NEURALSYNTH

Zéphir Lorne
NUS / McGill University
zephir@u.nus.edu
zephir.lorne@mcgill.ca

Jianan Zhang
NUS
e0950471@u.nus.edu

1. ABSTRACT

Current research shows how engaging with music is essential and can improve our overall well being in life. Yet, we are often faced with multiple barriers to creating music, such as lack of access to instruments or skill to play.

Also, our daily life finds our body movement to be very connective with music. When the music turns up, we tend to dance to jump to feel the energy with our body waves. We are convinced that the most natural form to connect music to ourselves is through body movement rather than obscure music theories.

As a result, we created NeuralSynth, a web-application with which anyone can start creating music, without any experience or instrument other than with a webcam equipped computer and your body. NeuralSynth works with a computer vision model capable of tracking your body movements and mapping them to different audio parameters to let you create music with your body.

2. INTRODUCTION

We believe music is fundamental in our lives, to let us express our creativity, let go of tension and stress we might be feeling, and reconnect with ourselves and others. Furthermore, it has been shown that engaging with music "can have significant benefits across the lifespan" [1]. On the other hand, we believe the most natural form for our body to synchronize with music is to dance. Suppose we can somehow connect dancing and synthesizing music together. In that case, it can not only enable these people with no music knowledge but also let the music flow in the most natural way. As a result, our project is about giving anyone the ability to create music with their body, no instruments or previous musical experience needed.

Instead of creating music with a physical instrument or via inputting keys and clicks on a computer, the idea behind this project is that your body movements, more precisely of your arms and hands, along with a webcam-equipped computer, is enough to create simple music. Then, the position of your limbs, determined by the camera and computer vision algorithms can be translated into different sounds of which the user can control the pitch, volume and other effects.

In this first implementation of our system, the right hand's height controls the frequency of some oscillator to modify the pitch of the produced sound. Meanwhile, the left hand's y-position controls the reverb effect for additional creativity. Furthermore, the volume of the produced audio can be controlled with your right hand's openness, and the distortion of the produced audio can be controlled with your left hand's openness.

3. EXISTING SYSTEMS

3.1 Computer Vision

3.1.1 Body Synth

Some systems with a similar vision of being able to create music with your body already exist, with some variability. For example, *Use All Five and Google Creative Lab* created *Body Synth*, a webapp in which each of your body parts can play a note of your choosing when you move. It was built using the tensorflow.js PoseNet model to detect body position from a webcam, as well as Tone.js for audio processing [2]. However, this project does not seem to be active anymore as we were unable to run it despite our efforts to make it work on different browsers and computers.

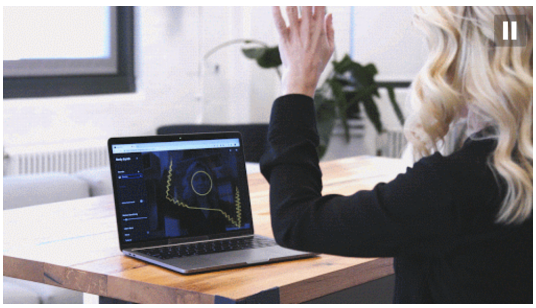


Figure 1: Body Synth example from Google

3.1.2 *Kagura*

Kagura is a paid software successfully funded by a Kickstarter in which you can play music with your hands using AR instruments overlays that you can customise and place as you wish. As this is not an open source project, there is no mention of the technology used behind [3].

Figure 2: Karuga software

3.2 Motion sensors

Another way some companies have come up with to create music with your body is with motion sensors you can strap to your wrist and ankles. For example, *Instrument of Things'* created a small wearable sensor, *SOMI-1* which can pair with your phone in order to send its velocity, slew, acceleration and tilt in three axes. Then, the app interprets these parameters to create music based on presets chosen from your phone [4].

