Project #2 - Group #2

A REPORT SUBMITTED AS A REQUIREMENT FOR CSE 155: INTRO TO HUMAN COMPUTER INTERACTION

Erick Barajas
Role: Developed Hand Tracking
University of California, Merced

Miguel Martinez
Role: Developed Hand Tracking
University of California, Merced

Jose Ochoa
Role: UI/UX Developer
University of California, Merced

Manuel Toro
Role: Developed Hand Tracking
University of California, Merced

Chester Yang
Role: UI/UX Developer
University of California, Merced

April 27, 2022

Abstract

The team was tasked with creating a gesture controlled media player. This was done by creating a media player that is controlled using hand gestures. In order to do this the libraries used were MediaPipe and Tkinter. MediaPipe was used to help detect a hand and count the number of fingers being displayed, while Tkinter was used to create a basic outline and layout for the media player. When creating the code, each part of the project was developed independently and at the end they were connected together. This method of coding allowed each program to run correctly on their own; however, when they were connected together there were a couple glitches in which the media player would not load correctly.

1 Goals and Motivations

The goal of this project was to create an application that would allow its users to control a media player using only their fingers with limited use of the mouse requiring the use of only one hand. The media player's main functions would be controlled by four specific hand gestures with each one pertaining to a specific function in the media player i.e. play, next, back, and pause. Some motivations for this project included making it easier for individuals with certain disabilities to listen to their music with ease of access, using the mouse only when necessary. Another reason someone could use this application would be simply for the quality of life changes that it could bring about such as when an individual is multitasking and is utilizing their hands for something else.

2 Design Process

For the media player the team wanted to go with a simple design with light colors. Before a user starts using the media player they can click on a button up top which displays instructions as to how to control the media player with certain gestures. After, depending on the version of the player the user can either manually add and remove songs of their choice using the mouse or alternatively can simply start using gestures if the songs are already preloaded into the player. The selected songs are listed in the player and the first song is automatically highlighted to allow for immediate use of the gestures as soon as the songs are loaded in to prevent further use of the mouse.

Most of the colors used in the media player were light since going with brighter and more abstract colors can sometimes be overwhelming to look at. Additionally, the colors that were chosen were picked for a specific purpose in mind as opposed to arbitrarily. For example the media player buttons were different colors to help differentiate their function. The next and previous buttons are blue since they both served the same purpose which is to jump to different songs. Play and pause however, are green and yellow which are associated with colors of a stop light which gives the color an intuitive purpose. As for the media player itself the background is a light blue color with black text used for displaying song information. When a song is selected it is highlighted in a gray background with white text. All of these colors contrast each other in a manner that isn't confusing, but rather informative and easy to follow. The final product of the media player can be seen in Figure 1.

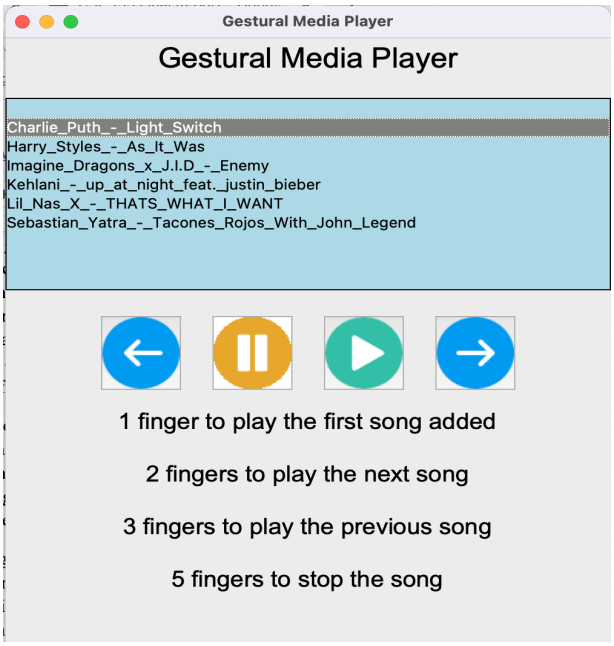

Figure 1: Media Player with Instructions

For the hand detection the team decided to use simple hand gestures that are easy to perform. Since the user would be controlling the media player with these gestures then they had to be something that would not strain the user's hand. For that reason the team decided to have the user display a certain number of fingers to perform different functions. When deciding what functions are performed by what gesture the team remembered that there is already a universal control system for most headphone/earphones. The majority of earphones are controlled by a single tap to play/pause, a double tap to skip, and a triple tap to go back to the previous song. The team used these universal controls for their media player but instead of tapping it would require the user to display that number of fingers.

