MuGeVI: A Multi-Functional Gesture-Controlled Virtual Instrument

Yue Yang
Department of Music AI and Information Technology
Central Conservatory of Music
Beijing, China
yewyang@mail.ccom.edu.cn

Zhaowen Wang*
Department of Music AI and Information Technology
Central Conservatory of Music
Beijing, China
wzw@mail.ccom.edu.cn

Zijin Li†
Department of Music AI and Information Technology
Central Conservatory of Music
Beijing, China
lzijin@ccom.edu.cn

ABSTRACT

Currently, most of the digital musical instruments cannot leave the use of dedicated hardware devices, making them limited in terms of user popularity and resource conservation. In this paper, we propose a new computer vision-based interactive multi-functional musical instrument, called MuGeVI, which requires no additional hardware circuits or sensors, and allows users to create or play music through different hand gestures and positions. It firstly uses deep neural network models for hand key point detection to obtain gesture information, secondly maps it to pitch, chord or other information based on the current mode, then passes it to Max/MSP via the OSC protocol, and finally implements the generation and processing of MIDI or audio. MuGeVI is now available in four modes: performance mode, accompaniment mode, control mode, and audio effects mode, and can be conveniently used with just a personal computer with a camera. Designed to be human-centric, MuGeVI is feature-rich, simple to use, affordable, scalable and programmable, and is certainly a frugal musical innovation. All the material about this work can be found in `https://yewlife.github.io/MuGeVI/`.

Author Keywords

Digital Virtual Instrument, Interactive Music, Gestural Control, Deep Neural Networks

CCS Concepts

•**Applied computing** → **Performing arts;** Sound and music computing; •**Human-centered computing** → *Interactive systems and tools;*

1. INTRODUCTION

New digital instrument designs can develop new forms of composition and performance for artists, enrich sensory stimulation, and improve educational accessibility. Digital instruments generally require two modules, control surface and sound synthesis, and a mapping strategy for both. Especially, the current advanced human motion tracking technology promotes the development of virtual musical instruments [11]. Gestural controller is widely used in virtual instruments that drive sound synthesis in real time. Due to the real-time requirements for sensitivity, accuracy, and repeatability, most of today's virtual instruments employ sensors to capture gesture information. This requires dedicated circuits and poses a challenge for the popularization of new musical instruments. For example, users who want to experience a new instrument will either have to buy or replicate the same sensor. The inconvenience and extra cost may discourage some users with little interest, thus affecting the popularity of the instrument.

We propose a computer vision-based interactive multi-functional digital virtual musical instrument without dedicated hardware, MuGeVI, which allows users to perform, compose or control music by various hand gestures. MuGeVI is jointly programmed by Python and Max/MSP (Max for short). The Python program processes video frames, detects hand key points, maps the hand gesture information to musical control information, and then sends it to a Max patch through the Open Sound Control (OSC) protocol. The Max patch finally edits the MIDI file or audio and performs sound synthesis.

MuGeVI currently has four modes: 1) Performance mode, which allows the user to play various notes by adjusting the position of hands and completing special gestures; 2) Accompaniment mode, which allows the user to control the scale degree and textures by gestures to accompany the singer or player in real time; 3) Control mode, which allows the user to control the transposition and volume of a track being played through gestures; and 4) Audio effects mode, which provides audio effects for instruments such as electric guitars in real time through changes in finger position. The main innovations of MuGeVI are: 1) No need to use sensors, easy to popularize and apply; 2) Support for both MIDI and audio; 3) Multiple modes switchable at any time; 4) Scalability and programmability.

2. RELATED WORK

A number of works have explored the development of gesture-controlled instruments. Gillian and Paradiso [4] adopted the Kinect depth camera to recognize the tap gestures, hand movements and contractions for instrument controlling. Han, Gold [7], Granieri and Dooley [6] used the Leap Motion sensor to perform new gesture-based instruments or augment the traditional keyboard instruments. Nishida et al. [12] conducted body tracking by the Kinect

*The first two authors contributed equally to this work.
†Corresponding author.

NIME'23, 31 May–2 June, 2023, Mexico City, Mexico.