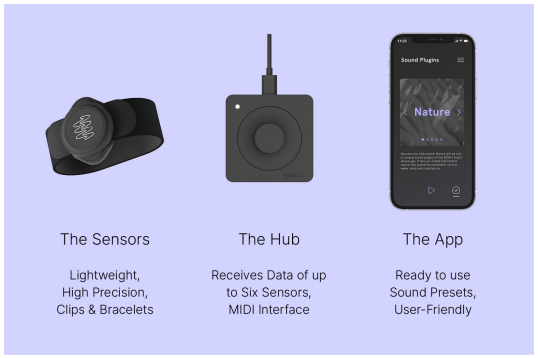


Figure 3: SOMI-1 parts

3.3 The Theremin

Actually, long before computers existed, an instrument was already capable of creating music without physical contact and just the movement of your hands: the Theremin. The electronic instrument was invented by Leon Theremin in 1920 and consisted in two antennas towards which you would direct your hands closer or further. As such, the pitch and the amplitude of some radio tube oscillations would be altered, producing music [5].

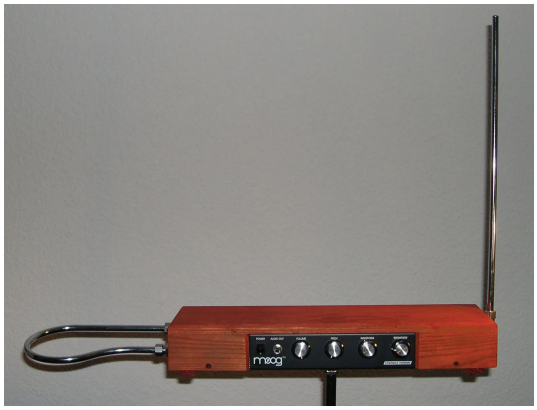


Figure 4: A Theremin [5].

4. PROBLEM FORMULATION

While systems to create music with your body exist as seen above, none appear to be achieving our goal of letting *anyone* create music with *just* their body and a computer. Indeed, the existing systems we researched are either paid, not active anymore, or require the use of external sensors and parts.

As a result, the initial scope of this project is to recreate some sort of Theremin, but using computer vision instead of antennas, that you can access as a webapp. As such, the goal we set ourselves to is to be able to control the pitch and amplitude of some sound with the movement of your hand and a webcam. Once this is achieved, the idea is to build upon that and add features to develop the virtual instrument further. For example, we can imagine the possibility to change the sound which we control between different presets or instruments. Furthermore, we could also imagine a looping/sampling system to allow more complex music creation, along with post-recording remix possibilities like moving your hand laterally to control the speed of the playback.

5. METHODOLOGY

The NeuralSynth system is composed of three main components. Namely, live pose estimation with the Mediapipe API, audio synthesis through Tone.JS, and the web-app user interface made with React and javascript.

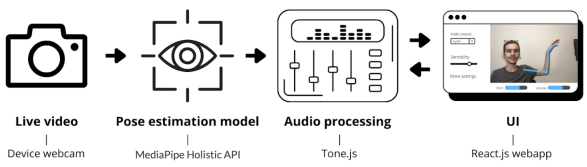


Figure 5: System structure schematic

5.1 Live pose estimation

The live pose estimation component is key in our system as it represents the input to our application based on which sound is generated. For our system to be able to perform live and from a web browser, it is thus important to have a lightweight yet accurate enough computer vision system for pose estimation.

5.1.1 MediaPipe Holistic API

Thankfully, the MediaPipe Holistic API meets these requirements perfectly. MediaPipe is a machine learning pipeline developed by Google for live media analysis. Initially, the framework was only used internally by Google in products like YouTube or Google Lens for applications like real-time object tracking and blurring [6]. However, the project was open sourced during the 2019 Computer Vision and Pattern Recognition conference, making available many powerful solutions like Object Detection, Face Detection, Hand Tracking, Multi-hand Tracking and Hair Segmentation cross-platform [7]. In our application, we use the Holistic Tracking API solution, capable of simultaneously perceive human-pose, face landmarks and hand tracking in real time. As a result, through this API and a computer webcam, we are able to track 543 landmarks: 33 for the body segmentation, 21 for each hand, and 468 for the face [see Figure 6].

