Harmony in Motion: Collaborative Gesture Based Sounds

By: Fourcan Abdullah, Jeffrey Hsu, Charles Richards, Anthony Williams

Problem Motivation

In the ever-evolving landscape of musical performance, the conventional boundaries of artistic expression are continuously being challenged. Recognizing the transformative potential of technology in the realm of music, this project sets out to address the limitations of traditional musical instruments by introducing a groundbreaking gesture/hand movement based sound.

Description/Goal

This project aims to develop a gesture/hand movement-based sound instrument using computer vision within a timeline of 4 weeks. The system will track the gestures of up to two individuals with a single camera, allowing them to manipulate sound parameters collaboratively. Users will use intuitive gestures to create a dynamic and interactive musical experience. We aim to create new methods for and expand on artistic expression in the musical performance space with the development of this instrument.

Technical Approach

01 **Gesture Recognition:** Developed a computer vision system that can recognize and track the gestures of user 1, for example, a closed fist representing a pure sine wave or a sound without many harmonics, and an open hand meaning more harmonics producing a brighter and wider sound.

02 **Multi-user Interactions:** Multi-user interaction where user 1 controls the sound design (timbre) allowing them to choose whether the instrument produces a pure sound with few harmonics or a wide and brighter sound with many harmonics. User 2 will use hand movement relative to a starting point to dictate the amplitude (ex: moving hand up and down) and pitch (ex: moving hand left and right) of the sound produced by user 1.

03 **Midi Data and Synthesizer Integration:** Integrate and relate our gesture data as midi data to our audio synthesizer.

Hand Gesture Recognition by Kazuhito Takahashi

This is a sample program that recognizes hand signs and finger gestures with a simple MLP using the detected key points. In addition, it builds its model by utilized linear stack of layers, which takes the output of a previous layer and uses it as an input for the next in a sequential fashion.

Input layer: Accounts for 21 features, which also account for sets of 2 for each feature

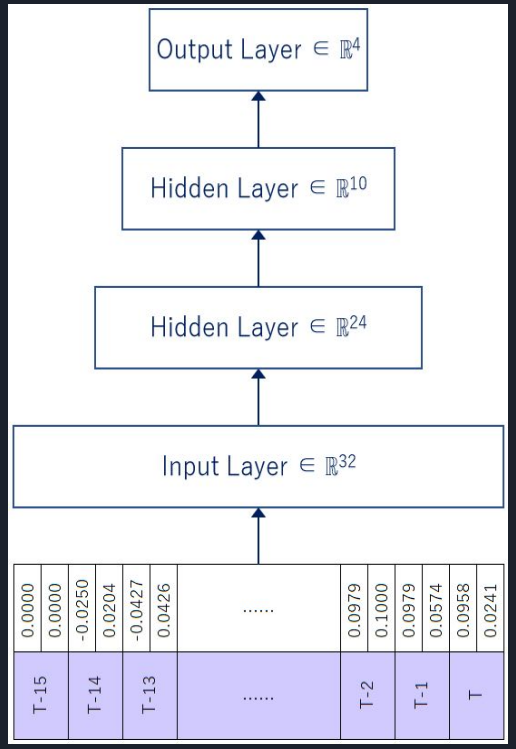
Output layer: Regularization technique, Dropout, in order to randomly set 20% of inputs to 0. This reduces dependency and promotes redundancy.

Redundancy

Redundancy, in this context, means that multiple pathways in the network can contribute to the same or similar features. This can improve the generalization of the model because it becomes less sensitive to the precise configuration of neurons. The model is better equipped to recognize features in various forms or contexts.

