Kevin Whitney

Design Project 1

Entry Log – Monday 12/17/2018

Over the past week, I have been working solely on learning OpenCV at a deeper level, as well as beginning work on the core foundation of what my overall project will be. That being the ability to get a live input feed from the webcam and locate and track the users hand movements. As much as this can be done in a small amount of code, the amount of possible ways that one can do this is fairly high.

The first approach I took this week was simple subtraction of two images. What I mean by this, is I had the program take the first frame upon starting and save it, where I would prompt the user to let them know to keep the ROI (the region they must have their hand in) empty. Once the frame is taken, it is then compared to every new frame and the initial frame is then subtracted from the new frame. Therefore, I am able to see only the changes to the frame (i.e. the hand, as it is the only object not in the original if the camera does not move). This works in theory and did have results I could use. However, the results were not always perfect, and the slight move of the camera would mean you would need to reset. I wanted the user to be able to bump their desk and still be able to continue. As well, laptops would have issues with this as they are not perfectly still like a webcam at a desktop may be.

I next tried to segment the hand in each frame using the color of the user’s face. The user’s face is easy to track using the built in Haar Cascade available with OpenCV. The plan was to then use k means clustering in order to find the dominant color of the user’s face, and then segment the image of each frame based on that HSV array (with a buffer in each direction) so that I could show only the user’s skin areas (which only the hand would be shown in my ROI). My issues with this were as follows:

1. I didn’t give myself enough time to learn or understand k-means clustering, so I wound up confusing myself and messing up my code.
2. If a user’s hand was much lighter than a user’s face, which is completely possible, the program would not work as efficiently.

Thirdly, and where my code is now, I am using an HSV array that roughly contains most user’s skin tones to segment each frame. This is not how I plan to keep the program, as I would rather have this value be created dynamically (the results are not amazing with the current value). Using this value, I first create an ROI where the user will see on their screen a white rectangle. This is where they will place their hand when gesturing. The frame by frame video feed from this is then converted to the HSV color space. A mask is then created on this frame using the lower and upper limit that I am using for testing. This then shows in white, anything in the original image that is in the HSV range. Any other pixel is black. This is the image that I will be using to recognize the hand gestures. Although I am currently getting workable data with this method, it’s not perfect. Moving forward this week I want to work the items below:

1. Clean up the noise on the output I am currently getting.
2. Make the upper and lower limit of the mask dynamic, so it is based on the user who is using it.
3. Continue learning TensorFlow with fake data, so that I am ready to jump in when the hand tracking is complete.
4. Have (hopefully) the entire translator aspect done by the end of Christmas break. That leaves me plenty of time after the holiday to turn this in to a full application. This is doable if I continue working diligently.
5. Begin uploading code to Git tonight, as it is now getting towards the end of testing and playing around. At this point, I need to decide on a method of image segmentation and stick with it.
6. Dedicate Tuesday morning to attempting to further understand k-means clustering. Using this to get the average color of the user’s palm could be the way to get the dynamic HSV range.