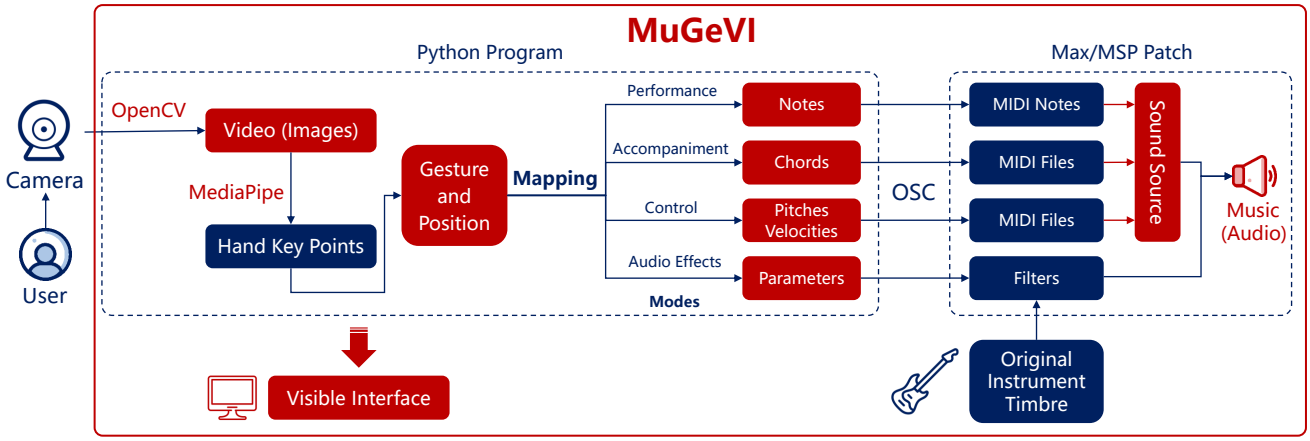

Figure 1: System Architecture. Users input image information through the camera, and MuGeVI outputs music according to different modes. The Python program recognizes hand gestures, maps them to the control information according to the current mode and transmits it to the Max patch, which completes the control of the music and then plays it.

sensor and generated MIDI notes when the hand is detected to move to the corresponding position. [14], [1] and [3] adopted the Myo armband to capture the gestures. Tanaka et al. [14] allowed users to design their own gestures and sounds, and record examples for the training of machine learning models. Brizolara et al. [1] performed meteorological sounds via gesture control. Pozas [3] realized the automated mapping of sensor inputs to MIDI messages. Leonard and Giomi [9] collected data with sensors and armbands, obtained motion features, mapped to physical sound-action features, and synthesize sounds. Lee [8] required the user to hold the smartphone to complete the gesture and thus affected the sound. The trained VGG network performs recognition based on the accelerometer and gyroscope data. Graf and Barthet [5] designed a virtual instrument using mixed reality technology. The left hand pose indicates whether to play a single note or a certain kind of chord, while the right hand plays on the 12 virtual keys. These works use different gesture recognition methods and mapping strategies, but most require hardware devices other than personal computers and regular cameras, and some of these works are relatively single-functional and not very scalable. There are also previous solutions use cameras and computer vision methods to implement camera-based interfaces such as the Very Nervous System [13] created by Rokeby and the EyesWeb project [2] developed by Camurri et al. These works can track body movements and gestures to control or create sound and music, but they are not designed to precisely identify hand key points and complex hand gestures.

3. ARCHITECTURE

This section introduces the overall architecture of MuGeVI and the methods we propose, including image capture, hand key point detection, data transmission, reception and processing.

3.1 System Architecture

Our instrument system, namely MuGeVI, first acquires images of the player continuously through the camera based on the OpenCV library and displays them in real-time, then detects the images using the neural network models and solutions provided by the MediaPipe library to obtain the locations of 21 hand key points. Next MuGeVI will obtain the hand position and gesture based on these key points, map them to the corresponding music information based on the current instrument mode, package the data using the Open Sound Control (OSC) protocol and transmit them to the Max/MSP program using the User Datagram Protocol (UDP). Finally MuGeVI uses the corresponding modules in Max/MSP to implement various functions. All programs except those covered by Max/MSP are written in the Python language. The architecture of MuGeVI is shown in Figure 1.

The camera we use has a resolution of 1080p. The frame rate ranges from 30 to 60 fps in practice with an 11th Gen Intel Core i9-11900 CPU and an NVIDIA GeForce RTX 3060 GPU.

3.2 Real-Time Image Acquisition

Obtaining a real-time view of the player through the camera is the first step of our instrument system. We adopt the OpenCV[1] library which is an open source computer vision and machine learning software library to capture the live images. We create a `VideoCapture` object and get video frames with the `read` function in a loop. Every frame is fed into the neural network models to be processed. The acquired images and the positions of hand key points are displayed in real time by the `imshow` function.