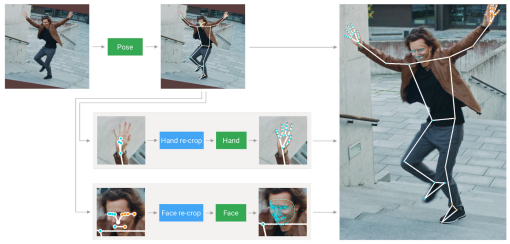


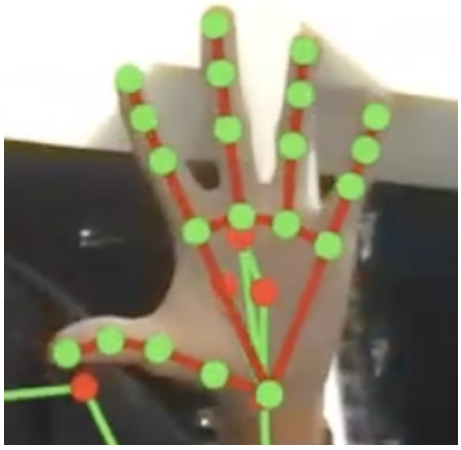
Figure 6: MediaPipe Holistic Pipeline Overview. [8]

5.1.2 Additional body tracking calculations

For each input frame, we then process these landmarks in order to make sense of more global the body tracking information. In our case, we chose to first focus on hand position and gestures.

To determine the center of one's hand, we simply calculate the centroid of the triangle formed by one's wrist, pinky and and thumb using the body pose data [Figure 7]. With this, we are able to obtain the real-time x and y position of each hand from which we can map different audio features.

(a) Closed Hand, average joint angle: 110º



(b) Opened Hand, average joint angle: 170º

Figure 8: Hand landmark and joint segmentation with MediaPipe

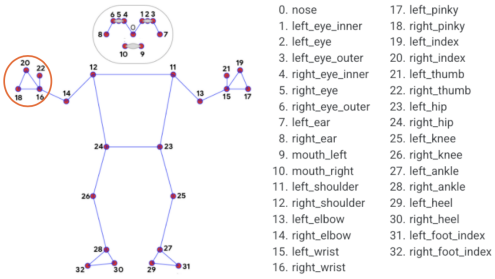


Figure 7: The 33 pose landmarks. 16, 18 and 22 are the ones we use to calculate the centroid of one's hand

Futhermore, we also calculate each hand's openness for this purpose. To do so, we create vectors between hand landmarks, representing all finger joints, and proceed to calculate the angle between each of them. Then, based on the overall average degree angle, we can determine how open or closed the hand is, with around 110º being closed and 180º fully opened [see Figure 8].

5.2 Audio processing

Following this, the idea is to map the relevant pose and hand tracking information we processed into different audio parameters such as pitch and volume to synthesize some original sound and create music. For this, we are using Tone.js, a popular Web Audio Framework for music creation [9].

To synthesize audio, we first create an oscillator with a sine wave of 440Hz. Then, whenever the position of the right hand changes, we update the frequency of the oscillator, as long as it is within frame. We map the y position of the hand from 10Hz when the hand is at the bottom of the frame, to 440Hz when it is at the top. In parallel, we map the right hand's openness from normal volume when it is fully open (average finger joint degree of 180º), to -30dB when it is closed (average finger joint degree of 120º) to gradually change the volume. If the average finger joint degree is below 120º, we set the volume to -100dB to completely kill the sound.

For additional control, we also create a reverb object that we chain to the oscillator, in order to add some effects to the produces sound. The degree to which the reverb is active is also mapped to the user's input, this time with the left hand's y position.

Finally, the user can also decide which type of wave is produced by the oscillator through a dropdown menu in the UI, between 'Sine', 'Triangle', 'Rectangular', and 'Sawtooth' to create a greater range variation of sounds [see Figure 9].

