Gestural musical interfaces using real time machine learning

by

Sai Sandeep Dasari

B.E., Osmania University, 2015

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2018

Approved by:

Major Professor
Dr. William H. Hsu

Abstract

We present gestural music instruments and interfaces that aid musicians and audio engineers to express themselves efficiently. While we have mastered building a wide variety of physical instruments, the quest for virtual instruments and sound synthesis is on the rise. Virtual instruments are essentially software that enable musicians to interact with a sound module in the computer. Since the invention of MIDI (Musical Instrument Digital Interface), devices and interfaces to interact with sound modules such as keyboards, drum machines, joysticks, mixing and mastering systems have been flooding the music industry.

Research in the past decade gone one step further in interacting through simple musical gestures to create, shape and arrange music in real time. Machine learning is a powerful tool that can be smartly used to teach simple gestures to the interface. The ability to teach innovative gestures and shape the way a sound module behaves unleashes the untapped creativity of an artist. Timed music and multimedia programs such as Max/MSP/Jitter along with machine learning techniques open gateways to embodied musical experiences without physical touch. This master's report presents my research, observations and how this interdisciplinary field of research could be used to study wider neuroscience problems such as embodied music cognition and human-computer interactions.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Pursuing this field of research was both emotionally and intellectually satisfying experience for me. I would like to thank my major professor Dr. William H. Hsu (Department of Computer Science) for his continued support and his knowledgeable mentorship throughout my Master's program. I would also like to thank my research professor Dr. Carlos Castellanos (Department of Art) for inspiring me to pursue interdisciplinary research and being a support outside my major. Lastly, I'd like to thank my committee/course professors and my family, friends for being a supportive community.

A special word of appreciation to everyone in the local music community at KState and Manhattan, KS that engaged in constructive criticism and made me enjoy my study and music here.

Preface

The idea to pursue this project of building virtual instrument came from my experiences of recording and producing music from a small bedroom. I have always had a passion for anything to do with music. My Master's program at Kansas State University and my research experiences working with Department of Arts pushed me to deeply understand the science and technology behind musical instruments, mixing and mastering. I took a keener interest towards audio engineering and machine learning and eventually that culminated in an interdisciplinary research featuring audio physics, arts, machine learning, embedded microcontrollers and neuroscience.

This report presents my methodology to build gestural virtual music instruments, the experiments designed, the observed results and further research in related fields. Chapter 1 introduces the idea of history of MIDI, electronic music and virtual instruments. It explains the evolution of music and the need for smarter interfaces to express ourselves, further reinforces the idea of teaching your interface to do more than one purpose (playing, mixing or mastering). Chapter 2 dives into similar research in the recent past and in other related fields i.e. human-computer interaction, machine learning, audio Engineering, and electronic music. I also present my previous attempts to work on similar projects for a course project (CIS 721) and how I plan to improve on the project. Chapter 3 presents my holistic idea to bring musical interfaces controlled by gestures on a user's laptop running a Digital Audio Workstation such as Ableton Live. This chapter talks about how the instrument can be put together with hardware, software programming, gathering data, training the

machine learning model and using the real time predictions. Chapter 4 presents the actual experiment conducted in MAX/MSP and Ableton Live using a Leap Motion controller, individual controls achieved on the console and how I interfaced a WiiMote with accelerometer readings to define and learn new gestures. Chapter 5 presents the results of the experiment, feasibility and accuracy of the model and how intensive the CPU usage is.

Through the course of the report, I have actively experimented with multiple interfaces including Arduino, Raspberry Pi and game controllers. I have also tested various arrangements within the program to set up MIDI note triggers and control changes on the DAW. I will be primarily reporting on the simplest one that I felt is a quick and easy instrument to master. The complex versions take more practice to master, but have a wide ocean of sound synthesis potential making them ideal for other forms of audio such as film scoring, visual experience audio design, experimental art installations and neuroscience. This report lets me proudly present all my hours of passion and research about new ways to create musical interfaces and experiences.

Chapter 1 - Introduction

Sound is one of the essentials of human experience and recorded sounds i.e. audio lets us communicate with each other anywhere around the world, which makes it a very important form of evolution through the years. Music has been a universal language for centuries now and has been exponentially evolving.

The invention of recording sciences in the 20th century gave us a way to store or share sound and music on a portable device. What followed was the beginning of a whole new industry of music. These recording and sharing services are really mediation technologies that we have invented for ourselves to express art. Over the past 60 years, there have been a huge number of new instruments, interfaces, recording, signal processing techniques. A good majority of these inventions are based on analog electronics and signal processing. With growing processing power and robust computing facilities, there is active research around the world in the music industry to emulate acoustic and analog sounds digitally. The work in this field opens an ocean of unexplored sounds that I could generated digitally. Computer-based technologies such as digital signal processing, machine learning, human-computer interaction, and audio synthesis open creative paths for artists, musicians and audio/game engineers to generate and use sound innovatively.

Creatively tying down all the above powerful tools, we can come up with elegant processes to build virtual instruments and design music for a wide range of experiences. One such application that is widely researched and developed as we speak is gestural musical interfaces. As with anything related to audio and music,

there is no single solution that works for every musician or audio engineer. Gestural musical interfaces are mere brushes for an artist to express himself. They can only be designed to be used in an abstract way. What the artist/musician does with the interface is purely subjective and creates the real essence of music in general.

Gestural musical interfaces usually use some type of sensor to read the environment and generate changes in the program to shape sound. The program essentially polls the user's actions and generate responses in the program. Actions are actively processed and recorded into the program so they can be recognized by the program. These actions or recognized gestures are varied and can be used to trigger any parameter in general according to the artist. Marc Leman (2008) in his book [1] talks about how the action-based interface and gestural interfaces bridge the gap between physical and mental perception of sound. Embodied music cognition is a major field of neuroscience that studies the role of human body in relation to all musical activities.

Gestural interfaces give an opportunity to let the artist explore sound and sound parameters in their own space via creative gestures. Like any other interface or instrument, however a gestural interface also has obvious limitations. By increasing the number of sensors and how they interact with a human body, a complete embodied musical experience can be created.

In this research experiment, I focus on the hand gestures of an artist to create a flexible instrument to play, control and mix music parameters in a Digital Audio Workstation. All the sound generated is virtual and there is no physical instrument

to touch. The actual design of the interface was inspired from classic USB MIDI keyboards made by AKAI. The instrument sends out MIDI triggers, control changes, pitch bends and any other MIDI data on specific channels to the DAW. Any DAW like Ableton Live is flexible in mapping the buttons in whichever way possible. The mapping of the triggers usually depends on the artist and what the artist wants to achieve ranging between performance, production, arrangement, mixing and mastering. The instrument also features easy to use code modules that let the user train gestures that they would want to use for a specific song or a show. Since all the modules of the instrument is strictly computer generated, the gestures and settings can be easily tucked into presets for convenient usage later.