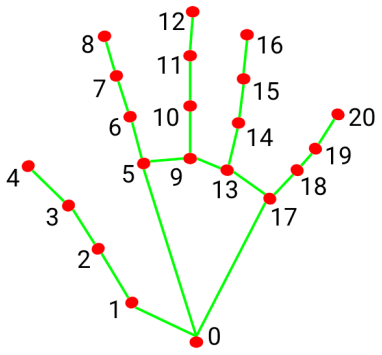

Figure 2: The 21 key points

[1]https://opencv.org

3.3 Hand Key Point Detection

Detecting the position of key points on the hands through computer vision technology is an important function of MuGeVI. Gesture recognition also relies on the location of hand key points. We adopt the solution provided by the MediaPipe[2] which is a TensorFlow-based machine learning library for live and streaming media[10]. It can obtain the positions of 21 key points of the hand, including their horizontal coordinates, vertical coordinates and depths relative to the wrist. The 21 key points are shown in Figure 2.

We instantiate the `hands` class in MediaPipe to realize the detection. It uses two deep convolutional neural network models trained on annotated images, one for the palm detection and the other for key point detection[16]. Both models are lightweight and suitable for use in mobile real-time scenarios. Besides, a strategy that uses the hand key points detected in the previous frame to infer the position of the palm in the current frame is employed, avoiding running the palm detector on each frame and reducing the amount of computation. After obtaining the position of the key points, MuGeVI will infer the gesture according to pre-defined rules, described in detail in section 4.

3.4 Data Transmission

The OSC[15] is a communication protocol designed for use in realtime musical performance. It is a highly accurate, low latency, lightweight and flexible method of communication and is therefore suitable for our new instrument. We package the data obtained from the hand positions and gestures into OSC packets, and transmit them via UDP sockets. The OSC messages contain OSC addresses that follow a URL or directory tree structure, data types such as `int32`, `float32`, `string`, etc. and data arguments. We use the python-osc library to complete the sending of data. In particular, we set up a UDP client via the `SimpleUDPClient` method and send the OSC message through the `send_message` method.

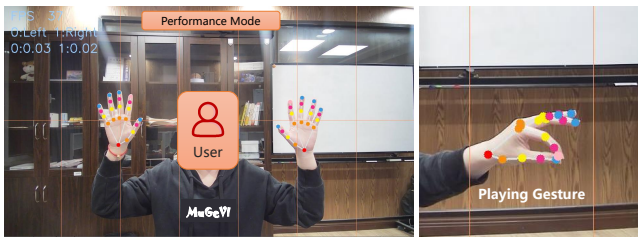

Figure 3: Interface and gesture of performance mode

3.5 Data Processing

Max/MSP is a visual and interactive development environment dedicated to audio and media production. It encapsulates the functions into different objects, which can be used by connecting the modules and setting a few parameters. We use Max to make it easier to modify music meta information, process audio and MIDI and change sound sources. The OSC is a UDP-based protocol, and Max can receive messages through the `udpreceive` module according to the set IP address and port. The message sent by the Python program includes address information and data information, and the Max patch uses the `route` module to determine the mode based on the address and then processes the signal accordingly.

[2]https://google.github.io/mediapipe/

4. FEATURES

This section describes in detail the functions, implementation methods and features of the four modes of MuGeVI, namely performance mode, accompaniment mode, control mode, and audio effects mode. Press `2` to switch to accompaniment mode, `3` to switch to control mode, `4` to switch to audio effects mode, and `1` to switch back to performance mode.

4.1 Performance Mode

Performance mode, also known as air piano mode, is shown in Figure 3. In this mode, the whole picture captured by the camera is divided into several parts according to the spatial position, and each part represents a note. The pitch and duration can be customized. The set gesture for playing the note is for the thumb tip (key point 4) to make contact with the index finger tip (key point 8) then separate, for both hands.

After obtaining the hand keypoints, if the distance between keypoint 4 and keypoint 8 after normalization is less than 0.03 and the time interval between the last transmission is more than 0.3 seconds, MuGeVI will send another note signal to the Max patch. The Max patch generates MIDI note-on and note-off messages with the `makenote` module based on the pitch, duration, and velocity information, then converts them to MIDI format with the `midifomat` module, and finally sends the MIDI signal to the sound source using the `midiout` module. MuGeVI uses the Windows wavetable piano sound source by default, and by changing the sound source, users can play more instruments. Since MuGeVI is highly scalable, the performance can be enriched by modifying parameters such as sending time interval, fingering.

