Spring 2025

EE 3530

Digital Signal Processing

Final Project: Hand Gesture-Controlled Volume

Spencer Hart

Introduction

This project focused on translating real-world, physical movement into digital feedback. The real-world object of interest was a human hand. The mediums for translating the movements of hands were image recognition and machine learning (ML). Digital feedback, namely, control of the volume of a song playing on a laptop, was produced by manipulating hand-landmark data points. The purpose of this project was to establish control over the digital world through physical-world action, in addition to gaining a greater understanding of image recognition, machine learning, and convolutional neural networks.

Problem Statement

The problem this project was trying to address was: “How can remote physical hand gestures be digitally translated into song volume control?” More specifically, “How can real-time video footage of hands be captured and processed to allow for hand recognition, tracking, and remote volume control?”

A fundamental understanding of digital signal processing (DSP) was necessary to complete this project. Any digital image is a large collection of digital signals, which, to utilize and manipulate, must be processed using often complex mathematical equations and models. Being that a video is a very large stream of digital images, DSP is essential for any type of real-time image recognition. Machine learning is the process by which image recognition is possible. By “training” a mathematical model on a set of images (which are no more than vast matrices), this model will begin to identify whether or not a newly presented image matches the previous set. Convolutional neural networks are the mathematical models most typically used for this application of machine learning, that is, image recognition. Convolutional neural networks

have three main components that allow them to operate correctly: convolutional layers, pooling layers, and fully connected layers (neural network layers).

In the first layer, the convolution layer, digital filters, like the Prewitt (edge detection) or median (noise reduction) filters, are applied to the input data—an image. The purpose of these filters is to create feature maps of the image to highlight image features like edges, textures, and shapes. After the convolution layer, the filtered data enters a pooling layer, which reduces the dimensionality of the feature map to make the model more efficient and robust to variations in the input images. Next, the data is passed into the fully-connected layers, or the neural net. These layers take the features extracted from the previous layers and use them to make the final classification decision. Classification in this project is defined as “hand”, “not hand”, “thumb tip”, “index finger tip”, “wrist”, etc. Finally, once the model has been trained on sufficient input data and the classification is complete, new images passed into the system are passed through the layers, resulting in a probability distribution of the different classes. Hand recognition and tracking will occur depending on these probabilities.

Data

The input data for this project will be a livestream of images (real-time video), mostly involving a hand in the frame. The output data will be the volume level percentage (and therefore an audible change in song volume) as well as distances between two pairs of hand landmarks and their ratios. X and Y values are normalized to values between 0.0 and 1.0 with respect to the image width and image height, respectively. This means that the scale for distance here is relative, not absolute (e.g., meters).

Evaluation Criteria

The levels of evaluation will be graded on a Yes/No scale and will be as follows:

1. **Livestream activation** - Can real-time video footage be acquired and displayed indefinitely?
2. **Appropriate video settings** - Is the video in RGB (not BGR), and is it a mirror image to replicate a “selfie” view?
3. **Hand recognition/tracking** - When no more than one hand enters the frame, is the hand recognized and tracked, even with hand movement in the physical X, Y, and Z plane (left-right, up-down, forward-backward)? (Hand recognition and tracking will be indicated in the livestream footage by the colored “hand landmarks” along the fingers and palm.)
4. **Hand-landmark data point acquisition and manipulation** - Can the hand-landmark coordinates be obtained and used to calculate finger-digit distances and their ratios?
5. **Volume control** - Can the finger-digit distance ratio be used to obtain a volume percentage and then act as an input to the actual device output volume? (Indicated visually on the device's soundbar and audibly in the music volume.)

Approach

Initially, research had to be conducted to understand how to use OpenCV, a real-time computer vision library, and MediaPipe, a well-known ML framework. There was a decent amount of information on both resources, but it had to be sifted and sorted through to obtain only relevant and useful information.

At a high level, Google's MediaPipe is a framework for creating data pipeline models. A "pipeline", in this instance, is essentially the system or code handling the input data. This data is contained within "packets", rather than a constant stream of individual data points, to improve handling and processing efficiency in the system. Here, a "packet" is defined as a frame in a video. As these packets move through the pipeline, they must be processed to obtain the desired output data. In this project, this processing is done by MediaPipe's convolutional neural networks to identify hands. For real-time pipeline processing, flow control, meaning packet loss, is necessary as a buildup and overflow of these packets occurs. Designing these kinds of pipelines is difficult, which is why Google's MediaPipe was conveniently chosen, as it had already been trained on over 30,000 images of hands.

Once the initial research was conducted and a basic understanding of how each resource functioned, it was possible to begin coding. Pertinent libraries were imported, and real-time video footage was acquired using OpenCV. The video was then passed through MediaPipe to recognize, track, and assign data points to certain regions of the identified hands.

With these data points, distances between the desired finger-digits to control the volume level could be obtained. Next, to preserve an accurate representation of the desired volume level with finger-digit distance from any distance from the camera, a ratio was necessary. Being that there is no absolute distance readily available within the video frame, the volume-control finger-digit pair distance, referred to as "dist", was compared to a second finger-digit pair distance, referred to as "base". The ratio between "dist" and "base" would stay consistent no matter how far away the identified hand was from the camera, ensuring accurate volume manipulation. The ratio was then converted to a volume percentage and fed into the volume change input.

Results and Analysis

The results of this project were a functional gesture-controlled volume system and relative finger-digit distance outputs, as well as their ratios. The system worked quite reliably and had only a few shortcomings. One of the main issues was functionality at multiple distances from the camera. Before including a distance ratio, the relative finger-digit distance to control volume, “dist”, would decrease as the hand moved away from the camera, even if the absolute distance between the digits stayed the same. This makes sense because the system merely computes the relative distance between the digits based on the pixel distance, that is, how many pixels are between the hand landmarks. The ratio method solved this issue and kept the volume level adjustment accurate from multiple distances.

Development

This system could be improved by introducing new image recognition techniques or training the model on additional images to recognize different hand formations. This could be used when trying to recognize specific cues to trigger different functionalities. For example, if the system were capable of recognizing a “thumbs-up” and a “STOP” hand formation, the system could respond by starting or playing an audio file or song, then pausing the audio, respectively.

Conclusions

This project allowed for further understanding of image processing and convolutional neural networks. It was also a great introduction to Google’s MediaPipe machine learning framework and OpenCV’s computer vision capabilities. The chosen solution to the problem

statement demonstrates a succinct and robust method for identifying and tracking hands in an image, in addition to manipulating a device's output audio volume based on the distance between two finger-digits.

Sources

- Gives an overview of what Google's MediaPipe Hand Landmark task is
    - https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker
- Gives an overview for implementing MediaPipe's hand landmark detection in Python
    - https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker/python
- Gives a more detailed explanation of MediaPipe intricacies and options with example code
    - https://mediapipe.readthedocs.io/en/latest/solutions/hands.html
- Gives an example to demonstrate MediaPipe capabilities and implementation
    - https://www.youtube.com/watch?v=Ye-lTW68pZc
- Troubleshooting MediaPipe installation
    - https://www.youtube.com/watch?v=KJepCMc0WMo
- To understand OpenCV syntax
    - chatgpt.com
- To understand how AppleScript works
    - https://stackoverflow.com/questions/45772011/how-to-change-volume-with-python
- To gain a greater understanding of convolutional neural networks
    - https://www.youtube.com/watch?v=-yGHUa5PL4A