



**University of Arkansas – CSCE Department  
Capstone II – Final Report – Spring 2023**

**Gesture Controlled Audio Software**

**Anusha Bhattacharyya, Winston Phillips, Maxx Smith, Troy Watts**

**Abstract**

By training AI models in video understanding of action recognition, our plan is to implement efficient gesture recognition and control to manipulate audio parameters for an increased uniqueness and sound-design usability in audio production. With deep learning, a type of machine learning that mimics the way humans are able to learn, the main objective is to use video input to perform a key-point based inference paradigm.

Using a key-point based inference recognition, our model takes video frames as input and isolates points on the hand to identify when an action is being performed such as pointing, waving, giving a thumbs up, etc. Based on the gesture, we can connect this with Spatial Audio Editing and play different sounds. The applications include live music without need for instruments, more personal and intuitive audio production, engaging presentations, enabling physically-impaired individuals, and more. The ability to control an audio workstation with your body will create a personal connection in the production process, making it more an artistic instrument as well as a production tool.

**1.0 Problem**

There are times where interacting with an audio system or a smart device can be difficult, like when cooking or in a noisy environment. Being able to have a gesture based system would be helpful in this situation. Gesture controlled audio software applications could be revolutionary in live music making and audio production, more specifically in editing, automation, mixing, and mastering of tracks. This includes anything from orchestral music to DJ music, where music can have many different components to consider and the artist only has a limited number of hands. Many times when interacting with an audience, demonstrations are important for their attention. Implementing this technology would allow for more engaging presentations be it for a business meeting or for a one man magic show. Additionally, this could be utilized in theater productions in order to reduce the sheer amount of equipment backstage. Behind the scenes in theater productions involve many technical components which typically require a large team to manage. Having a gesture-based audio system would allow for a more adaptable technical setup, less equipment to manage, and less potential factors that can go wrong.

Gesture Controlled Audio Software

This software application could also be used by the common public as well, such as physically-impaired individuals that may not be able to easily operate an instrument. For example, while someone may not have the strength or mobility to use a heavier instrument (like a standing bass or a set of drums), this application could allow them to continue pursuing their passion without needing nearly as much physical strain. On top of this, it could make transitioning to a new instrument remarkably easier. Instead of relearning how to play specific notes or chords, the individual can use the same hand motions with any “instrument” the program is set to play. It will also help lower the bar into the world of music, as it can drastically reduce the costs of playing when compared to buying traditional instruments. This application will enable a much wider audience to enjoy playing music in a way not previously possible, and at a much lower cost.

## **2.0 Objective**

The objective of this project is to design, prototype, and build an application that can take in a stream of video input and turn that into audio output depending on what gestures are being made using an AI-operated algorithm. The input should be processed immediately upon being taken so that the user is able to hear the corresponding output for their gestures in close to real-time.

## **3.0 Background**

### **3.1 Key Concepts**

- Key-Point Inference Algorithm/Paradigms

This model is able to learn what a good key-point is defined as and should be able to capture key-points on multiple scales. Upon being given an image, it will use a fully-convolutional recursive network to output a series of key-points. This involves giving images (in this instance images of hands) in order to be trained on well identified key-points.

- MIDI (Musical Instrument Digital Interface)
- DAW (Digital Audio Workstation)

A Digital Audio Workstation is a production application, primarily used for recording, producing, and editing audio. Most DAW's will operate on a main user interface and have multiple plug-ins, or Virtual Studio Technology (VST), to handle individual editing tasks, such as equalization (EQ), gain, compression, side chaining, etc. EQ is used to manipulate audio files frequencies, gain is the level, or volume, of a track, and both compression and side-chains are used for creating a shared and equal sound design.

### **3.2 Related Work**

There are several research projects that have been conducted in relation to producing audio using gestures, though the majority of them are much more hardware based, rather than video input based. To be more specific, most of the gesture systems that mimic instruments are created using custom-designed gloves that incorporate some element of hardware. In a similar project by Pranshu Jhamb and Arvind Rehalia, they refer to these gloves as “transmitter-I” and “transmitter-II,” an important distinction to make since each glove has separate functions. Their

## algorithm for gesture control was based on the Inertial Measurement Unit (IMU) gesture of Gesture Controlled Audio Software

transmitter-I, the bend measure of the user's middle finger on transmitter-I, and the IMU gesture of transmitter-II [3]. Most of their audio output was based on imitating the fretboard of a guitar, though with gestures far less straining on the user's hand. Their system was reported to have worked well, but this project aims to eliminate the need for a specialized glove or device in order to interpret the user's gestures [3].