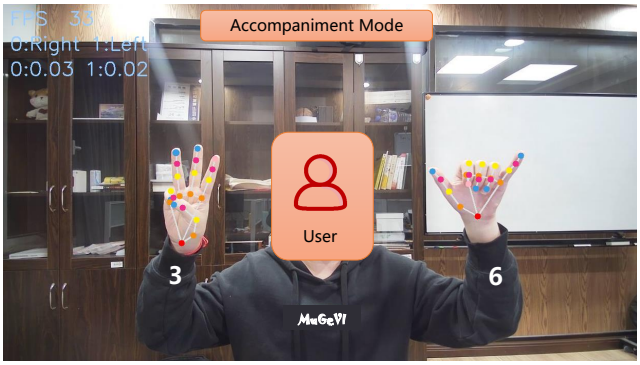

Figure 4: An example of accompaniment mode which indicates the third texture of the submediant chord.

4.2 Accompaniment Mode

In this mode, the user can control the chords with gestures, allowing real-time accompaniment without having to master the chord-playing skills. Besides using the existing chords, users can also edit or upload their own MIDI files for accompaniment. The left hand gesture in Figure 4 is the number 3, indicating the third texture, and the right hand gesture is the number 6, indicating the submediant chord. MuGeVI presets recognizable numbers from 1 to 8, as shown in Figure 5.

MuGeVI determines whether the finger is extended by comparing the distance from the fingertip (key points 8, 12, 16, 20) and the finger root (key points 6, 10, 14, 18) to the palm (key point 0). In practice, the position of the

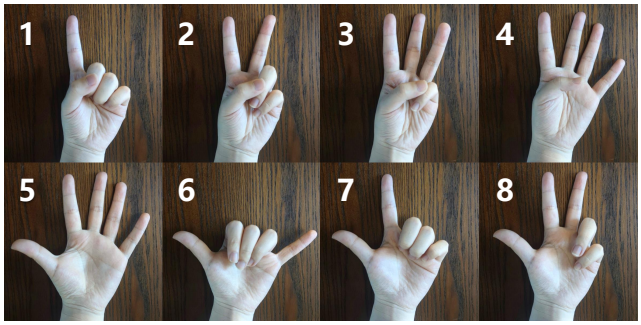

Figure 5: Hand gestures for the numbers 1 to 8

thumb interferes a lot with the identification of 1 and 7, 2 and 8, so MuGeVI connects key point 13 and point 0 in a line and compares the distance from point 2 and point 4 to the straight line to determine if the thumb is extended, as shown in Figure 6.

After recognizing the number represented by the gesture, the python program transmits it as a control signal to the Max patch, which selects the corresponding MIDI file and plays it according to the control signal. To improve the sense of use, the accompaniment mode solves three key problems: 1) Real-time file reading. MIDI files are regularly named by texture and scale degree, allowing precise file selection while supporting multiple texture types. 2) Snap to the right rhythm. The BPM of all MIDI files is set to 120 and the time signature to 4/4. To avoid disturbance of the accompaniment process by the control signals constantly sent by the Python program, MuGeVI sets the chords to switch once in half or a bar in Max. 3) Immediate response to gestures. The ideal situation for the accompaniment is to start the accompaniment as soon as the gesture is recognized and to play it completely as the first chord for half or a bar. Therefore MuGeVI sets an additional boolean variable that is set to 0 every 3 seconds, and the timing in 2) starts only when the variable value is 1.

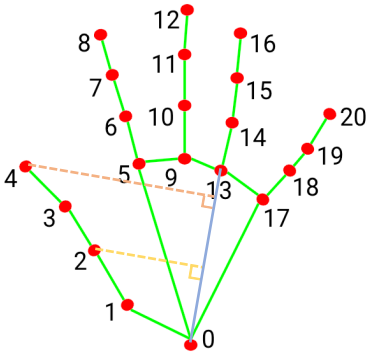

Figure 6: Diagram for determining whether the thumb is extended

4.3 Control Mode

The control mode only uses the position information of the index fingertip (key point 8), and the visualization interface is divided into several areas vertically, as shown in Figure 7. The middlemost part indicates the original key, and the upward and downward parts indicate the ascending and descending keys respectively, changing one semitone at a time.To the left means volume down, to the right means volume up. The control mode provides great convenience for transposition and volume adjustment of songs, for example, when singing songs that do not fit the singer's range.