This means if the user wants to start playing the selected song they would display a single finger and the song will play. In order to skip to the next song the user will display two fingers and in order to go back to the previous song the user would display three fingers. In order to stop playing music the user can pause the song by showing 5 fingers. This control was thought to be something simple since someone holding up a hand with all five fingers open, usually, symbolizes someone gesturing you to stop.

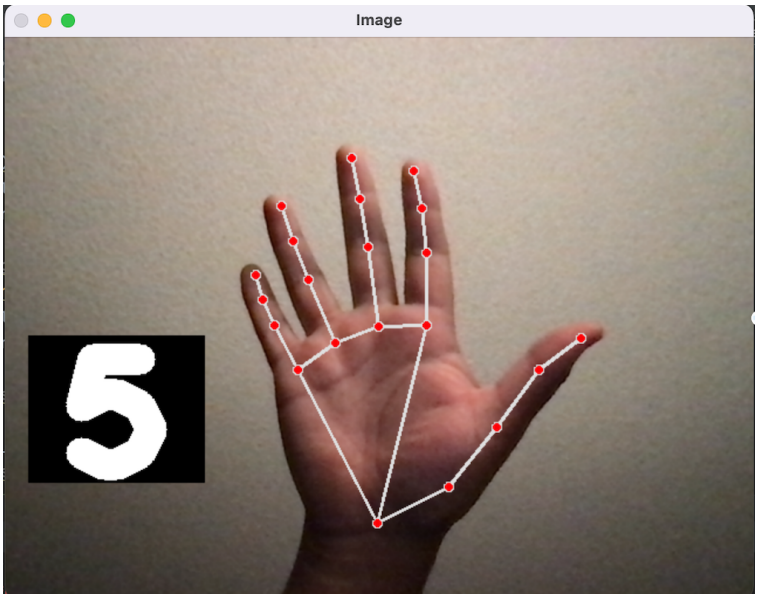

Figure 2: Hand Detection Window

In order for the user to know if their hand is being registered, the team added a window that will display the hand along with the number of fingers the program is detecting. If the user's hand is being detected it will display an outline on the user's hand and a box will appear displaying the number of fingers that are detected. **Figure 2** shows the window that displays the users hand and the number of fingers that are detected.

3 Implementation

Frontend

For the frontend a number of libraries were used which included tkinter, pygame, and PIL. Each of these libraries served a different purpose in building the media player and when put all together give us the final product. Tkinter was the library used to create the graphical user interface (GUI) that displays everything the user sees on their screen in a separate window. Tkinter gave the team the ability to create a player with custom looks being able to change many different aspects of the player such as background color, window size, font color, and many more parameters. The reason the team decided to go with tkinter since it's the only library built into Python's standard library. Additionally, tkinter is cross-platform meaning the code will run the same on either Windows, Linux, or macOS only slightly changing its look based on the device's OS. Pygame is another library that was used since it allowed for basic audio manipulation. Using pygame allows sound to be initialized through a specific command line giving accessibility to basic audio functions such as playing and pausing. These were the main functions used for this media player since otherwise imported songs would be static and overall not do anything besides display a name. The final library that was used was PIL which gives an application the ability to add image processing capabilities. In the team's case PIL was used to render and manipulate images that served as buttons that controlled the media player.

Creating the media player was done with the help of a youtube video by John Elder that served as a guide. In his video he created a media player covering what each library did in a detailed manner helping us understand how we could create our own media player using these libraries. After learning the basics of the libraries the team got to work and created their own player with their own custom parameters and looks.

Backend

The backend consists of the gesture tracking component of this project, this was done by detecting if certain fingers were open or closed. To do this, the team used MediaPipe and OpenCV. MediaPipe has a basic layout needed for tracking hands, this package provides a feature that draws landmarks all over a hand if it is detected. These landmarks are used to detect if a hand is open or closed. OpenCV allows the program to access the device's camera and lets the program use it to display the image being received by the camera. There were other options for hand detection however, all of the group members were unfamiliar with this type of project so the easiest thing to use was the option that had the most online resources. This means there were lots of videos and articles that explained how MediaPipe can be used, what functions it provided and examples of code that showed it being used.

When starting the code, the group began on deciding how they can detect a hand. This was done by using a video by Murtaza Hassan. His youtube video explained how he used MediaPipe to track the movement of the index finger and the thumb, he took the distance between them and set it as the volume for his device. From the video the team took the HandDetectionModule.py file, this file had different functions that initialized the hand detection and drew the landmarks on the Hand, and a function that finds the x and y coordinates of each landmark. To start off the code the team first initialized the devices camera window and gained access to the devices camera. Then an array is created that holds the landmarks of the fingertips that are shown in **figure 3**. The fingertips will be used later in the code to determine if a finger is open or closed. After saving the landmarks the hand detection is initialized by calling the functions in the HandDetectionModule. Here the detection function is called, this function will find a hand in the image

being input by the camera, then the findHands function is called and has an image passed through it, this image is the image of the hand. The function will read in the image and then create all the landmarks that are on **figure 3** on the hand. Then we call on the fingerCounter to find how many fingers are open and closed.