While these gestural interfaces do pose a steep learning curve and a new way to look at music, they do unleash creative musical experiences and can easily be used to generate alternative sounds such as film scoring, meditative audio therapy, dance generated music etc, if not the regular commercial music. They also give us a potential tool to study and understand neuroscience problems such as embodied music cognition and human-computer interaction to make art.

Chapter 2 - Background and Related Work

2.1 History of electronic music interfaces

2.1.1 Theremin (1920)

Touch free music instruments go back to as early as the 1920s when Leon Theremin, a Russian and soviet inventor used a variable capacitor circuit to control pitch and volume of an oscillator to generate amplified sound. This instrument was totally hands free as it needed no physical contact. The hands acted as ground plates in the variable capacitor circuit. The distance of the hands from 2 antennae determine the pitch and volume. The Theremin was very successful and used in many major scores, the most famous score being the *Star Wars* theme song. Inventors around the world over the course of next 100 years have been continuously evolving and developing models using similar circuits such as light sensitive resistors, sonar rangers or infrared distance detection.

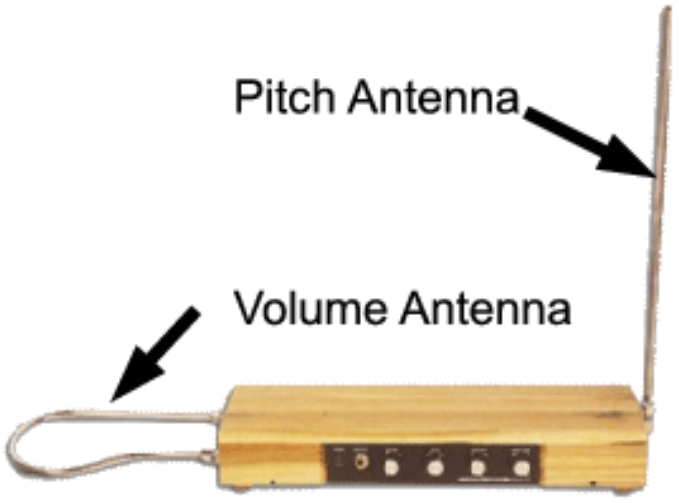

Figure 2.1 A modern Theremin design used in the classic *Star Wars* theme

2.1.2 MIDI (1980-1983)

In the early 1980s, there were wide variety of electronic music instruments and interfaces already manufactured by different companies and being used. However, there was no standardized means of synchronizing them. This need for a universal communication standard pushed a bunch of audio engineers and technology enthusiasts to come together and build a standardized protocol called MIDI or Musical Instrument Digital Interface. The MIDI protocol created opportunities for many other companies to develop innovative interfaces such as the sequencer, sampler, DJ console, looping machines, mixing consoles and mostly recently MIDI interfaces to control parameters within a program. MIDI stood the test of time and invention, improved speeds and utilities through USB and now even wireless. Open Sound Control (OSC) [2] is a protocol for wireless communication between electronic instruments over network ports in the form of UDP packets. Simply put, OSC is the wireless version of MIDI and can be used to develop innovative and portable MIDI technologies. OSC is still being actively developed by CNMAT, UC Berkeley.

MIDI is a huge subject of study in itself, interested reader is referred to “The Complete MIDI 1.0 specification” [3].

Figure 2.2 Akai APK Mini MIDI controller

The gist of the MIDI standard can be summarised by Table 2.1 (below), consisiting of MIDI notes and how they can be related with frequency of sound. Any MIDI note ranges between 0 – 127. The table presents only one octave of the notes. Ever since, MIDI has been actively used in pretty much every gaming controller, music interface or wireless device to transmit and receive data from the device.

Table 2.1 MIDI notes and their relationships to different names and frequencies

MIDI note number	Key number (Piano)	Note names (English)	Frequency (Equal tuning at 440 Hz)
48	28	C3	130.81
49	29	C#3/Db3	138.59
50	30	D3	146.83
51	31	D#3/Eb3	155.56
52	32	E3	164.81
53	33	F3	174.61
54	34	F#3/Gb3	185.00
55	35	G3	196.00
56	36	G#3/Ab3	207.65
57	37	A3	220.00
58	38	A#3/Bb3	233.08
59	39	B3	246.94
60	40	C4	261.63

2.2 Hardware and software applications

2.1.3 Audio engineering and software

Once a robust communication protocol for music interfaces is set up, embedded computers and microcontrollers burst onto the scene. Inventors came up with small portable devices with microcontrollers and circuits built in that could generate sound or transmit data to another computer.

Taking it further, programmers began to develop software being developed specifically to edit music such as Digital Audio Workstations, or mix and play music

(DJing software), music composing software, and MIDI plugins.