In a similar vein, British music artist Imogen Heap has worked with a team of innovators to develop a program called "Glover" that aims to allow other artists to use off-the-shelf user interfaces to produce music of their own [1]. Heap's team also develops their own gloves, called "MiMu," that offer the user much more customizability when compared to third-party alternatives. Heap shares that their MiMu gloves are Open Sound Control addressable, which allow the user to not only create music, but also control other stage performance elements like lighting and other visuals [1]. Once again, though the capability of Glover is impressive, it relies on the user having some form of hardware interface that allows them to interact with the program. There are several limitations when considering the use of a custom glove as an interface. The most clear of which is that it becomes much more difficult to replicate or mass-produce these gloves if the goal is to deliver the product to the public. Heap and her team have a slight workaround by allowing the use of Glover with interfaces other than their own, MiMu, but this comes at the loss of functionality for the end user [1]. Without the need for a custom glove, or any hardware interface for that matter, this project will be more applicable on a larger scale.

There has also been extensive research completed in relation to detecting key-points on body parts, though a large portion of these projects have been conducted using still images, rather than live video feed. In a machine learning project completed by Zahar Chikishev, the use of fastai library v1.0.42 and resnet34 was found to be mostly successful in determining the key-points on a human hand after being given 51 train images [2]. Each of these hands was oriented the same direction, against similar backgrounds, with similar skin tones, and were manually labeled by Chikishev. From the results, even with a small training size, identifying key-points on a human hand is very doable. However, Chikishev's research was conducted with images rather than live video, which is part of the objective for this project. Sayak Paul has also conducted a similar project, though his version implements TensorFlow and the imaug library. Paul's project worked to identify the key-points of dogs of differing sizes in various poses, based on the Stanford dogs dataset [4]. While dogs may not seem relevant to this project at first, being able to identify body parts in different orientations is crucial to the success of correctly identifying various hand gestures. Combining the results of Chikishev's work with Paul's indicates that it should be possible to correctly label key-points on human hands in varying shapes and gestures, which is the main technical goal of this project.

## 4.0 Approach/Design

### 4.1 Requirements and/or Use Cases and/or Design Goals

#### Requirements:

1. Must have a fully functioning Graphical User Interface that displays the following:

- a. Live video stream of machine's webcam
    - i. Highlighting of hand(s) on screen
    - ii. Key-points of digits
- Gesture Controlled Audio Software

- b. Relevant data of action recognition and gesture control
  - c. Audio parameter(s) current value
2. Fully functioning, responsive, and efficient gesture recognition
  3. Developed using the agile development cycle
  4. Adhere to best software engineering practices
    - a. Clean and readable code
    - b. Well documented progress (take screenshots of progress and note significant developments)
    - c. Abstraction (enable program to easily be run on another device\_

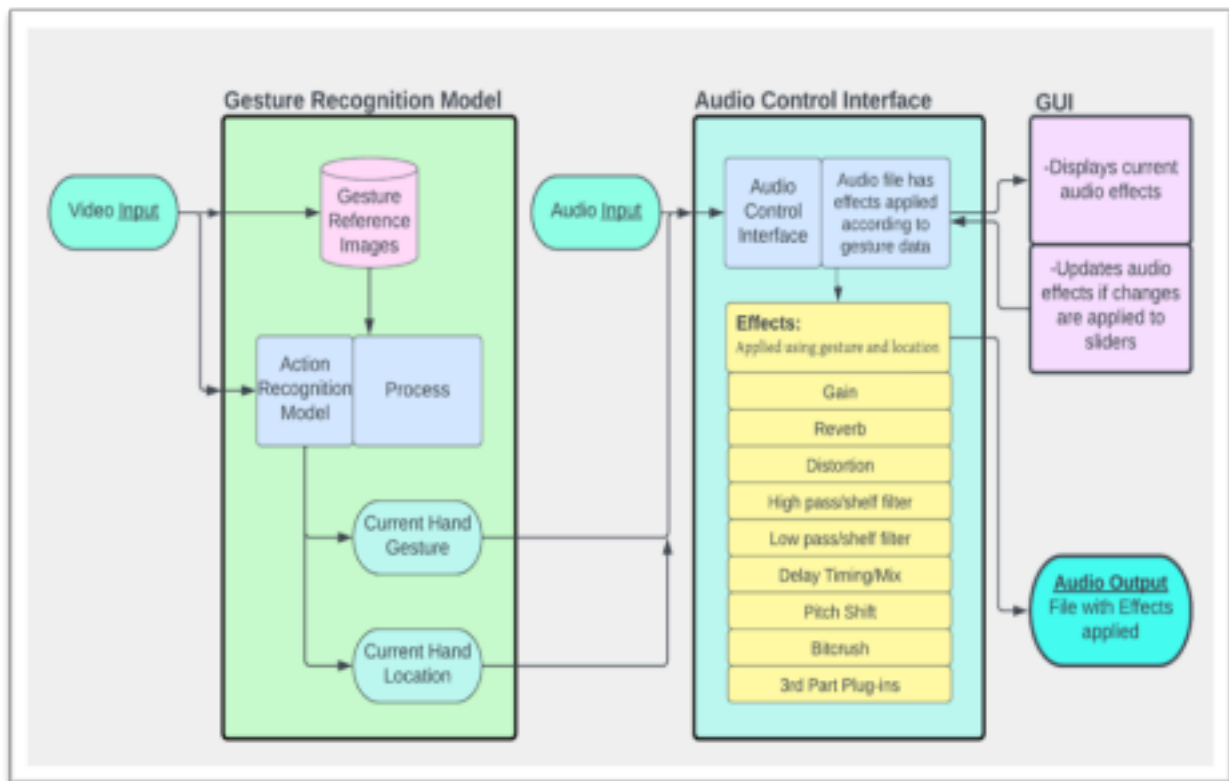
#### Use Cases:

