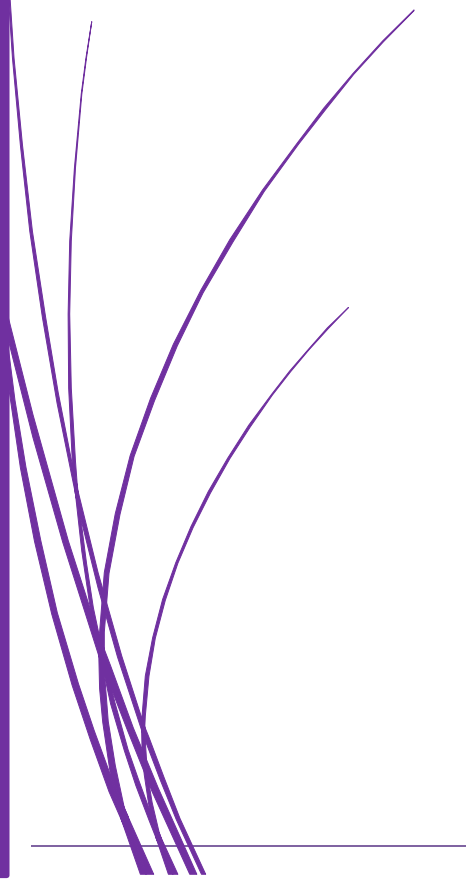
Tandon School of Engineering

Department of ECE

DSP Lab Project Report

Real-Time Gesture Recognition



Aditya Mohana Sivaraj (ams1803)
GRADUATE STUDENT - ECE

Real Time Hand Gesture Recognition.

1. Introduction:

With the advent of transistors technology has developed in speeds unbeknownst to mankind ever before. Since the introduction of smart home appliances, as far the average consumer goes, the only way to control them other than the traditional way is through voice/sound commands processed by a commercial voice assistant like Google Nest's Google Assistant, Apple Home's Siri, Alexa, etc. But just like the traditional way where one must search for controls and move there, voice control is not practical in many situations like places where quietness is important, or places too noisy for the processor to hear one's voice. And hence, there are other methods being investigated in the industry. Gesture recognition is one such method being looked into, and hence as a project I have a developed a concise module to recognize a set of four gestures to control music locally. This module can also be used to be integrated with other commercial platforms like Phillips Hue lightings, which, provide their own easy to use APIs, and Sonos Systems – an audio equipment manufacturer who also provides their own easy to use APIs called sonos, and almost all the commercial music streaming services provide their own APIs. I initially integrated Google Music API with this code, but, since it required its users to enter their credentials and I am required to submit the code for evaluation, I could not do it. And, on the topic of the ideas that failed on this project, others were a plan to implement this on Raspberry Pi platform and that did not work out because the graphical processing power required for this could not be supported on a small platform like Raspberry Pi, algorithms using only Image Processing by OpenCV have been implemented, but, they were not as accurate and consumed a lot of computing power as long as they were on, the other plan that failed was to implement a newly published research on this topic, which, used two Convolutional Neural Networks, one is a light weight network for detecting if the pre-trained gestures had been performed, and the other for classifying the gestures which runs only when the first network detects a gesture since it is computationally a very heavy network, but, their module is built keeping only GNU based systems in mind and would take a lot of modifications to be able to run in other environments, but nevertheless, for my paper report I have reviewed their paper, and finally I tried adding voice command control to the module but that was too demanding computationally, but perhaps, it may work more efficiently in systems that is solely dedicated for this purposes, but, it couldn't be implemented in a PC.

2. Libraries Used in this Project:

Tensorflow: A machine-learning library created by Google for training networks. Even if it's not required for running the module, since it is already trained, it is required for Keras (elaborated below).

Tensorflow-gpu: It is the same as the one before, but it can run using GPUs if the system has one, with the help of GPU modules like CUDA. This makes the processing faster. You can use either one.