The Python program detects the coordinates of key point 8, determines which area it belongs to vertically and its relative position horizontally, and then quantifies it into pitch and velocity information and transmits it to the Max patch via the OSC protocol. The three key parts of the control mode are: 1) Controllable playback. MuGeVI sets the key `p` to play and `o` to stop. The python program determines whether a key is pressed and sends a switch control variable to the Max patch to control the playback and stopping of the music. 2) Real-time transposition. the python program determines where key point 8 is located and transmits the number of semitones to be transposed to the Max patch, where the value is added to the MIDI pitch. 3) Real-time volume control. The horizontal slide of the index fingertip is similar to the volume control bar, and the Max patch receives the transmitted velocity information and controls the volume level via MIDI controller No. 7.

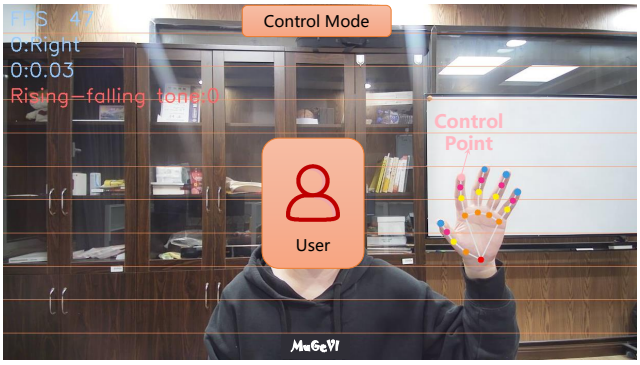

Figure 7: Interface and gesture of control mode

4.4 Audio Effects Mode

This mode provides audio effects to the instrument in real time through finger position changes, and we take wah-wah effect as an example. Different from common foot-pedal wah-wah effects and preset plug-ins, MuGeVI uses gestures for flexible control and easy parameter adjustment (fewer parameters and visual adjustments). Unlike the other three modes that operate on MIDI, this mode processes audio. Note that this mode is an auxiliary mode for instrument playing, so it is used when the player already has an instrument and an audio interface (which converts the instrument's analog signal into a digital signal for computer processing), and no additional hardware is needed for the effects themselves. This mode maps the change in the position of the index finger tip (key point 8) to the change in the center frequency of the filter to achieve the wah-wah effect, as shown in Figure 8. After the Max patch receives the current location change information from the Python program,

Figure 8: Interface and gesture of audio effects mode

it converts it into an audio signal using the `sig~` module in order to monitor its spectrum, and uses the `ramsmooth~` module to smooth the discrete, small-range signal to achieve a better result. It is then converted back to numbers by the `snapshot` module and mapped to a appropriate range by the `zmap` module before fed into the `filtergraph~` module. In this way the changes of finger position can change the center frequency of the filter. The output of the filter is then applied to the input instrument sound and played back, providing a wah-wah effect for the instrument in real time. Additional effects can be implemented by adding or changing filter parameters such as filter type, resonance peak value.

5. CONCLUSION

The MuGeVI proposed in this paper is a gesture-controlled multi-functional digital virtual instrument with four different functions without the need for dedicated hardware devices. MuGeVI combines artificial intelligence technology with electronic music, completing computer aspects such as gesture recognition and data transmission and music aspects such as MIDI file design and editing, sound synthesis, and audio effect creation, and has been tested in all four modes. MuGeVI has certain advantages in usability, frugality, richness, and scalability.

5.1 Evaluation

By inviting participants, we got some user feedback, summarized as follows:

1) Delay and jitter. Users indicated that they basically did not feel any delay between the gesture change and hearing the sound, and there was no jittering situation where one gesture caused two notes to be played repeatedly, which indicates that our system is stable.

2) Functionality. Users indicated that MuGeVI has a variety of functions, which are very practical and interesting. Users would like to support more free rhythm and style types in accompaniment mode, and faster and smoother changes in effects mode.

3) Novelty and convenience. Most users said they had not used a similar instrument and also found MuGeVI to be fun to work with and more convenient and cost-saving than most instruments. In addition, users wanted to be able to use MuGeVI on mobile devices such as cell phones.

Detailed evaluation results can be found here[3]. In the future we are considering evaluations in real performance and music education scenarios.

