Ethan Owen
1/28/2024
EE113DW Status Report 2
Gesture Control for Home Devices Project - (solo project)

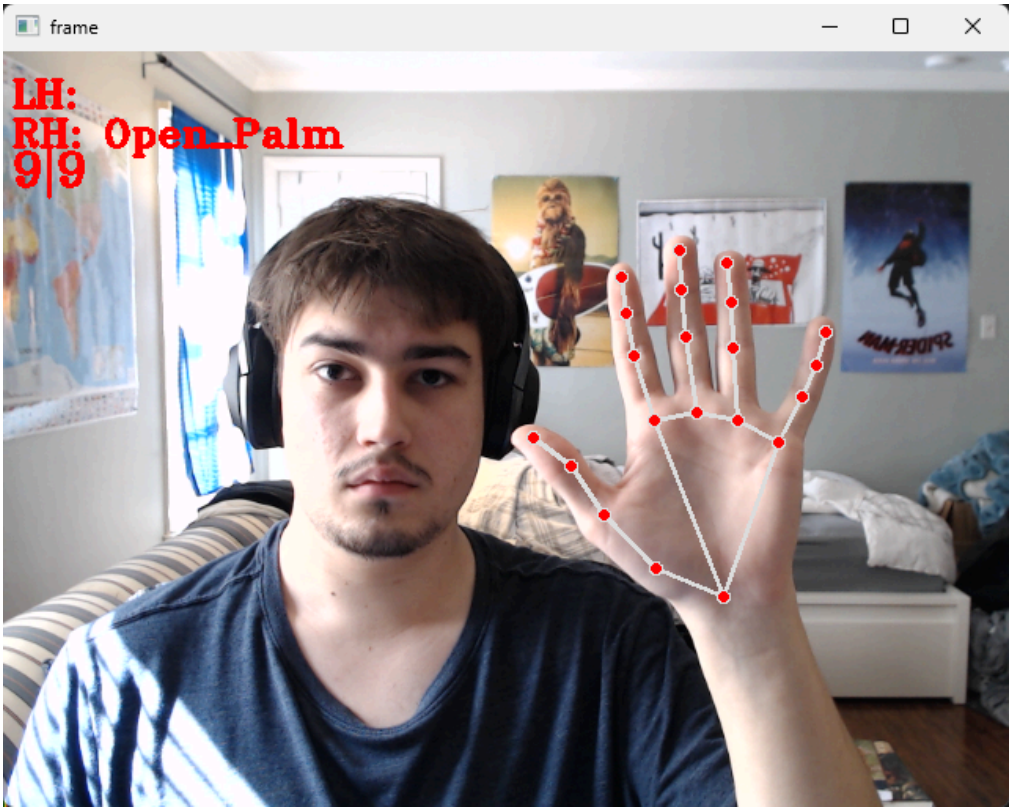
Main Intention:
This week was spent improving upon the naive detection program that the first two weeks were spent researching and building. In the naive detection program, there was no 'classification' of gestures, only the drawing of the twenty point feature maps that the MediaPipe package provides. This week, I researched, downloaded, and used a model from MediaPipes (trained by Google), to implement real-time gesture classification. This melds both the overlay from the feature maps with the real-time classification of the gestures themselves. This is a huge and critical step in building this project; having classification working properly is the keystone to timely and contiguous project development. Since everything is built on the back of classification of gestures, this step solidifies progressive development over the next few weeks.

Conceptual Development:
Over the last weeks, I researched OpenCV (a computer vision library) and MediaPipe (a package with in-built gesture recognition capabilities). These two packages are instrumental to the project. This week I delved deeper into researching MediaPipe. After digging around, I found out that MediaPipe has a gesture recognition model that the developers at Google have trained. This is hugely good news because one of the most difficult and error-prone parts of using a model is training the model itself. This basic model can recognize 7 gestures: unrecognized, closed fist, thumbs-up, thumbs-down, pointing up, victory, and I love you. The model can be used relatively simply as well, requiring it to be downloaded and loaded into your code with a gesture_recognizer object being configured with the model. There is another key importance to this model thought: you can train your own model. Yes, that's correct, you can train your own model using the MediaPipe framework. While not something I am eager to do just yet, this leaves the door open for a more comprehensive set of recognizable gestures to be used by my program. I think that, if development concludes early, training a new mode to recognize more gestures could be an interesting advancement to the project.

Now straying away from code-research, I also begin to learn more about raspberry pi computers. My eventual goal is to have my code run on a RPI rather than on my desktop. However, even with my limited exposure to my programs running, I have realized that the program is quite slow. This means I will need a powerful and quick RPI to be able to keep up with the detection in real time. I found that the RPI 5 could be a suitable candidate, if not for it being expensive. However, I believe it is necessary for this project to be successful, so I purchased one to begin experimenting with in the next weeks of development.