CUDA: To be used only if you are using Tensorflow-gpu and have NVIDIA GPU.

Keras: This contains training modules built on top of the Tensorflow libraries and modules.

OpenCV-Python: A popular open-source image and video processing library.

Pandas: A popular data-science library, used to process the training data.

Pygame: A media library built for python, to handle the music in this project.

Numpy: An open source library to handle datasets and mathematical operations.

Matplotlib: Library to handle graphical calculations.

Pillow: To display and handle images.

The rest of the required libraries are usually included in almost all python distributions. Nevertheless, a text file containing all the required libraries with their version number is included in the folder. You can run:

```
pip install -r requirements.txt
```

3. How it Works and How it was Formed:

First, I tried using the standard free hand-gesture data set from Kaggle, but that didn't work out because, the model required the training data to be input to be in RGB format. Then, I tried copyright datasets like EgoGesture and NVIDIA Hand Gesture Dataset, but their distribution format didn't work well for me and even I tried doing with a small part of their dataset, the testing phase showed pretty bad results for my model. At this point I tried to train the network with my own images. But, that required taking pictures from various different angles and that became a pretty arduous task to do alone. Fortunately, I got a small model with just 5 gestures trained from Github.

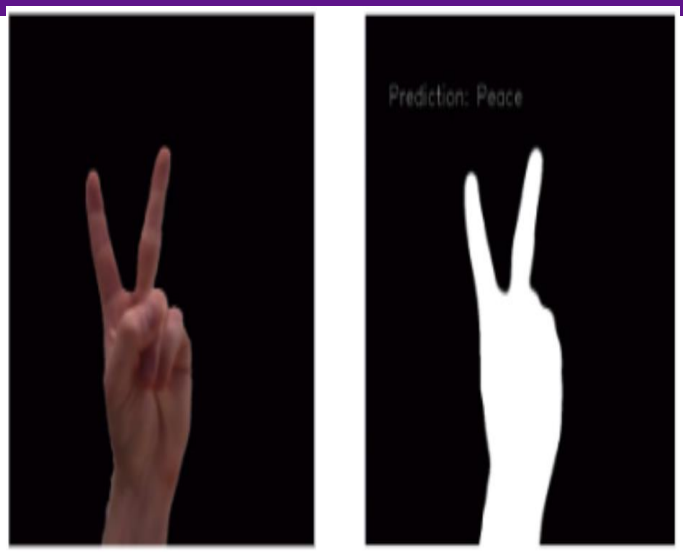


Figure 1: Example of Binary Thresholding.

Then for background elimination, a simple idea was implemented here, where once the module captures the background image without hands, then whenever a hand is placed in the region of interest, it would be able to read the hands separately by masking the background.

Then, this module makes it simpler for the computer to read by performing binary thresholding. This is a process where the pixels below the threshold are converted to full black – 0 and the ones above the threshold are converted to full white – 255.

These images are then fed to the algorithm to be compared with the pretrained model to see which class of gestures it matches with best. This is where the Keras library comes to help. The library loads the model onto the module and then with the help of its predefined functions we can compare the models inside a defined function. The prediction accuracy is also calculated inside a different function, again, using the help of Keras library.

After the classification process is done, it feeds the classification nomenclature to another function, which, determines what needs to be done for each of the classes of gesture.

In this module, pygame library is used to handle the music, since that is what we are doing as part of this module. PyAudio which was taught in class is not equipped to handle mp3 files, which, is what most music files are encoded in today.

For future development, this module is equipped with functions that can be used to further train gestures. Also, this module can be integrated with other smart home appliance APIs and music streaming services' API for further home integration.

4. Results:

The model's performance exceeded expectations. It classified nearly every gesture in the test set correctly, ending up with a 98% F1 score, as well as 98% precision and accuracy scores.

As any seasoned researcher knows, though, a model that performs well in one environment is not a guarantee for it performing well in other environments as well. Having experienced the same failure with my initial model, I was cautiously optimistic that this model would perform well on gestures in real time.