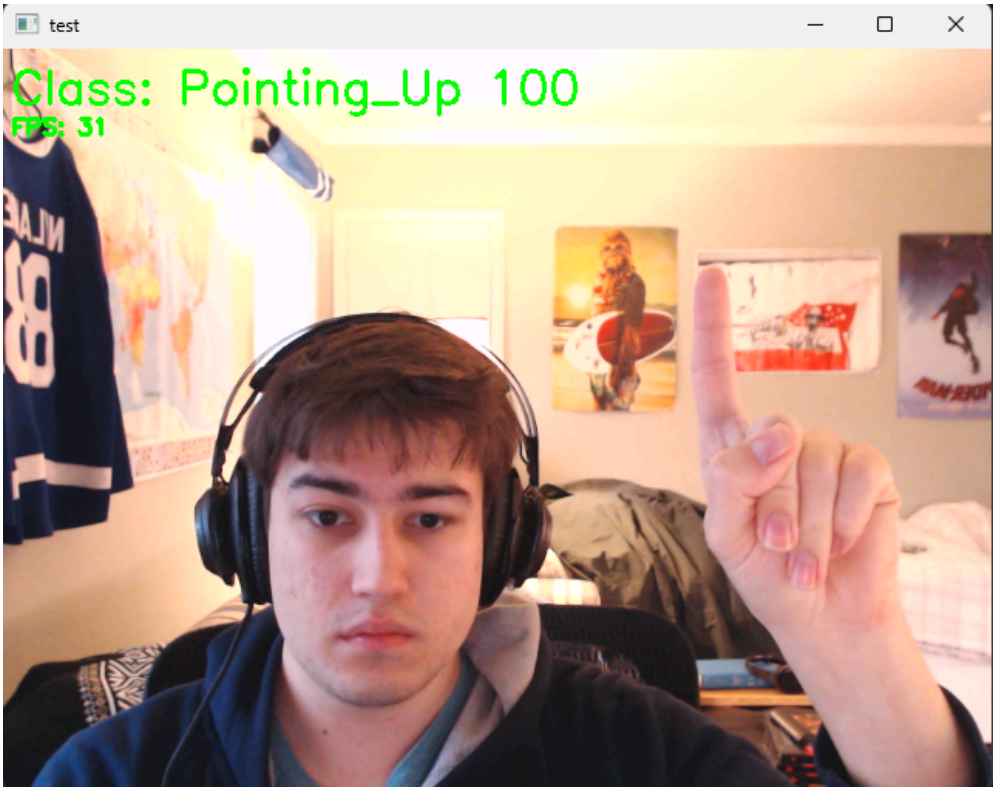
Conceptual Development:
This week's research was less involved than prior weeks. My goal at the end of week 3 was to have coded a working recognizer and to have tested it against different backgrounds. Both of those tasks were accomplished last week, so I was running ahead of schedule.

With this in mind, I researched ways to improve the code approach I was taking in performing the gesture recognition from last week. My prior approach used poorly coded global variables and had their values set via my recognizer's callback function. This was the way the MediaPipe tutorials showed it being done and, for the time being, it was working. However, this method was sloppy to me, so I looked into ways to use a different mode of MediaPipe to perform the same task. MediaPipe, when doing gesture recognition, runs in three modes: Image, Recorded Video, or Live Stream. I was using Live Stream since I was live streaming from my webcam. I disregarded the Recorded Video mode since it didn't fit my needs, but the Image mode I overlooked. I realized that it might be possible to simply parse each frame individually through the Image mode which would greatly improve my code's structure and localization. However, despite my enthusiasm, this method proved to be incredibly slow and inefficient so I scrapped it.

The other large thing I looked into this week was the library known as 'Asyncio.' It is a library for python dedicated to asynchronous programming. This is required because the in-built function for MediaPipe's live stream gesture recognition mode is asynchronous and to improve the call back function it uses, I would need to use asynchronous programming. I learned about asyncio and realized that I could write a class to encapsulate the recognizer I was using and then reference the class instead. This would solve some of the messiness I was seeing from global variables and make the code more production-quality. This would be a great avenue to explore in future weeks if I have time, since I ended up being unable to code this method this week.

Implementation and Development:
This week I improved on last week's model driven classification and slimmed down the code.

Image 1: Right Hand Detection



Notes: Gesture shown in top left with confidence next to it. Fps shown below, massive improvement over last week's fps.

Image 2: Left Hand Detection

Notes: Gesture shown in top left with confidence next to it. Fps shown below, massive improvement over last week's fps.

It is worth noting that, when comparing last week's program to this program, the feature map overlay has been removed. Drawing the features onto the hands dropped the fps to about 30% its actual value and the overlay won't be necessary to processing, it was purely visual. Thus it made sense to remove it in the name of efficiency. Of course, when this program gets migrated to a device (RPI5), there will be no video output so the processing speed will increase even more.

Resources:
I continued to use online resources from both OpenCV and MediaPipe to advise me on the project. However, this week I spent a lot of time just changing the code structure in my existing program, removing excess processing, and building in subtle new features.

Individual Contribution:
This project is an individual project and as such, all development was undertaken by me.

References:

1. https://github.com/googlesamples/mediapipe/blob/main/examples/gesture_recognizer/python/gesture_recognizer.ipynb - Website for MediaPipe with an installation guide as well as a python example.
2. https://docs.python.org/3/library/asyncio.html - Website for Asyncio library
3. https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer/python#live-stream - Website for MediaPipe with a python implementation of a hand tracking overlay.
4. https://docs.opencv.org/3.4/dd/d43/tutorial_py_video_display.html - Website for OpenCV's example on getting as video feed functioning