Model Training

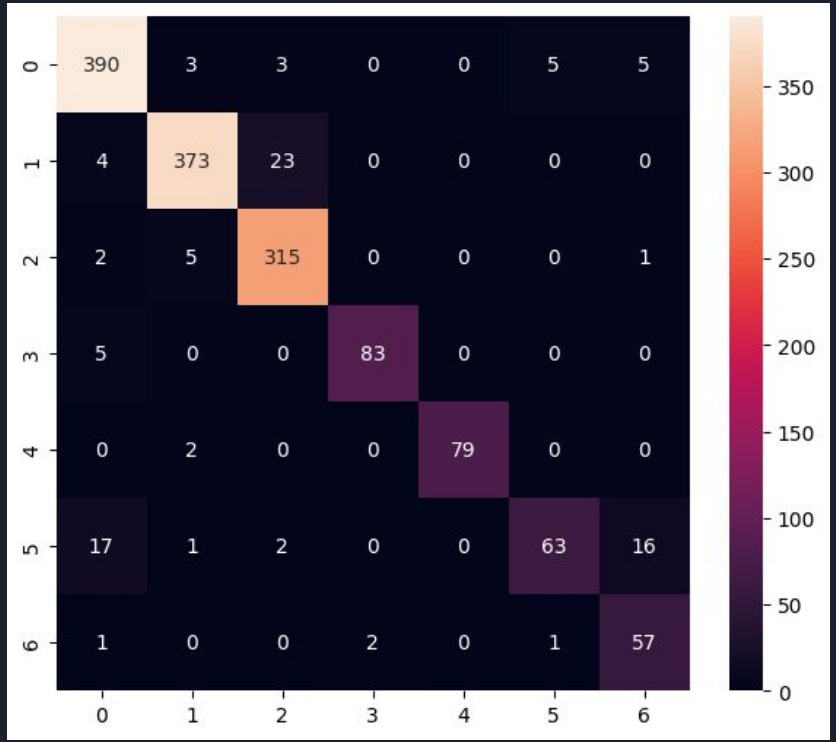
The model is trained on a dataset represented by X_train and y_train, which contain input features and corresponding labels, respectively. The training process involves iterating over the entire dataset for a specified number of epochs (in this case, 1000 epochs) and updating the model's weights based on the optimization of a chosen loss function. The training is performed in batches of 128 samples at a time, enhancing computational efficiency and memory usage.