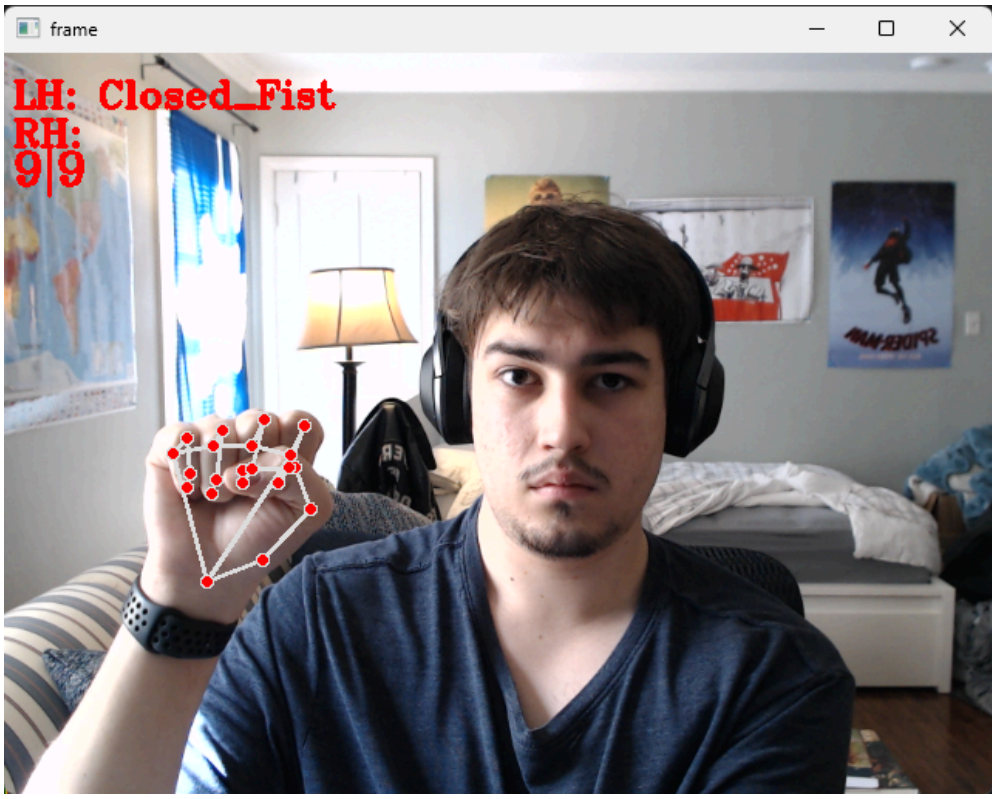
Implementation and Development:
This week, I continued to use OpenCV alongside MediaPipes but this time, using a trained model instead of a naive approach to add classification as well as feature overlay. I use OpenCV to open a video camera instance, then using the model from MediaPipe, I process each frame individually and classify the gestures within it. This result is then overlaid, alongside the twenty point feature map of the hand, back onto the original frame and is then displayed.

Image 1: Right Hand Detection



Notes: This image shows detection of the right hand and the program correctly determined that I am holding up my hand in the ‘open palm’ orientation.

Image 2: Left Hand Detection



Notes: This image shows detection of the left hand and the program correctly determined that I am holding up my hand in the ‘closed fist’ orientation.

Resources:
To code this new variant of my detection program, I continued to use OpenCV and MediaPipe. I utilized example programs detailed in the documentation of MediaPipes and did not require an example from OpenCV because I already had a function program built that utilized the components of OpenCV that I wished to use. I realized this week that the documentation that MediaPipe has is pretty terrible; their examples were actually incorrect for this part of the project and I had to make divisive and sweeping modifications to how the code was structured in order to get it to work properly. I guess that is the point of doing a new project. This week, I also began backing up my files to my GitHub in order to allow development both on my master desktop as well as my work laptop. My GitHub is linked in the 'references' section.

Individual Contribution:
My team only has one person on it, so everything I detailed and worked on encompasses the entirety of development up to this point.

Satisfaction:
This week went better than I had planned. Now that I have a working gesture classification program using the pre-trained model from the MediaPipe package, the foundations of my project are complete. I fully believe that it will be possible to complete the project early and even implement features not detailed in my initial project proposal. I think that I can work at my current pace and the project will be completed in a timely manner.

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://developers.google.com/mediapipe/solutions/vision/gesture_recognizer/python#live-stream - Website for MediaPipe with a python implementation of a hand tracking overlay.
3. https://docs.opencv.org/3.4/dd/d43/tutorial_py_video_display.html - Website for OpenCV's example on getting as video feed functioning