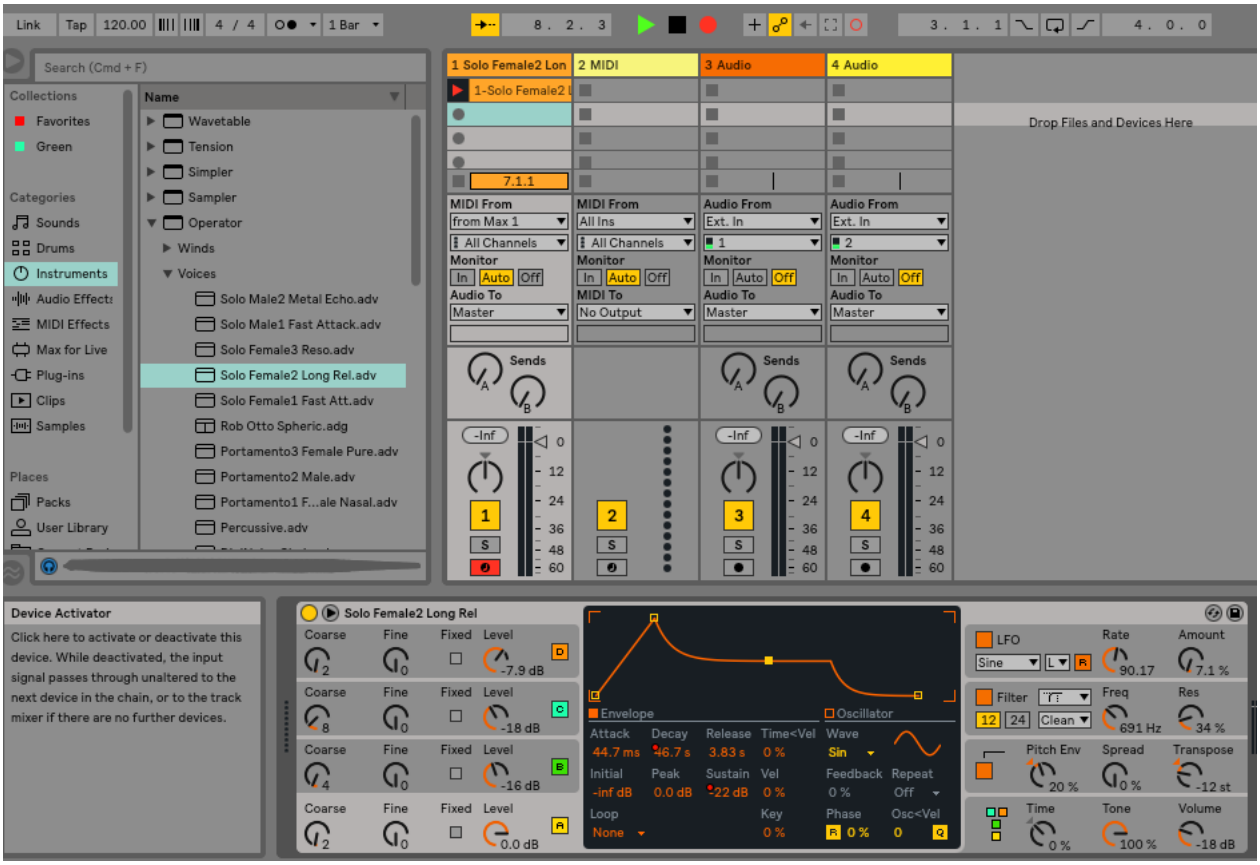

Figure 2.3 A screenshot of workflow in Ableton Live

Among all the developed software for computer music, one category that stood apart is that of audio programming languages such as Max/MSP, Pure Data, ChucK, SuperCollider. These programming languages bridged the gap between audio engineers, electronic engineers and programmers to come together and develop complete instruments that serve multiple purposes.

Max/MSP/Jitter: Max is a visual programming language for music and multimedia developed and maintained by Cycling'74. The program is modular with most routines

existing as shared libraries and makes it very easy to create shareable modules or abstractions. Understanding the purpose of building a programming language to control music and multimedia is key to developing efficient and usable interfaces. The programming approaches in Max are modular and very well thought out right from the beginning as written in detail by the inventor, Miller Puckette, in [4] and [5].

Cycling'74 is known for their community of developers which is a major part of surviving the competition through the years.

Max can be viewed as a dataflow programming language as the patch cords visually translate data between the modules and sub modules. The design was inspired from patch cords used in modular analog synths built in the 70s and 80s.

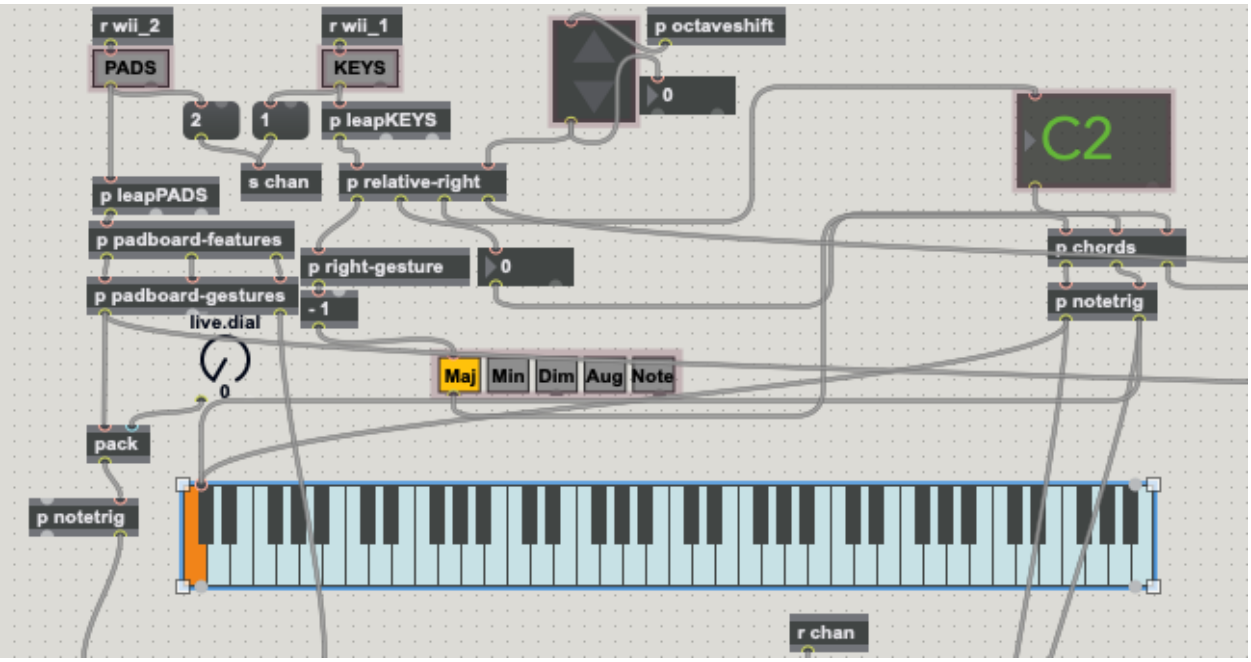

Figure 2.4 Programming patch in Max/MSP

A wide range of sensors and hardware controllers are easy to set up and communicate with Max/MSP through MIDI or OSC. Once the data is received by Max, it can be processed just like any other programming language. The processed data is used to create triggers in an instrument or interface to achieve virtually anything that a regular controller can do. The flexibility and the ease of development makes Max a powerful and efficient tool for musicians, performers, art enthusiasts and most importantly audio engineers / programmers.

Jitter is a matrix of data used to create pixels of visualization from within Max/MSP. Jitter is actively used to create visual interfaces for the program.

2.1.4 Sensors and microcontrollers

The key to building virtual instruments or specifically touch free music is robust sensors that capture information about the musician and relay it to the processing modules in real time. The synthesizers use the buttons of the keyboard as sensors. Sensors from other sciences such as heartrate sensors, piezoelectric sensors, ultrasonic rangers, light sensitive resistors are all actively used to capture data to build an instrument.

To build gestural interfaces, there is a strong need to capture motion data from the user. The problem of capturing motion data expands to many other fields of technology. Powerful cameras form a major portion of these sensors. Flexible stress

sensitive hand sensors are an expensive but strong contenders to capture gesture data from human hands.