1. Operate created GUI with supported gestures
2. Be modular for implementation of other programs (Other DAW's and potentially other applications not audio related).

#### Design Goals:

1. Be clean and organized
2. Easily maintainable
3. Easily understandable
4. Create a unique and responsive User Interface
  - a. Easily accessible and simple to use
  - b. Allows user to visualize audio changes

## **4.2 Architecture**



Gesture Controlled Audio Software

#### Gesture Recognition Model:

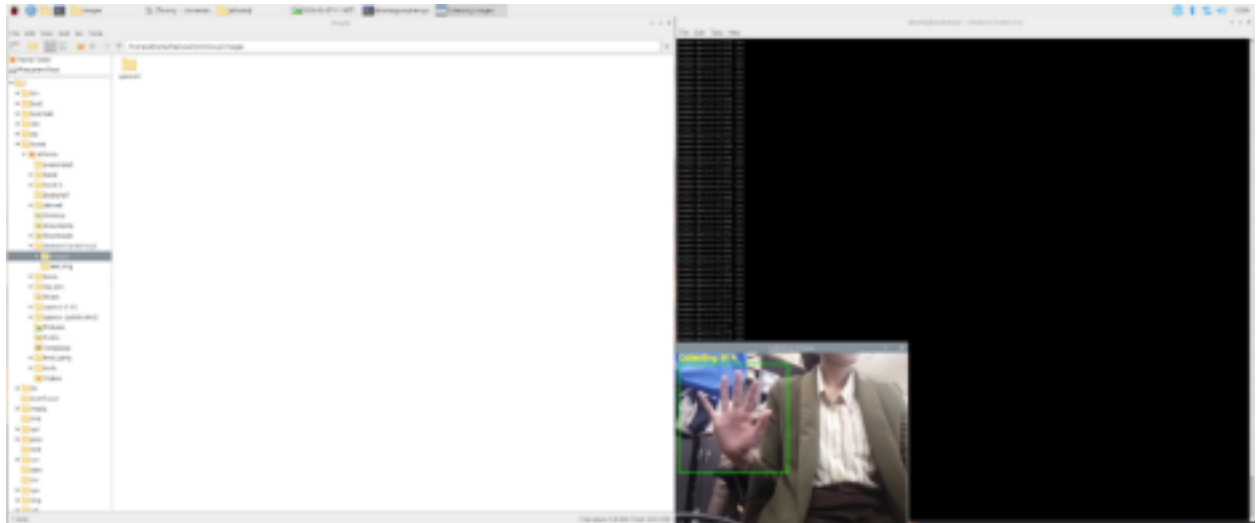
- **Gesture Reference Images**
  - The model needs to be trained on a large set of gestures to increase recognition accuracy and consistency, so there is an option to record the gestures to be used for training through video input.
  - The set of gestures used can technically consist of anything the user chooses, but the expected inputs for training are: thumbs up, open hand, closed fist, peace-sign up, peace-sign down, and pointing left.
- **Action Recognition Model**
  - The model, after being trained, uses the device's connected webcam to allow the user to input their gestures to be recognized. The recognized gestures, along with the x-position and y-position of the gestures, are output as strings and sent to the Audio Control Interface