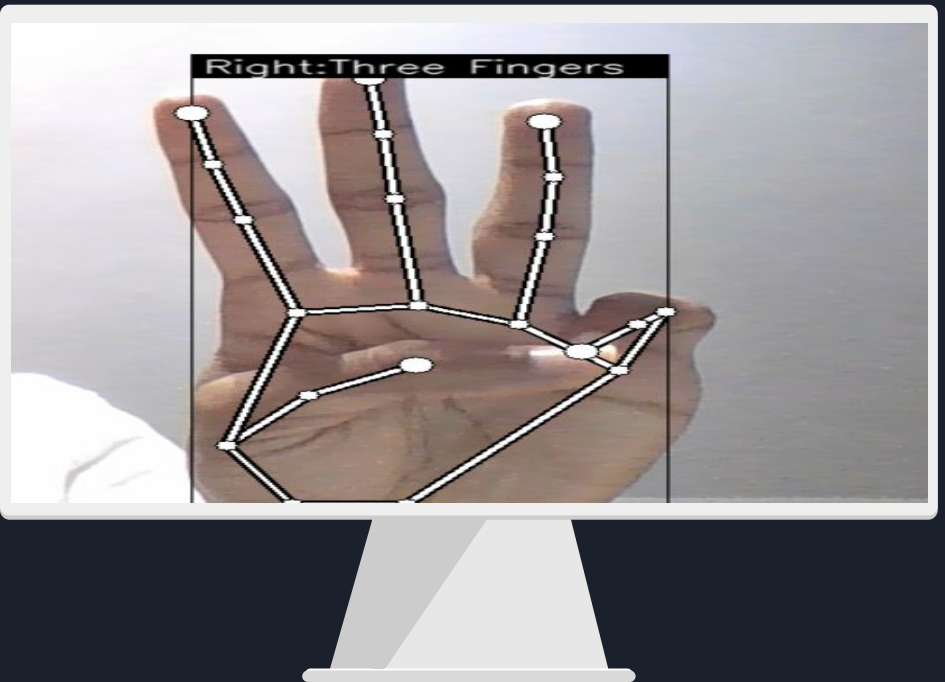
Confusion Matrix

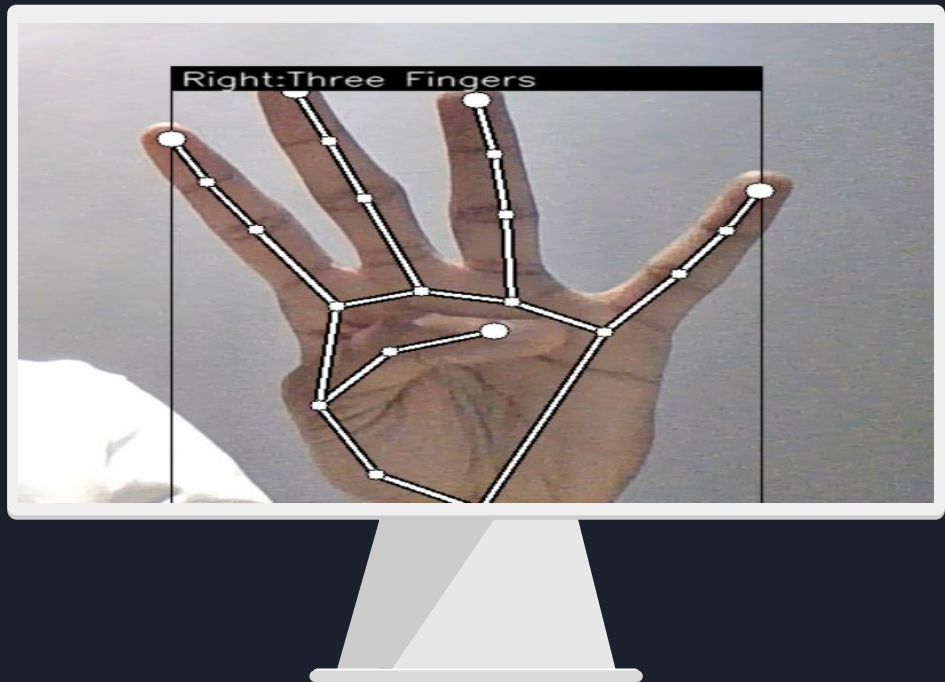
Classification Report

	precision	recall	f1-score	support
0	0.93	0.96	0.95	406
1	0.97	0.93	0.95	400
2	0.92	0.98	0.95	323
3	0.98	0.94	0.96	88
4	1.00	0.98	0.99	81
5	0.91	0.64	0.75	99
6	0.72	0.93	0.81	61
accuracy			0.93	1458
macro avg	0.92	0.91	0.91	1458
weighted avg	0.94	0.93	0.93	1458



Errors





Mediapipe Hands, and OpenCV Implementation

This is a program that uses Mediapipe hands and OpenCV to detect hand movement and hand gestures to control an instrument.

Mediapipe hand is used to detect the key points in the hand

OpenCV is used for image processing and Video analysis.

OpenCV

- OpenCV is the world's largest open source computer vision library
- Provides a variety of tools, including data collection + pre-processing, model architecture, validation, feature detection + matching, etc.
- Used in conjunction with ML frameworks to train models

Capabilities used:

- Video capture
- Image color space conversion

MediaPipe

Overview:

- Open-source library developed by Google
- Works in conjunction with OpenCV
- Contains pre-trained models for hand tracking, face detection, etc.

Challenges for hand tracking:

- Compared to faces, hands lack high-contrast features
- Large scale span (~20x)
- Articulated fingers will result in a high number of bounding box ratios
    - Can lead to high computation cost, difficulty in learning, and overfitting for certain ratios
- Solution: 2 models working together: palm detector + hand landmark model

MediaPipe BlazePalm Detector

Palm detector: operates on full image and locates palm in oriented bounding box around hand

- Single-shot detector model (SSD)
- Different spatial resolutions to generate feature maps at multiple scales
- Palms can be modelled using square bounding boxes (only 1 ratio), reducing number of anchors needed
- Non-max suppression

During training, focal loss is minimized to support large amount of anchors due to large scale variance

- Deals with class imbalance by helping the model focus on hard-to-classify example and reducing impact of easy-to-classify examples