5.3 User interface

For the user interface and interactions, we built a webapp using React.js, a JavaScript library for building UIs based on components.

On the right, we display the mirrored webcam view of the user with the tracking data juxtaposed on his body for greater immersion [see Figure 9].

On the left, we present a control panel displaying the real time tracking data of each hand's x-y position and openness. By default, the audio is turned off as many browsers don't allow sound to automatically start without any user input upon the loading of a page. As a result, we added a button to toggle the audio on and off. Furthermore, we also created sliders that update based on the produced the audio output, namely volume and frequency to provide more feedback to the user. This also enables the user to experiment without the sound actually turned on. Finally, we also added a dropdown menu to allow the user to select different oscillator wave types to further customize the output audio [see Figure 9].

In addition, the user is greeted with a welcome page introducing the concept of our web-app along with some instructions on how to use it.

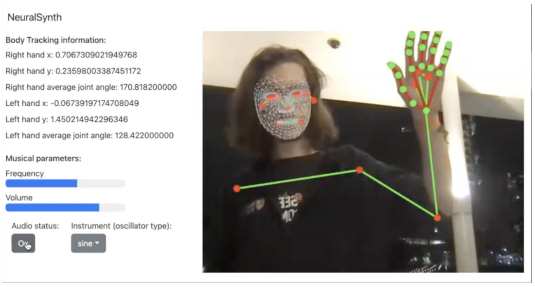


Figure 9: Screenshot from our NeuralSynth web application

6. DEMONSTRATION

Our web-app is fully functional. You can try it out on our GitHub page here: https://rebrand.ly/NeuralSynthDemoWebsite. For the best experience, please use Google Chrome.

Alternatively, a recorded demo video can be viewed here: https://rebrand.ly/NeuralSynthDemoVideo.

Furthermore, we made our code open source and freely available on GitHub here: https://rebrand.ly/NeuralSynthGitHub.

7. FUTURE PLAN

In the future, we want to make our system performable. Our vision emphasizes putting our system down to practical musical uses. We divided this vision into small sub-goals–multi-input and multi-channel functionality, Digital Audio Workstation(DAW) Plugin functionality, and live performance functionality.

7.1 Multi-input and multi-channel

In the modern music industry, music is made by overlapping one instrument over another. It is essential to enable multi-input devices and allow multichannel modification.

Figure 10: Modern Digital Audio Workstation with multi-inputs/channels

In our scenario, we want to achieve the functionalities of a beat-box loop station. We can have our voices recorded on the Webpages and loop it with a preferable BPM (Beats Per Minute.) We can add layers and build our music from low-frequency to high-frequency, starting from drums and basses. As human voice can not perform a very high-frequency/low-frequency voice, our device empowers them to expand a broader frequency range using the pitch-changing function. Reverb is also a significant audio effect these days, as it can strengthen the power of a sound. Our system also enables the reverb-changing function to control the decay of the pre-recorded voices. Using Reverb can make a human-voice bass/drum more profound.

Figure 11: Beat-box Loop station

7.2 Digital Audio Workstation Plug-in

As the modern music industry is so occupied with Digital Audio Workstation (DAW), we want to adjust our interface also to be a DAW plug-in. In the mixing process, there is an essential part called automation.