The artificial intelligence model that we have been building has seen significant progress since the beginning of this project. The two biggest obstacles encountered when designing the AI model were dependency issues and time. Many of the packages we planned to use either did not cooperate with each other's most recent versions, or entirely failed to work properly when running on the newest Raspberry Pi OS. We resolved this by using a legacy version of the Raspberry Pi OS and forcibly downgrading specific packages to versions that would compile and run correctly (most notably Keras 2.1.1, Keras-Applications 1.0.8, tensorflow 1.13.1, and protobuf 4.21.9). On top of this, we found that it takes several hours to compile the AI model

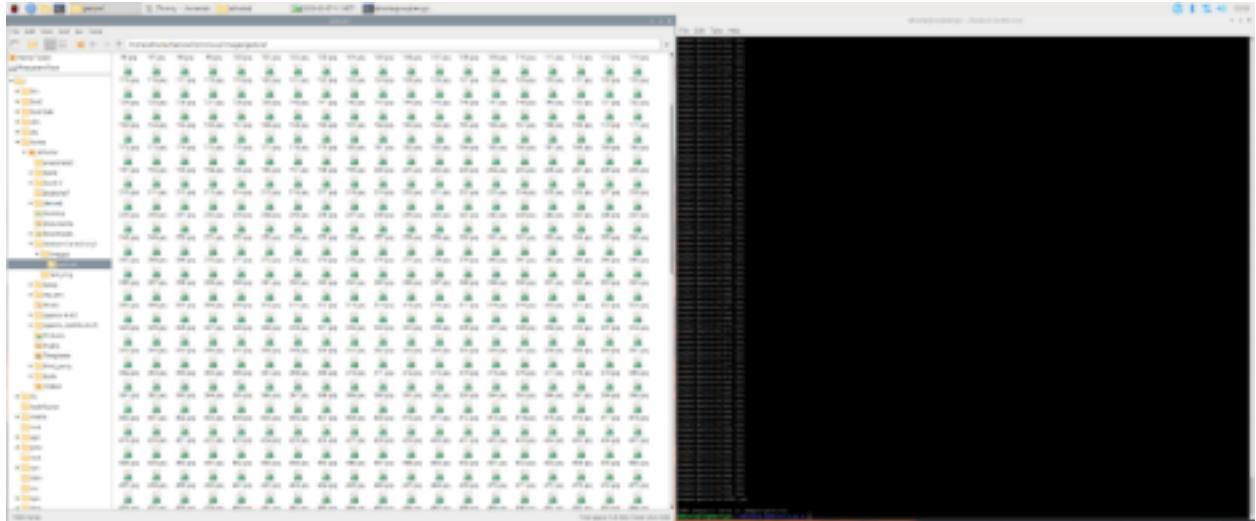
with a new set of gestures, so we have to plan our workflow accordingly.

Since the previous version of the project, we adapted the code to function on devices other than the Raspberry Pi. We found that it did not have enough processing power for the project's needs, so we began running the code on our home computers instead. We also tested several versions of the AI model with different reference images. These could have a different subject, a different background, or simply just a greater number of pictures.

A few screenshots of live feed being captured to train the model:

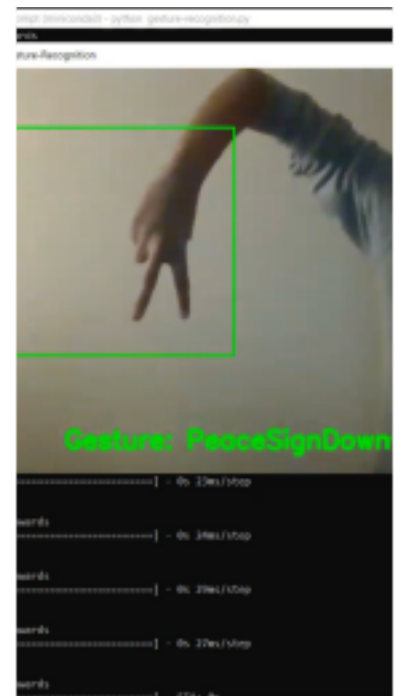
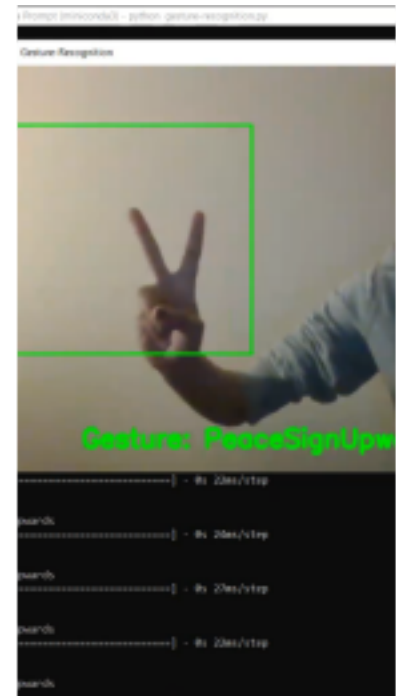
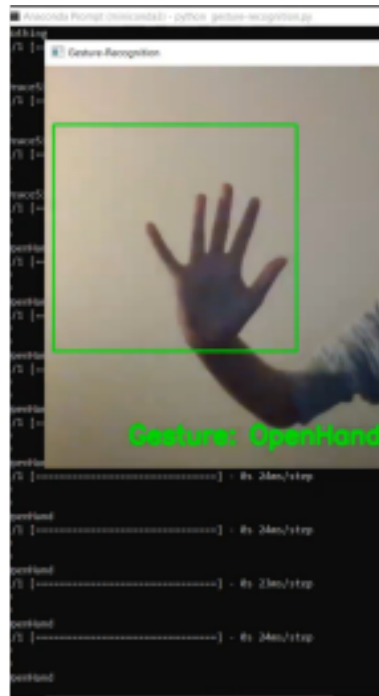