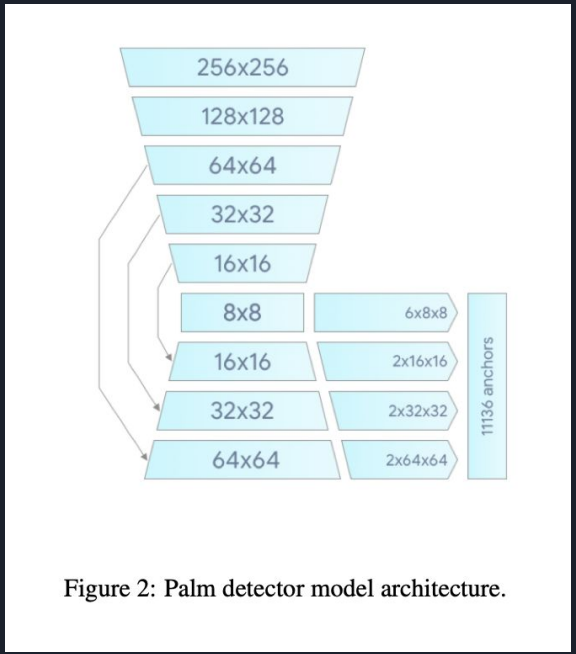


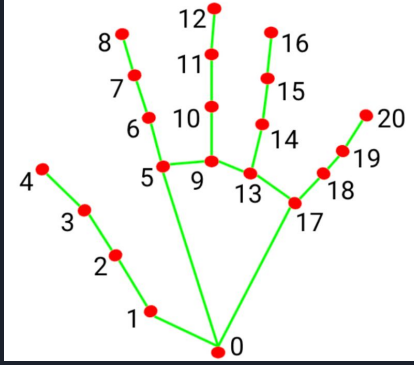
Figure 2: Palm detector model architecture.

Zhang et al., 2020 . *MediaPipe Hands: On-device Real-time Hand Tracking.*

MediaPipe Hand Landmark Model

Hand landmark model works within the cropped bounding box and has 3 outputs

- 21 2.5D coordinates inside detected hand regions
- Hand flag (presence of hand)
- Handedness (left or right hand)



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

MediaPipe Hand Landmark Model

Hand landmark model works within the cropped bounding box and has 3 outputs

- 21 2.5D coordinates inside detected hand regions
- Hand flag (presence of hand)
- Handedness (left or right hand)

Instead of traditional object detection methods that use classification, the hand landmark model uses regression

- Directly predicts the bounding box coordinates
- Simpler and more efficient than having a separate region proposal network

Different versions can accommodate different processing capabilities and accuracy requirements

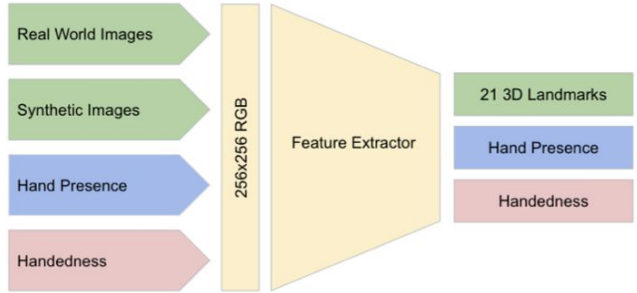


Figure 3: Architecture of our hand landmark model. The model has three outputs sharing a feature extractor. Each head is trained by correspondent datasets marked in the same color. See Section 2.2 for more detail.

Zhang et al., 2020 . *MediaPipe Hands: On-device Real-time Hand Tracking.*

Model Architecture

Using the image input it predicts the locations of keypoints and creates a oriented bounding box around the hand palms.

In a live setting this is run once since a bounding box is calculated from the landmark prediction of the previous frame as input for the current frame.

Thus palm detection is run either once at the beginning or when it loses track of the palm.