Leap Motion: Leap Motion is a computer hardware sensor that supports hand and finger motions as input and has seen active applications in human-computer interactions, hands free graphical user interface, hardware movements, virtual reality, gaming etc. Leap Motion uses simple IR cameras to track finger joints and hands. Being a relatively cheap option for tracking data, a rising number of researchers use Leap Motion as the tool to track fingers and hands to create interfaces.

Figure 2.5 Hardware of a Leapmotion

The Leap Motion sensor has 2 cameras and 3 infrared LEDs that can track infrared light with a wavelength of 850 nanometers, which is outside the visible light spectrum. Also, the interaction area is about 8 cubic feet and can be seen as an inverted pyramid in Figure 2.5 below.

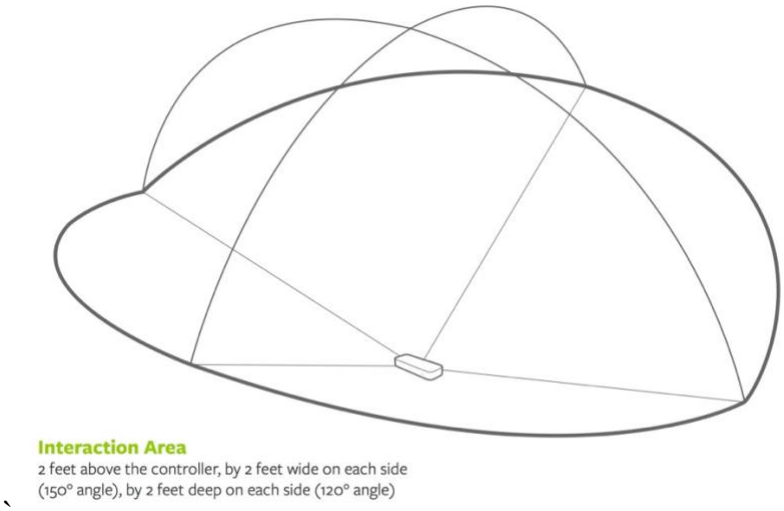

Figure 2.6 Interaction Area

Wiimote: The WiiMote is the primary controller for Nintendo Wii console. The Wiimote is mostly known for the motion sensing capability, which allows users to interact with and manipulate items on screen via gesture recognition and pointing using accelerometer and optical sensor technology. Although it was primarily made only to be used as a game controller with the Wii console, developers and hackers quickly found ways to relay Wiimote messages on OSC and MIDI. The accelerometer readings and optical sensor technology made it ideal to collect continuous data to train models and recognize gestures.

2.3 Hand Gesture Recognition

Hand gestures are simple movements of the hands. Gestures can be classified into

1. Static Gestures: Gestures that do not take the movement of the hands into account and are solely dependent on the static positions of the fingers or joint relative to each other.
2. Dynamic Gestures: Free flowing movements of the hands over a short time creates a complete dynamic gesture.

This experiment does not research dynamic gestures, as they are a much difficult problem to solve. Also, the dynamic movements would compromise the pitch and velocity of the note held. Instead, this experiment focuses on using static gestures and simple motion tracking on a virtual scale laid out in front of the user.

Using sensors, hand movements can be tracked accurately over time. The sensors create streams of data continuously at each frame of view. The collected data can be processed and pattern matched against an already recorded gesture. Gestures can be recognized either through static machine learning algorithms or dynamically using machine learning.

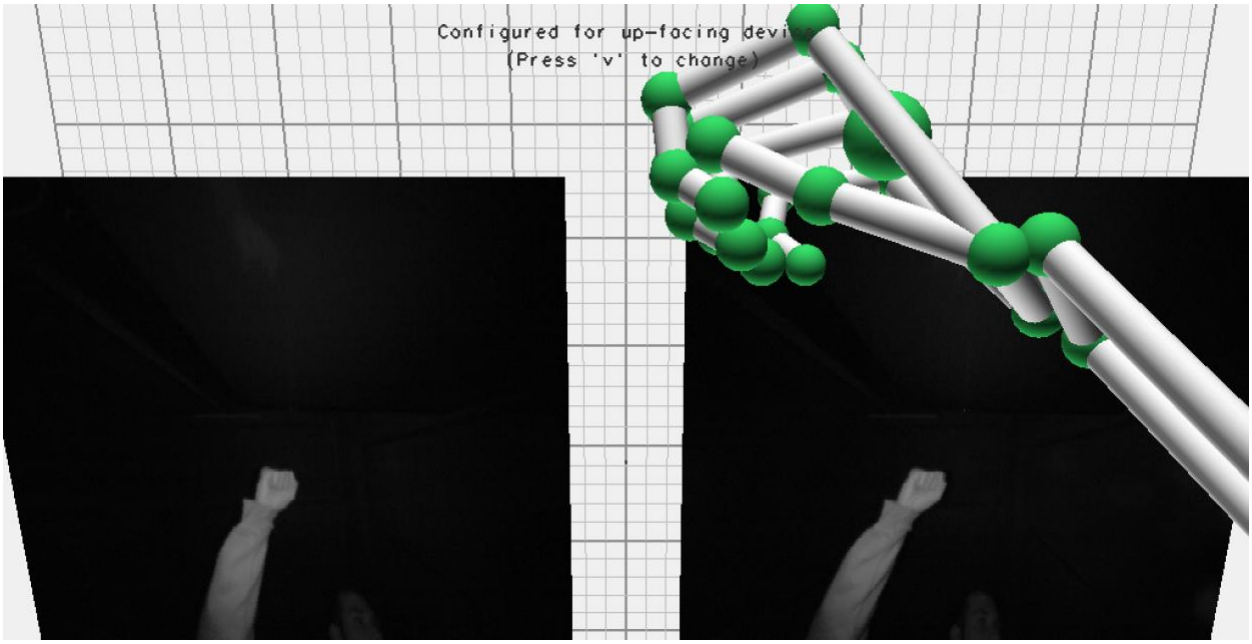

Figure 2.7 Simple gesture of a closed fist

2.3.1 Recognize static gestural patterns

Certain gestures such as pointing an index finger or a closed fist can be easily programmed to be recognized using features of the sensors. These patterns are not complex to recognize as they have a definite form of occurrence. For example, a closed fist always has the same distances between fingers no matter how the hand is held. It is possible to design an algorithm to recognize such simpler gestures robustly and quickly without compromising on processing power for quick changes during music. These easy to track and simpler gestures should be used to control parameters that need quick feedback and precise control. For vague or complicated gestures, machine learning techniques should be used.