Gesture Controlled Audio Software

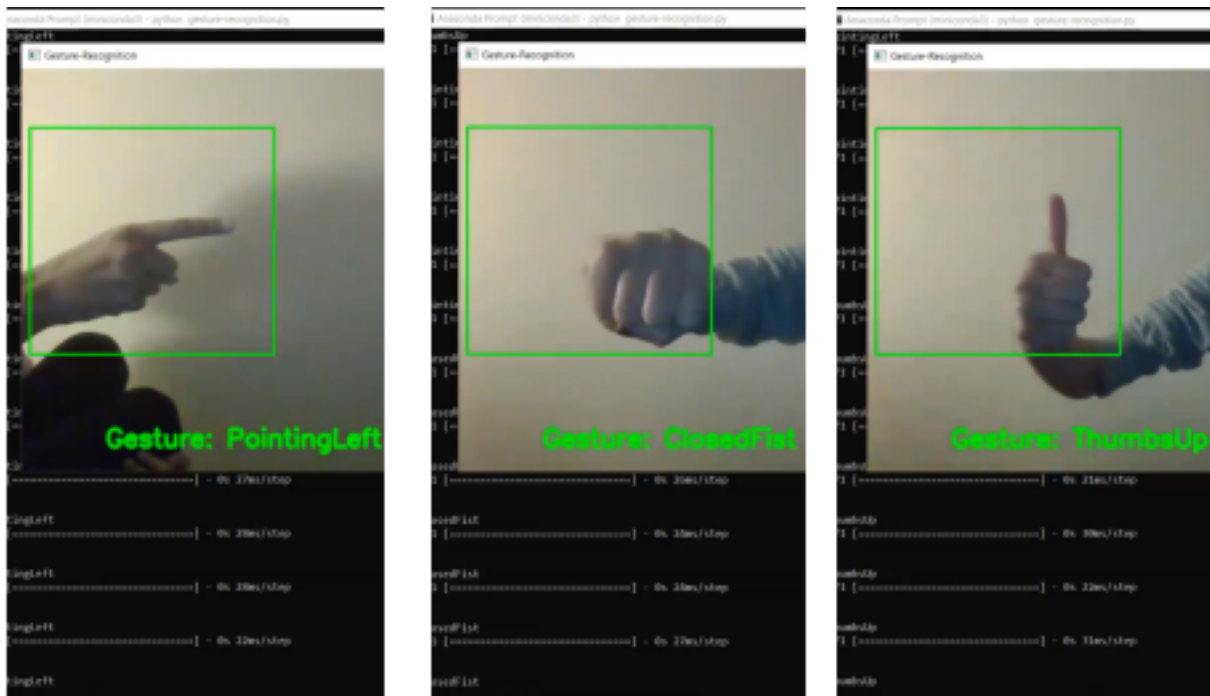


This collection of images is then used to train the AI to recognize the above gesture.

These are runtime screenshots from the newest version of the AI model recognizing all six gestures. The OpenHand, PeaceSignUpwards, PeaceSignDownwards, and PointingLeft all have the best recognition success. The ThumbsUp and ClosedFist gestures are more difficult for the AI to recognize at first, but it can get it correct with enough time.