Zhang et al., 2020 . *MediaPipe Hands: On-device Real-time Hand Tracking.*

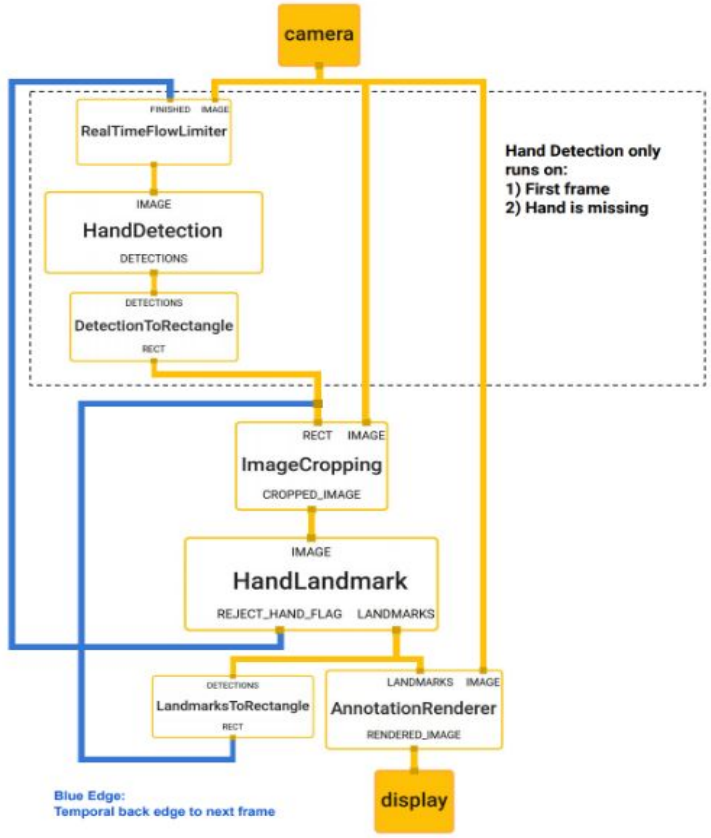


Figure 5: The hand landmark models output controls when the hand detection model is triggered. This behavior is achieved by MediaPipes powerful synchronization building blocks, resulting in high performance and optimal throughput of the ML pipeline.

MediaPipe Hands - Dataset

Three types of datasets were used for training:

In-the-wild dataset: Contains 6K images of hands in various geographical variety, lighting conditions and hand appearances. All the hands though were in simple positions.

In-house collected gesture dataset: 10K hand pictures of limited variation, but with alot more varied gestures, angles and hand positions.

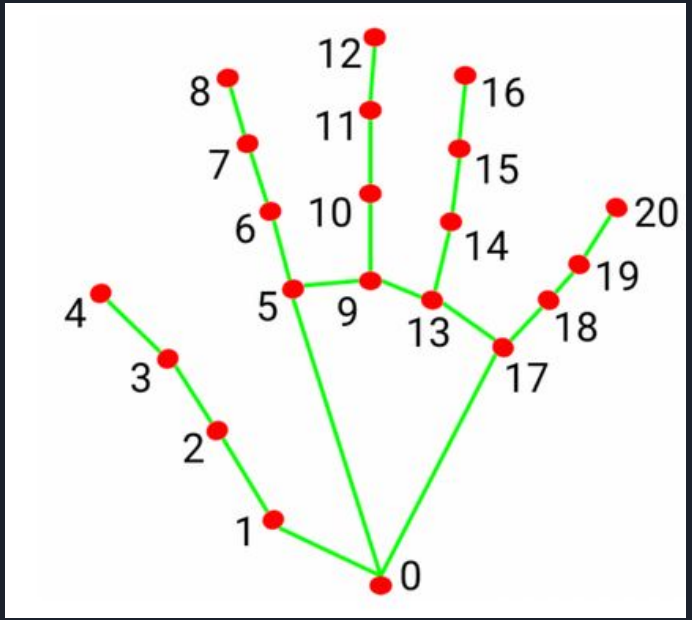
Synthetic dataset: 100K images obtained from synthetic dynamic hand model videos in various lighting and geographic backgrounds.

The 21 landmarks were added to all the images.

Implementation

In this program using the locations of the 21 landmark positions we can register certain gestures, such as fists, open palms, thumbs up etc. Using these gestures we can control which kind of waves the music will play

The X,Y location of the wrist landmark is used with the input image size to control the amplitude and frequency of the sound. The distance between the thumb tip and the index finger tip is used to control the timbre of the sound. This distance is found using linear interpolation.