2.3.2 Machine Learning

Quoting Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence from 1959 "Machine learning is a field of artificial intelligence that uses statistical techniques to give computer systems the ability to 'learn' from data, without being explicitly programmed." Machine learning requires data in large amounts to show the ability to learn. Although, machine learning and statistics have been around for active research from the past 60 odd years, applications of machine learning in music have been rare, except a few experimental art installations. However, with powerful computers that can process and store data streams efficiently, machine learning models have been everywhere including arts and music.

Machine learning with Max/MSP was a novel idea already researched in detail in [6]. ml.lib is an open source max and pure data object for employing a wide range of machine learning algorithms within Max and Pure Data. Specifically, it is a wrapper written around Nick Gillian's Gesture Recognition toolkit [7] in C++.

2.3.2.1 Classification

Precise sensors that capture motion data are critical to the functioning of an efficient machine learning model. In the application of gestural music instruments, an efficient machine learning algorithm is a perfect tool to recognize and map patterns in sensor data as gestures. The problem of recognizing gestures and translating them to musical changes can be viewed as a classification problem in

machine learning. The gestures in this problem are translated into classes to be recognized and the streams of data from the sensors are translated into feature data required to train on.

Basic classification algorithms such as logistic regression, naïve bayes classifier or decision trees work efficiently depending on the feature data. These simpler techniques are ideal as we have complete transparency to understand exactly what the algorithm is doing. However, these basic classifiers are limited in functionality only to linearly separable data. If the gestures involve complex features that are separable only by a non-linear hyperplane, a non-linear method involving some type of transformation needs to be applied to your input dataset. Slightly more advanced techniques such as support vector machines (SVMs) using kernel tricks do a great job in solving classification problems which have a complex relationship between the data and the class of the gesture.

2.3.2.2 Support Vector Machines

Support vector machine is a classification method that works on the principle of fitting a boundary to a region of points which belong to a certain class. It is a powerful algorithm to solve classification problems with complex data. SVMs have been used in for solving gesture recognition problems as early as the paper [17] "Vision Based Static Hand Gesture Recognition using Support Vector Machines".

The SVM only requires data points at the boundaries of a class. Once the boundaries of a class are defined, most of the internal training data is redundant.

These data points closer to the boundaries are called support vectors. The boundary is generally a hyperplane of N-1 dimensions where N is the number of features in the dataset. The actual boundary line or the classifier can be denoted by the equation

$$w^T x + b = 0$$

The boundary line has a thickness or wideness called the margin denoted by 'd'. The two hyperplanes H1 and H2 on either side of the boundary line can be denoted by

$$w^T x + b \geq +1 \quad \text{or} \quad w^T x + b \geq 0 \ for\ d_i = +1 \ \ldots\ldots H1$$

$$w^T x + b \leq -1 \quad \text{or} \quad w^T x + b \leq 0 \ for\ d_i = -1 \ \ldots\ldots H2$$

where w is a weight vector

x is input vector

b is bias

The optimization scheme of the entire algorithm is to maximize this margin. SVM classification models cannot be visualized by humans once the dimensional feature space has more than 4 dimensions. A simple classification plot (using a linear kernel) for 2 dimensional feature space classified into 2 classes is shown below in Figure 2.7.

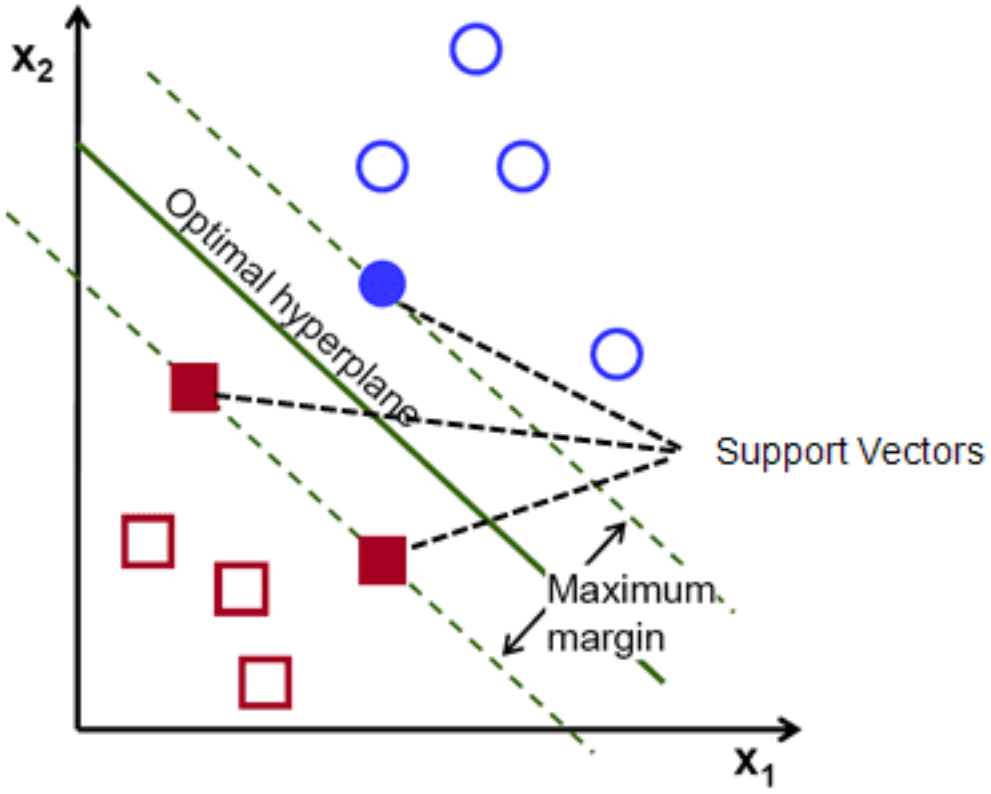

Figure 2.8 SVM classification of a 2 dimensional feature space with 2 classes

SVMs have 3 main tuning parameters that are critical to maximising the margin 'd'

1. Kernel: Choosing a kernel decides how the feature data is transformed for the classification. The idea is to gain a linear separation by mapping the data to a higher dimensional space.

   Linear kernel : $\boldsymbol{f(x) = B(0) + \sum(a_i * (x, x_i))}$

   Polynomial kernel : $\boldsymbol{K(x, x_i) = 1 + \sum(x * x_i)^d}$ *where d is degree*

   Radial Basis kernel: $\boldsymbol{K(x, x_i) = \exp(\frac{-(x-x_i)\hat{}2}{2\sigma\sigma^2})}$
   Where x is the input vector and

x_i is each support vector

2. Regularization (C) : This parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example.
3. Gamma: The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.