Gesture Controlled Audio Software



#### Audio Control Interface:

The Audio Interface is simply designed to take in strings and position integers and translate them into audio effect changes. The song to be modified is also input into this module. These strings are representative of the gestures recognized by the AI model. For instance, when the ThumbsUp gesture is detected by the AI, it is then sent to the Audio Interface. The y-position determines the level of gain, and the x-position determines the compressor threshold and ratio as such:

```
# Start the audio stream
with sd.OutputStream(device=output_device, samplerate=samplerate, channels=2, callback=callback):
    # Loop for updating effects during audio playback
    while not audio_finished:
        # Update the effects based on the gesture and hand location
        if GESTURE == "ThumbsUp":
            gain_plugin.gain_db=HAND_LOCATION_Y
            compressor_plugin.threshold_db=HAND_LOCATION_X
            compressor_plugin.ratio=HAND_LOCATION_X*1.5
```

This process will continue until the audio stream has finished and the song has ended. The Audio Interface will also output an audio file with the changes to the song saved for replayability.

#### Graphical User Interface:

Finally, we are creating a graphical user interface for better ease of access for the user. We have decided to name this module "SoundWave Audio Workstation." This GUI will allow the user to see their live video feed and see that their gestures are being recognized. It will also

Gesture Controlled Audio Software

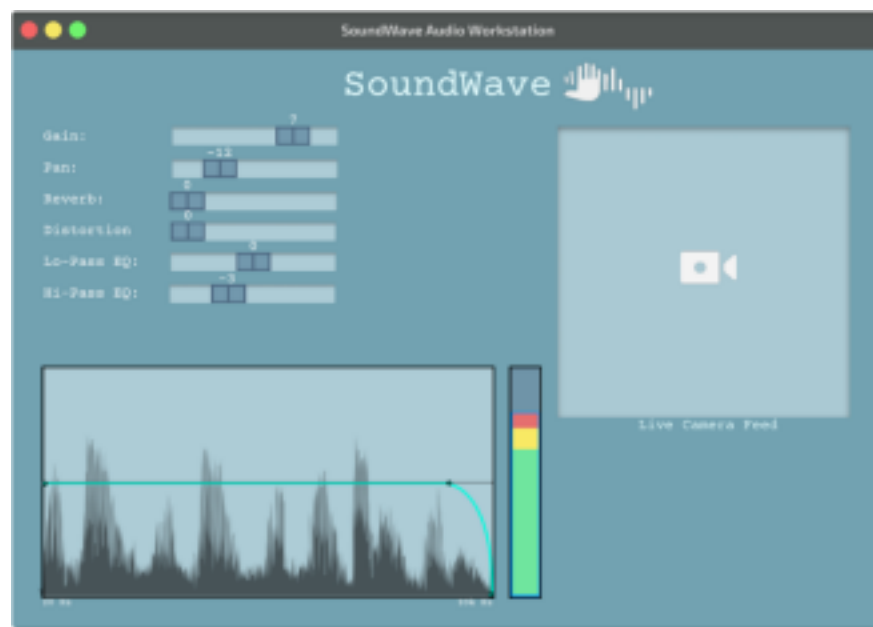
display how the audio is being modified to allow the user to understand what changes are being made based on their gestures. This is essential to provide an enjoyable user experience, and will strongly impact the final demonstration.

The GUI determines how the user will be able to interact with the application and will “receive” the user’s gestures. While the GUI is not what is actively processing the gestures, it will show a live video feed of what the camera is seeing. The gesture guide is as follows: Gain = Thumbs up, Reverb = Open hand, Delay = Closed fist, High-pass/filter = Peace-sign upwards, Low-Pass/filter = Peace-sign downwards, and Pitch shift = Pointing left. The application will read the gesture and the gesture’s location and will apply the audio changes accordingly, which will be displayed as changes to the effects sliders on the GUI.