5.2 Future Work

In the future, MuGeVI is expected to expand in the following areas:

1) Adding facial expression recognition and control;
2) Adding control of drums, bass and other tracks;
3) Adding multiplayer gesture control;
4) Implementing more audio effects.

6. ACKNOWLEDGEMENT

We would like to express our gratitude to:

1) Hao Liu from Central Conservatory of Music (CCOM) for his instruction on this work;

2) Ziao Mu from CCOM for providing the equipment and participating in the video recording;

3) Yuqin Liu and Yubo Gao from CCOM for participating in the video recording.

This work was supported by 22VJXG012, National Philosophy and Social Science and 2022DMKLB003, Key Laboratory of Intelligent Processing Technology for Digital Music, Ministry of Culture and Tourism.

[3]https://yewlife.github.io/MuGeVI/

7. ETHICS STATEMENT

All participants agreed to use MuGeVI and committed to give honest feedback. There was no material waste in this work.

8. REFERENCES

[1] T. Brizolara, S. Gibet, and C. Larboulette. Elemental: a gesturally controlled system to perform meteorological sounds. In R. Michon and F. Schroeder, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 470–476, Birmingham, UK, July 2020. Birmingham City University.

[2] A. Camurri, S. Hashimoto, M. Ricchetti, A. Ricci, K. Suzuki, R. Trocca, and G. Volpe. Eyesweb: Toward gesture and affect recognition in interactive dance and music systems. *Computer Music Journal*, 24(1):57–69, 2000.

[3] V. de las Pozas. Semi-automated mappings for object-manipulating gestural control of electronic music. In R. Michon and F. Schroeder, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 631–634, Birmingham, UK, July 2020. Birmingham City University.

[4] N. Gillian and J. A. Paradiso. Digito: A fine-grain gesturally controlled virtual musical instrument. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Ann Arbor, Michigan, 2012. University of Michigan.

[5] M. Graf and M. Barthet. Mixed reality musical interface: Exploring ergonomics and adaptive hand pose recognition for gestural control. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, The University of Auckland, New Zealand, June 2022.

[6] N. Granieri and J. Dooley. Reach: a keyboard-based gesture recognition system for live piano sound modulation. In M. Queiroz and A. X. Sedó, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 375–376, Porto Alegre, Brazil, June 2019. UFRGS.

[7] J. Han and N. Gold. Lessons learned in exploring the leap motion(tm) sensor for gesture-based instrument design. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 371–374, London, United Kingdom, June 2014. Goldsmiths, University of London.

[8] M. Lee. Entangled: A multi-modal, multi-user interactive instrument in virtual 3d space using the smartphone for gesture control. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Shanghai, China, June 2021.

[9] J. Leonard and A. Giomi. Towards an interactive model-based sonification of hand gesture for dance performance. In R. Michon and F. Schroeder, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 369–374, Birmingham, UK, July 2020. Birmingham City University.

[10] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. G. Yong, J. Lee, W. Chang, W. Hua, M. Georg, and M. Grundmann. Mediapipe: A framework for building perception pipelines. *CoRR*, abs/1906.08172, 2019.

[11] A. Mulder. Virtual musical instruments: Accessing the sound synthesis universe as a performer. In *Proceedings of the First Brazilian Symposium on Computer Music*, pages 243–250, 1994.

[12] K. Nishida, A. Yuguchi, kazuhiro jo, P. Modler, and M. Noisternig. Border: A live performance based on web AR and a gesture-controlled virtual instrument. In M. Queiroz and A. X. Sedó, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 43–46, Porto Alegre, Brazil, June 2019. UFRGS.

[13] D. Rokeby. Very nervous system. `http://www.davidrokeby.com/vns.html`, 1986.

[14] A. Tanaka, B. Di Donato, M. Zbyszynski, and G. Roks. Designing gestures for continuous sonic interaction. In M. Queiroz and A. X. Sedó, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 180–185, Porto Alegre, Brazil, June 2019. UFRGS.

[15] M. Wright and A. Freed. Open soundcontrol: A new protocol for communicating with sound synthesizers. In *Proceedings of the 1997 International Computer Music Conference, ICMC 1997, Thessaloniki, Greece, September 25-30, 1997*. Michigan Publishing, 1997.

[16] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C. Chang, and M. Grundmann. Mediapipe hands: On-device real-time hand tracking. *CoRR*, abs/2006.10214, 2020.