To understand SVMs in detail, refer [8]

2.3.2.3 Performance metrics

Generally, machine learning algorithms are verified using performance metrics such as accuracy, precision, recall. Once a machine learning algorithm is run i.e. predictions are made on new feature data with 2 possible classes positive and negative, the predictions each belong to one of the following four classes:

1. true positives TP (number of correctly classified positive examples)
2. true negatives TN (number of correctly classified negative examples)
3. false positives FP (number of negative examples falsely classified as positive)
4. false negatives FN (number of negative examples falsely classified as positive)

Then Accuracy = TP + TN / TP + TN + FP + FN

Accuracy can be misleading if TP and TN are imbalanced. Hence, precision and recall are used to calculate F-measure.

Precision = TP / TP + FP

Recall = TP / TP + FN

F-measure is the harmonic mean of precision and recall. There is often a trade off between recall and precision when data suffer from the class imbalance problem.

F-measure = 2 * precision * recall/ precision + recall

However, when there are more than 2 predicted classes, the performance metrics are generalized for the multi-class scenario and micro and macro averages are used to calculate the precision and recall.

In micro averaging, precision and recall are calculated for individual classes. Assuming a model has k classes,

$$\boldsymbol{Precision_{micro}} = \boldsymbol{TP_1} + \boldsymbol{TP_2} + \cdots + \boldsymbol{TP_k} + \boldsymbol{FP_1} + \boldsymbol{FP_2} + \cdots + \boldsymbol{FP_k}$$

$$\boldsymbol{Recall_{micro}} = \boldsymbol{TP_1} + \boldsymbol{TP_2} + \cdots + \boldsymbol{TP_k} + \boldsymbol{FN_1} + \boldsymbol{FN_2} + \cdots + \boldsymbol{FN_k}$$

In macro averaging, the average of the performance of all the clases are taken

$$\boldsymbol{Precision_{macro}} = \frac{\boldsymbol{Pre_1} + \boldsymbol{Pre_2} + \cdots + \boldsymbol{Pre_k}}{\boldsymbol{k}}$$

$$\boldsymbol{Recall_{macro}} = \frac{\boldsymbol{Rec_1} + \boldsymbol{Rec_2} + \cdots + \boldsymbol{Rec_k}}{\boldsymbol{k}}$$

2.4 Related Work

Brain Opera [9]

Using sensors to translate gestures into machine understandable data is a long-standing problem solved up to a certain extent. However, the exact application of using gestures to interface with a music instrument in real time is relatively new. Joseph A. Paradiso from the MIT Media Laboratory experimented with gestural

sensors for musical interaction and performance very early in 1996. Brain Opera created participatory electronic musical installations using embedded systems, MIDI systems and computers to enable the users to interact with the computers. These installations were built on multiple sensors that are pressure sensors, sonar rangers, touch sensors, tactile sensors and optical trackers. The data from the sensors usually in voltages is converted into Digital using a regular Analog-to-digital converter and fed to MIDI systems that generate a pitch and velocity of a note.

Brain Opera used technologies to generate musical experiences primarily based on electronic circuits and basic MIDI programming. The instruments were elegantly built with piezoelectric sensors to calculate pressure on a carpet, sonar rangers to calculate distances. The resulting sounds from the instruments were arranged to mimic an orchestra of instruments that the audience can interact with. When the Brain Opera was designed and orchestrated in 1996, embedded computing and processing itself was in a nascent stage of development.

Therminal C by WaveLicker

Swiss company WaveLicker develops Theremins that can control analog synthesizers. The design is simplistic in converting analog voltages into CV and feeding it into the synthesizer. Theremins usually control the pitch and amplitude of a wave signal, but this design goes further and controls pretty much anything on the synthesizer such as Pitch, Amplitude, Filters, LFOs, anything that can be controlled by an analog voltage. This project does not feature gestures but forms the basis for

the need of a Theremin-like synthesizer that can control sound parameters within a program (in this case, a synthesizer.)

Figure 2.9 MI.MU musical gloves

MI.MU Gloves [10]

Years later, MI.MU gloves burst on to the scene in 2010 at a Ted talk when Imogen Heap, a musician and technology artist presented a pair of gloves that flexible sensors that calculated the movement of each joint in the hand. The gloves were a combination of textiles, electronics, sensors and software made for a professional artist interested in making complex music. They had multiple versions of the gloves and professional artists around the world enjoyed the relationship between electronic

music and organic hand gestures. The gloves were steeply priced at around 5000$ (excluding the laptop and software) and didn't make it beyond the odd world tour for a professional musician. However, the gloves inspired a new field of gestural interfaces and devices.

These are some of the projects that were actively studied before designing the experiment. The current gestural interface would not be possible without the inspirations and ideas of all the people involved in these projects.