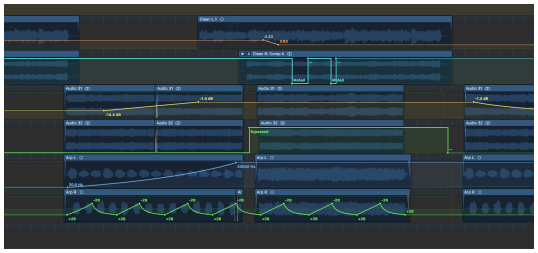


Figure 12: Mixing Automation

Automation means the change of audio effect parameters through the timeline. [see Figure12] Spreading out the timeline, the curves and lines indicate the change of some audio effects parameters like reverb, pitch, and volume. These pre-set modulations allow electronic instruments to have a sense of dynamics. Avoiding electronic instruments to sound like an emotionless robot, dynamics is also an important topic in the digital music industry that has yet to be fully solved. One common and efficient way to modify dynamics is through the "write" function. It enables producers to adjust these audio effects parameters in run-time, and the computers can record these modified changes. For our scenarios, we believe that using our human body movement can be more natural and elegant than twisting your computer's mouse. Our DAW plug-ins can help to perform a more natural and elegant workflow for producers to "write" the automation.

7.3 live performance

Another main focus of our project is human-computer interaction (HCI.) We believe that music is the most natural thing connecting your body movement, also known as dancing, to itself. Digital music suffers from the unnatural discrete representation format, but our body movement to the music can help transfer our innate continuous natural waves to it. [see Figure13]

Figure 13: Live Performance with body tracking

What we envision starts from the beat-box loop station that a musician can use our interface to perform live. However, we also want our system to have functionalities to let users upload their music and even their audio effect pedals/devices. We want to let our system be very flexible. This means we need to get a more downstream API than Tone.js.

8. CONCLUSION

As a whole, we designed and built a synthesizer interface that uses your hand gestures to control. We implanted volume, pitch, reverb, and distortion functions into it. Also, you can select from sine, square, and triangular wave forms for the synthesizer. We envisioned this as the first step of building a multi-output multichannel interface with Digital Audio Workstation plug-ins and live performance functionalities.

By using the body tracking algorithm, Mediapipe, we enable the user to interact with music with no music knowledge needed. When he puts up his hand, he can have a very straightforward feeling of pitch and how music can be built from his own hand. This achieves our pre-set goal–overcoming barriers to creating music, such as lack of access to instruments or skills.

9. DIVISION OF TEAMWORK

9.1 Zéphir Lorne

I came up with this project idea and did the preliminary research about existing systems and background information for this project. In addition, I chose the different tools we would need for it and how to use them (MediaPipe, Tone.JS, React.JS). Furthermore, I designed and coded the UI/UX with React and implemented the MediaPipe API in the code. I also coded the additional body tracking calculations needed to determine the center of one's hand and the degree to which it was opened or closed. Finally, I took care to map this data to the output audio to modify the pitch and volume of an oscillator with your right hand and Tone.JS.

9.2 Jianan Zhang

I joined Zéphir's project and followed it with his preliminary research. During the first part of our project, I have to learn React.js to keep up with Zéphir's progress. We met occasionally and updated each other's progress. I finally kept up with Zéphir 's progress and implanted the left-hand motion tracking with the Reverb-changing and Distortion-changing functions.

10. REFERENCES

[1] N. Rickard and K. McFerran, Eds., *Lifelong engagement with music: Benefits for mental health and well-being*, 1st ed. United States of America: Nova Science Publishers, 2012.

[2] "Use all five google creative lab, body synth," 2018. [Online]. Available: https://experiments.withgoogle.com/body-synth

[3] "Shikumi design, kagura," 2016. [Online]. Available: https://www.kickstarter.com/projects/142645468/kagura-change-your-motion-into-music

[4] "Instrument of things, som-1: Turn your movements into sound," 2021. [Online]. Available: https://www.indiegogo.com/projects/somi-1-turn-your-movements-into-sound#/

[5] "Theremin," 2019. [Online]. Available: https://www.britannica.com/art/theremin

[6] M. G. Yong, "Object detection and tracking using mediapipe," *Google Developers Blog*, 2019. [Online]. Available: https://developers.googleblog.com/2019/12/object-detection-and-tracking-using-mediapipe.html

[7] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, and J. Lee, "Mediapipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[8] Google, "Mediapipe holistic." [Online]. Available: https://google.github.io/mediapipe/solutions/holistic

[9] "Tone.js." [Online]. Available: https://tonejs.github.io