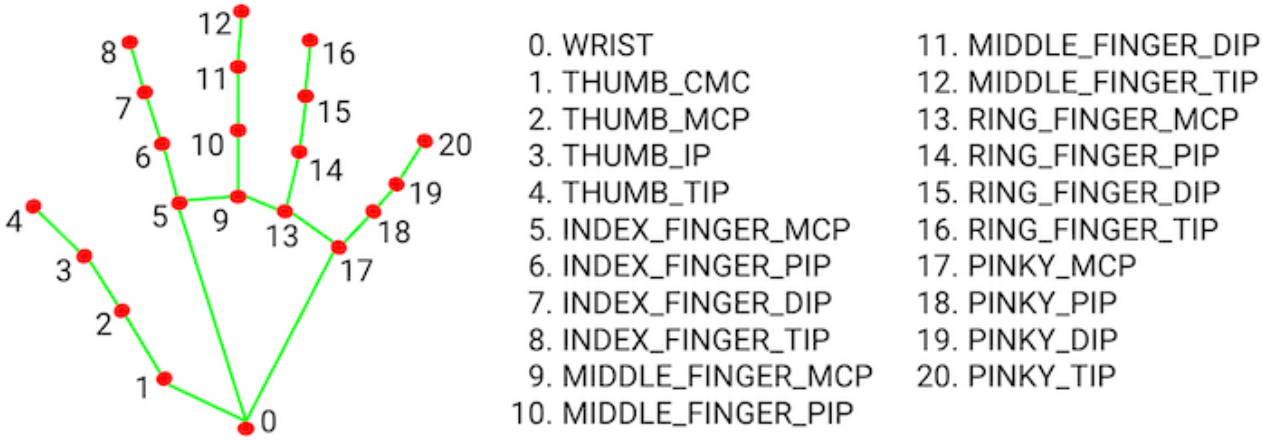

Figure 3: MediaPipe Landmarks

Using the HandTrackingModule the team implemented a file called FingerTracking.py, this file contains a function, fingerCounter, that counts the number of open fingers. In this function an empty list is created, this list will hold 5 values, one for each finger, the value can be either a 1 if a finger is open or a 0 if a finger is closed. This function takes in the landmarks at the tip of each finger [4, 8, 12, 16, 20] these landmarks are stored in a list which we go through in the fingerCounter function. The way we determined if a finger was open or closed was we checked the x,y values of the fingertips and compared it to the x,y values of a landmark that was 2 numbers under the fingertip. For example, landmark 8 would be compared to landmark 6 and landmark 12 would be compared to landmark 10. If landmark 8 had a greater y value than landmark 6 then the finger is considered to be open so a 1 is stored in the list to represent an open finger. If the value of landmark 8 was less than the value from landmark 6 then the finger is considered to be closed so a value of 0 is stored in the list to represent a closed finger. This process is repeated for every finger, except the thumb. For the thumb the same concept is used but instead of comparing it to the value that is 2 under we compare it to the landmark directly under so the landmarks used are 4 and 3. The fingerCounter function returns a filled list that allows us to check how many fingers are open or closed.

In the main file the list is tested to see how many values of 1 are stored in it. When one finger is open in the list then the media player will activate its play function, if two fingers are detected then the media player will skip to the next song. If three fingers are detected then the media player will play the previous song and lastly when five fingers are detected the media player will pause.

At this moment the media player does not completely work. When the program runs it loads up the camera display screen but the media player does not seem to fully load in. The user is able to use the hand gestures to play the first song and pause it but when they try to go to the next song it will crash because the other

songs are not fully loading in. The team worked with their TA to try to figure out what was wrong but they were unable to fix the problem within the remaining time.

4 Example Use Case

The application allows the user to control a media player using four simple hand gestures. By having the user control the media using hand gestures it allows one to multitask. An example for when this would be used is when you are in the kitchen making a meal. You would like to listen to music or skip a song but your hands are covered, dirty from the ingredients that you are using. This is an issue since you wouldn't want to touch your device like this. With a simple hand gesture you would get your music playing, skipped, or what you may require.

This application could also be used by people with certain disabilities. A person may not be able to hold onto a device or have some type of difficulty using a certain device. All one would need to do is prop a device where the user could display their hand to create the four simple hand gestures. All the user needs to do is display one hand, this way the user could enjoy by playing, skipping, going back, and pausing the song.