Chapter 3 - Proposed Model

3.1 Data Flow

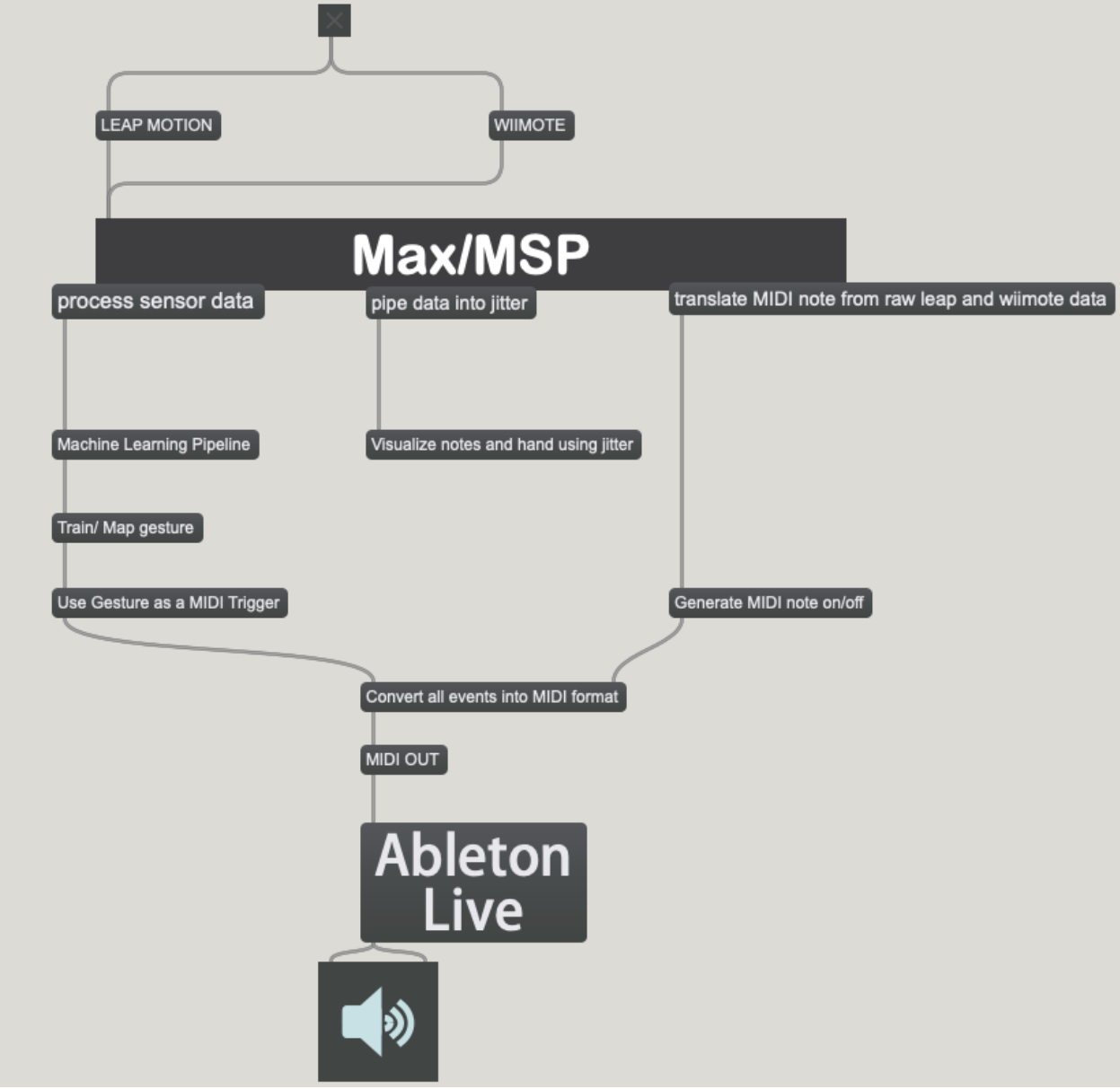

Figure 3.1 Data Flow Chart

3.2 Capturing motion data

The sensors chosen to capture motion data from the user continuously trigger the data flow. A sensor or controller has multiple triggers that can all be simultaneously used to switch and change the way the instrument behaves or creates sound. Having multiple sensors and controllers to capture the data ensures accurate representation of the gestures. Hence, synchronous timing of the sensors is critical to the data flow. Usually the sensors or controllers will broadcast the required data either through USB or OSC on a fixed port number on the network. Data from the sensors can then be broadcasted to more than one program or system to create various musical or visual triggers.

3.3 Receive data and organize training pipelines:

The data received via USB or OSC is processed to either recognize simple gestures or button presses on the sensor or create a training pipeline of data for the machine learning models.

3.3.1 Simple gestures and controller button interfacing

Once the data is received in the raw state, it is pre-processed to separate features of data that can be directly used to interact with the music software. For example, 3-D co-ordinates can be used to control the pitch and volume of the note in real time without relying on any prediction or classification. If a controller has buttons or

wheels, they are separated from the feature data and used as direct triggers to start and stop the playback of sound specifically.

These simple buttons and gestures are a quick and direct way of interacting with the system. Also, as the recognition of these gestures does not require any processing or predictions, the lag is virtually unnoticeable for the user. If a musician or artist needs to immediately mute out a certain sound or module, these first layer of real time MIDI triggers serve the purpose.

3.3.2 Feature Engineering

For the machine learning algorithm to learn and generalize, it is critical that the training data is cleaned, pre-processed and engineered to represent the gesture motion of the hand. Features that do not contribute directly to the formation of a gesture are discarded. Multiple features that can be aggregated into a single feature having more direct relationship with a gesture are formed. Raw sensor readings usually do not have a direct relationship with the gesture class and end up as noisy data for the algorithm.

For example, to detect the gesture of holding a knob or slider, the relevant features would be the relative distances between the fingers, relative distances between fingers and the palm. As the optical sensors can only detect approximate the 3-D coordinates of the joints, specific feature engineering to convert the coordinates into relevant features is done in this stage. The feature engineering stage in any machine learning algorithm defines the efficiency and ability of an algorithm to generalize

well. [11] presents a dataset of feature data and how they can be engineered for efficient use.

3.3.3 Train and Deploy Framework

The feature dataset is now ready to be sent into machine learning model with an appropriate class labelled on each row. In this module, the parameters of the model are tweaked before the training begins. The data is stored in collections available to the model. Training pipelines are always updated if gestures or classes are added or removed. Once the training completes, the model is set into 'map' mode.

During the map mode, new feature data with no class label comes in from the sensors or controllers, and the model makes a prediction in real time and sends the prediction gesture out into the MIDI processing module.

3.4 MIDI processing module

The MIDI processing module is an abstraction that takes in directly recognized gestures, controller values and predictions and acts upon them. In this module, the MIDI triggers or control changes are generated based on the movement of the user. This module can also be referred to as the actual sound generation module in the entire program.

Incoming gestures or controller changes are processed and scaled according to the required need and mapped into a MIDI value. MIDI is always a number between

0-127 on a certain channel. This module can also be used to switch between presets within the program to totally change the functioning of the instrument.

For example, the MIDI processing module can act as a synthesizer or drum sequencer or mixing interface or a digital pedalboard. This module also adds visualization for the triggered notes in the program to give the user active feedback. The module will have a MIDI outlet to transmit the MIDI messages to any virtual instrument or software that can accept MIDI.

3.5 Digital Audio Workstations

Digital Audio Workstations (DAWs) are software programs within a computer that accept MIDI triggers and generate sound or control parameters that shape sound. DAWs are powerful tools to compose, edit and produce music all together. Many companies manufacture DAWs that support MIDI and can interface multiple MIDI interfaces within a computer. Producers around the world use DAWs actively as they are one-stop solution to all the sound engineering. Most DAWs also come with their own range of Audio Effects, MIDI effects and VSTs. Some DAWs also allow users to run free audio applications within the DAW as plugins, making the instrument chain complete with Digital Signal Processing effects.

The DAWs generate sound primarily but also, offer a wide variety of output options from the computer. They offer mixing and mastering options within the program. Controlling the mixing / mastering parameters with the instrument set to

a certain preset will make the production process easier and effective. This experiment chooses Ableton Live as the DAW due to it's compatibility with Max/MSP.

Chapter 4 - Experiment

Based on the model proposed in Chapter 3, an experiment was designed to develop a virtual instrument, interface. Like any other developer, I actively tried and built as many as five prototypes before settling with the current one being presented in this report. Brief outlines of some of my earlier experimental prototypes and how I made changes to facilitate a complete robust playable instrument.