Initial prototype:



Stylized version:



Gesture Controlled Audio Software

### 4.3 Risks

Risk	Risk Reduction
Noticeable latency between Reaper and Gesture Control Program	Keep code as efficient as possible. If we compartmentalize each part of the project into well-defined scripts, we can easily diagnose issues and work on speeding up individual sections
Difficulty finding hand due to inconsistent lighting, angles, and backgrounds	Train our model to be as well rounded as we can get it with the differences in mind
Difficulty getting our model to work as intended	Consult Dr. Luu when necessary, stick to the schedule, and do thorough research on the subject

### 4.4 Tasks

1. Research Action Recognition to gain a deep understanding of the high level concepts required for gesture controlled software. (COMPLETE)
2. Research and practice using scripting features in any DAW. (COMPLETE)
3. Create a clean and efficient Github page for the foundation of the project. (COMPLETE)
4. Improve Discord with well planned channels and bots for automating tasks and reminders. (COMPLETE)
5. Implement a program to gather live video feed from the user's machine's camera/webcam. (COMPLETE)
  - a. Documentation (COMPLETE)
6. Create simple User Interface with CPU usage in mind (COMPLETE)
  - a. Add live video stream to UI (COMPLETE)
  - b. Add section for data output to UI for testing and demonstrating purposes (COMPLETE)
  - c. Improve design on a simple level just for increased usability. (COMPLETE)
7. Implement object recognition to CV programs to recognize some objects. (COMPLETE)
  - a. Test with recording data to UI in simple cases of recognizing object or not and color of object. (COMPLETE)
  - b. Documentation
8. Connect CV program output to GUI/DAW parameters. (COMPLETE)
  - a. Test with turning track on and off, change volume, and change high-pass frequency level from 0 to -10db.
9. Implement higher level object recognition to detect hand on screen. (COMPLETE)
  - a. Gather data for hand recognition and train model. (COMPLETE)
  - b. Improve UI to draw constraint boxes around found hands. (COMPLETE)
  - c.

Test with different backgrounds, lighting, hand(s), and angles. (COMPLETE)  
 Gesture Controlled Audio Software

d. Documentation (COMPLETE)

10. Add constant detection for moving hand locations. (COMPLETE)
11. Implement finger recognition and gesture control. (COMPLETE)
  - a. Add parameters for adding gesture control. (COMPLETE)
  - b. Recognize simple hand gestures such as closed fist or open palm. (COMPLETE)
  - c. Documentation
12. Add data values based on hand gestures. (Thumbs up = gain, open hand = reverb, closed fist = delay, peace sign up = high pass, peace sign down = low pass, side pointing = pitch shift) (COMPLETE)
13. Improve UI for better appearance and function. (COMPLETE)
14. Implement hand location tracking data, example: “Right Hand Thumbs Up” moves towards top of the screen, DAW increases gain automation to +n right stereo. (COMPLETE)
  - a. Documentation
15. Test, clean code, and improve efficiency/reduce latency where applicable. (COMPLETE)

#### 4.5 Schedule

1	Research Action Recognition to gain a deep understanding of the high-level concepts required for gesture controlled software.  Members: All. Status: Complete.	10/23-11/28
2	Research and practice using scripting features in a DAW. Members: Win, Troy. Status: Complete.	10/23-11/28
3	Create a clean and efficient Github page for the foundation of the project.  Members: Win, Maxx. Status: Complete.	1/17-1/23
4	Improve Discord with well planned channels and bots for automating tasks and reminders.  Members: Win, Maxx. Status: Complete.	1/17-1/23
5	Implement a program to gather live video feed from the user’s machine’s camera/webcam.  Members: Anusha, Maxx. Status: Complete.	1/23-2/6

6	Create a simple User Interface with CPU usage in mind. Members: Win, Troy. Status: Complete.	2/6-2/20
---	--	----------

#### Gesture Controlled Audio Software

7	Implement object recognition to CV programs to recognize some objects.  Members: Anusha. Status: Complete.	2/6-2/20
8	Connect CV program output to GUI/DAW parameters. Members: Anusha. Status: Complete.	2/2-2/27
9	Implement higher level object recognition to detect a hand on screen.  Members: Anusha. Status: Complete	2/27-3/13
	10 Add constant detection for moving hand locations. Members: Anusha. Status: Complete.	3/13-3/27
	Implement finger recognition and gesture control. Members: Anusha. Status: Complete.	3/13-3/27
	Add data values based on hand gestures.  Members: Win, Troy, Maxx. Status: Complete.	3/27-4/3
	Improve UI for better appearance and function. Members: Win. Status: Complete.	4/3-4/10
	14 Implement hand location tracking data, example: "Right Hand Thumbs Up" moves towards top of the screen, DAW increases gain automation to +n right stereo.  Members: All. Status: Complete.	4/10-4/17

	Test, clean code, and improve efficiency/reduce latency where applicable.  Members: All. Status: Complete.	4/17-end
--	--	----------

## Gesture Controlled Audio Software

### 4.6 Deliverables –

- Camera device with on user's computer able to process video and output images
- Raspberry Pi board (or something similar) which acts as the interface module
  - Will be fully trained to identify key-points on a hand
  - Python code for the key-point inference algorithm to be trained
- Microcontroller which serves as the Control Signal Dispatcher
- A Transmission Interface
- Final Product: A Gesture Controlled Audio Software Application that can be run on a
- Final Report on the findings and results of the project