JUCE - Audio Synthesizer

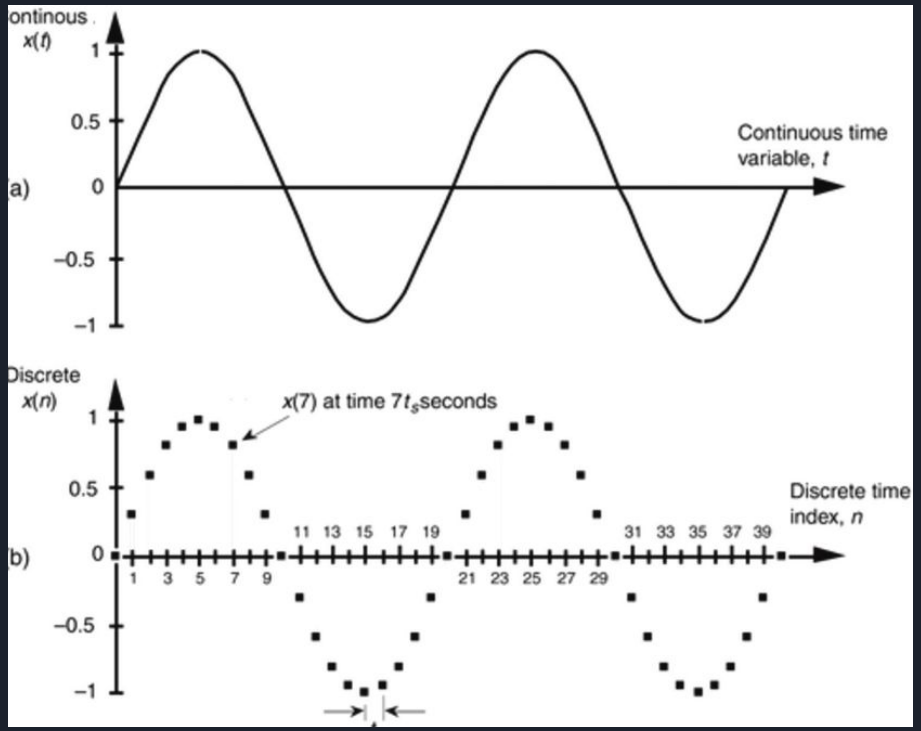
Juce is a widely used, open source, audio framework in C++ for developing Audio Applications and Audio Plugins

We are using JUCE to develop an in-house audio synthesizer, which will allow us to have the greatest control over how we represent gesture data as audio data.

There is some waveform, and the program is generating points of the waveform at discrete times. This is audio data which can then be played.

OSC allows us to take the results from Python and send those results to JUCE.

JUCE - Audio Synthesizer



https://www.researchgate.net/publication/320523811/figure/fig1/AS:551529365344263@1508506255617/A-time-domain-sinewave-a-continuous-waveform-representation-and-b-discrete-sample.png

JUCE

Synthesizer

$$y(t) = A\sin(2\pi ft + \varphi) = A\sin(\omega t + \varphi)$$

- A, *amplitude*, the peak deviation of the function from zero.
- f, *ordinary frequency*, the *number* of oscillations (cycles) that occur each second of time.
- $\omega = 2\pi f$, *angular frequency*, the rate of change of the function argument in units of radians per second.
- φ, *phase*, specifies (in radians) where in its cycle the oscillation is at $t = 0$.
  - When φ is non-zero, the entire waveform appears to be shifted backwards in time by the amount $\frac{\varphi}{\omega}$ seconds. A negative value represents a delay, and a positive value represents an advance.
  - Adding or subtracting 2π (one cycle) to the phase results in an equivalent wave.

JUCE - Open Sound Control (OSC)

“Open Sound Control (OSC) is an open, transport-independent, message-based protocol developed for communication among computers, sound synthesizers, and other multimedia devices.”[1]

Our gesture tracking is built using MediaPipe with Python and JUCE is a C++ framework.

We need to be able to take the results from Python and send those results to JUCE.

We use OSC (Open Sound Control) to do so. JUCE has a built implementation of OSC, and there exists a library python-osc. We leverage the python-osc library to create a client, and with JUCE as an OSC server.

We are able to send real time data as real values [0,1], which we can then use in JUCE.

https://ccrma.stanford.edu/groups/osc/spec-1_0.html

We’ve mapped several gestures to difference synth functions.

x, y coordinate data from hand detection is normalized to a value between 0 and 1.

We also have the option of representing this in a single data point, for example 0=no gesture, 0.2 = gesture 1, 0.4 = gesture 2, et...

JUCE - Implementation

We are mapping three dimensions of data to our synthesizer.

- The x-coordinate is mapped to pitch / frequency
- The y-coordinate is mapped to volume / amplitude
- Distance between two pinched fingers is mapped to timbre / brightness of the sound

We are intending to map at least two hands, such that each hand controls one instrument, for a total of 6 data points.

Resources

MediaPipe Hands: On-device Real-time Hand Tracking

MediaPipe: A Framework for Building Perception Pipelines

MediaPipe Hands

Hand Keypoint Detection in Single Images using Multiview Bootstrapping

Hand landmarks detection

Hand Gesture Recognition by Kazuhito Takahashi