## 5.0 Key Personnel

**Winston Phillips** – Phillips is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed CSCE 2004 and 2014 (Programming Foundations I and II), CSCE 3193 (Programming Paradigms), CSCE 3513 (Software Engineering), and CSCE 4523 (Database management systems). Phillips has 8 years in audio production and composition, was Project Manager and Member for CSCE 3513 Lazer Tag Software Project, and has created multiple low level VST for Logic Pro X using the JUCE Framework in C++. Phillips will be responsible for front-end design, audio production, creating and testing VST's, and project management.

**Anusha Bhattacharyya** – Bhattacharyya is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed CSCE 2004 and 2014 (Programming Foundations I and II), CSCE 3193 (Programming Paradigms), CSCE 3513 (Software Engineering), CSCE 4613 (Artificial Intelligence), and CSCE 4523 (Database management systems). Bhattacharyya was a J.B. Hunt Software Engineering intern where they worked on skills like software development, database management, and statistical data analysis, a part of a NSF-funded REU where they gained proficiency in data analytics, applied statistics, and computer modeling, and also a Data Science Intern for a startup business where they learned and applied data analytics and used SQL and R to develop a program to collect unorganized data on Twitter. Bhattacharyya will be responsible for more back-end development, the training/integration of AI algorithms (more specifically the key-point interface algorithm) via Machine Learning, and communications with professors/possible resources on campus.

**Troy Watts** – Watts is a senior Computer Engineering major in the Computer Science and

Computer Engineering Department at the University of Arkansas. He has completed relevant courses, such as CSCE 3613 (Operating Systems), CSCE 2114 (Digital Design), and CSCE 2004 (Programming Foundations I). He has also experimented with manipulation of audio directly using Python scripts. He will be responsible for managing the hardware for the project (camera, host computer, etc.)

**Maxx Smith** – Smith is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed CSCE 2004, CSCE 2014 (Programming Foundations I and II), CSCE 3193 (Programming Paradigms), CSCE  
Gesture Controlled Audio Software

3513 (Software Engineering), CSCE 4523 (Database Management Systems), and CSCE 4613 (Artificial Intelligence). Smith worked in the CSCE 3513 project as Group Manager, implemented the front-end/back-end communication, and helped establish and manipulate the project's database system. He will be responsible for assisting on front-end and back-end development where needed, establishing group meetings, and leading presentations.

**Dr. Khoa Luu, Thanh-Dat Truong** – Dr. Khoa Luu is an Assistant Professor in the CSCE Department who specializes in Computer Image and Image Understanding. He is the Lab director for the CVIU (Computer Vision and Image Understanding) Lab where he conducts research on computer vision, pattern recognition, biometrics, machine intelligence, and quantum machine learning technologies. Thanh-Dat Truong is a Ph.D student who works in Dr. Luu's CVIU Lab and his research interests mainly lie in Machine Learning, Deep-Learning, and Computer Vision.

## 6.0 Facilities and Equipment

We will be using the Reaper Digital Audio Workstation to connect to our computer vision program, as ReaScript allows for C, C++, and Python implementation. For our artificial intelligence training we will be utilizing the help of Dr. Luu and the Computer Vision and Image Understanding (CVIU) Lab. Equipment will consist of webcams, a MIDI keyboard, and possibly other instruments.

## 7.0 References

- [1] C, C. (2021, August 1). *Gesture Control For Electronic Music*. Sound on Sound. Retrieved October 23, 2022, from <https://www.soundonsound.com/techniques/gesture-control-electronic-music> (“Gesture Control For Electronic Music”)
- [2] Chikishev, Z. (2019, February 6). *Hand Keypoints Detection*. Towards Data Science. Retrieved October 23, 2022, from <https://towardsdatascience.com/hand-keypoints-detection-ec2dca27973e>
- [3] Jhamb, P., & Rehalia, A. (2018, January 27). Wireless Hand Gesture Controlled Multiple Musical Instruments. *International Journal of Information Technology*, 12(1), 19–25. <https://doi.org/10.1007/s41870-018-0096-1>
- [4] Paul, S. (2021, May 2). *Keypoint Detection with Transfer Learning*. Keras

Documentation. Retrieved October 23, 2022, from  
[https://keras.io/examples/vision/keypoint\\_detection/](https://keras.io/examples/vision/keypoint_detection/)