Table 4.1 Experimental prototypes

Attributes of the experimental prototypes				
Prototype	**MIDI/ pure wave**	**Gestures**	**Range**	**Ease of use**
1 Arduino-controlled Theremin with distance sensors (ultrasonic ranger or infrared distance sensor)	Pure wave	No	Narrow range	Easy
2 Leap Motion synthesizer	Pure wave	Yes	Geospatial range	Difficult: navigating notes
3 Leap Motion MIDI	MIDI	Yes	Geospatial range	Moderate: Continuous MIDI trigger
4 Leap Motion MIDI with 2 hands	MIDI	Yes	Geospatial range on 2 hands	Complex
5 WiiMote MIDI trigger	MIDI	No	Limited	Easy

In the above Table 4.1, prototypes and their descriptions are listed along with their attributes of functioning. A pure wave prototype is a kind of synthesizer and does not need an external sound module to generate the sound. Earlier designs had no

gestures or very simple gestures requiring no machine learning or statistical learning. The lack of gestures does not make them useless instruments. In fact, an artist or musician on simple Theremin-like playing will have a great time with the simpler version. Especially because the Arduino is a small and inexpensive portable microcontroller. The latter focus on finding the balance between playing MIDI notes and keep the operation simple and expressive for the user. The last prototype uses a WiiMote only to trigger notes to emulate the gesture of pressing down and releasing the keys of the keyboard or piano.

Learning from these experiments, the final model was proposed in Chapter 3.

4.1 Interfacing sensors with the programming language Max/MSP

Leap Motion is a powerful sensor that has been previously used to solve gesture recognition problems. The sensors tracks hands frame by frame and produces lists of timed data for each joint of the hand, palm of the hand and vector data such as velocity and acceleration. The range of data tracked by the Leap Motion is displayed in Figure 4.1. For classifying static gestures however, all the lists of data are an overkill and can lead to overfitting of the model. The Leap Motion is used in the experiment to control the number and velocity of the note played on a horizontal keyboard apart from the actual gesture recognition. However, using the Leap Motion alone continuously triggers MIDI notes with no note holds.

To overcome this challenge, a controller such as Wiimote is used in the experiment.

The Wiimote can detect button presses on the controller or continuous

accelerometer readings. Wiimote is used to hold and release notes like a traditional keyboard, change settings from the buttons or have simple gestures that control the pitch bend and tremolo of a note held down. This also ensures that no unwanted notes are played while moving across the instrument to find the next note. Using these 2 sensors together creates a robust playable instrument with no errors or mishaps through the music. However, the challenge of synchronizing the sensors to work simultaneously sits with the programming interface.

Max/MSP has direct objects within the programming language for interfacing Leap Motion sensor data. However, Leap Motion discontinued their legacy APIs after version 3. Hence, the experiment will be sticking to the skeletal tracking system in Leap Motion V2.3. The latest version Leap Motion Orion is a much more advanced and robust tracking system for developers, however the added overhead of getting the data into an audio programming language such as Max pushes developers to the legacy versions.

Orion Beta and future versions may solve potential problems of occlusion and faster frame rate, which improves the overall gesture prediction model.

OSCulator: OSCulator is a software that links controllers to a music or video softwares using the Open Sound Control protocol (OSC from Chapter 2). The ease of use and support for a wide range of wired and wireless controllers makes OSCulator a perfect tool for developers who do not want to reinvent the wheel in the networking or interfacing sensors.

Connecting a Wiimote to Max/MSP is quick and easy. OSCulator broadcasts the wiimote controller data as UDP on a specified port. Max receives the data from the part and triggers the data flow to other modules in the data flow model from listed in Figure 3.1.

4.2 Processing data into pitch, velocity and gestures

Data received from the sensors or controllers is cleaned to remove unwanted lists of data. Simple co-ordinates of fingers and palm are mapped to pitch and velocity between 0-127 to emulate a MIDI note. The horizontal position of the hand generates pitch and the vertical position generates velocity of the note.

For the Wiimote, Max/MSP receives UDP packets and translates the toggle messages from the buttons into simple if-else decisions to trigger direct interactions with the program. The accelerometer readings are used to control simple pitch bends and tremolo based on 3-D vector of acceleration. For example, the Wiimote throws out data about roll, pitch and yaw of the controller when moved in those particular directions. The instrument uses roll to control the pitch bend of the note in both directions i.e. Note can be bent up or down and the pitch is used to control the trigger of a tremolo wave on a note.

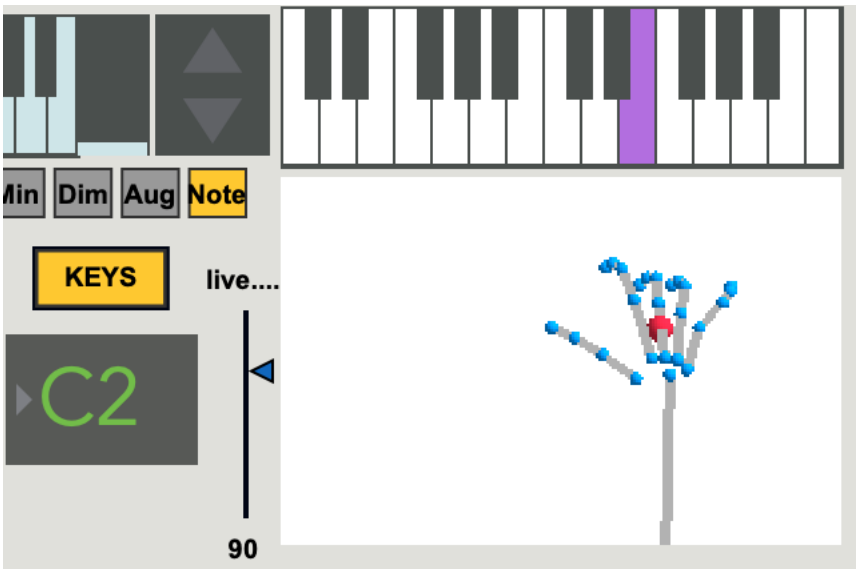

Figure 4.1 Position of palm in X-Y direction generates note

On the other hand, for the gesture recognition system, feature sets are built or engineered depending on the kind of gesture to be recognized. Feature sets containing relevant data are crucial to the working of the machine learning algorithm. Features that are not directly correlated to the gesture are avoided. Instead multiple features can be engineered to create a single feature that is strongly correlated to the gesture. For example, to recognize static gestures to play a major chord, the raw data received from Leap Motion only has the 3-D coordinates of the joints, fingertips and the palm. The raw data do not definitely define a major chord when the hand moves around over the sensor. Instead, if the relative distances between different data point co-ordinates in the system creates a mutual pattern that can be easily recognized by the machine learning algorithm.

4.3 Machine Learning model

4.3.1 Feature Engineering

A simple feature set was proposed that can be used to easily classify static gestures. The feature sets used for recognizing gestures for this instrument are

1. Euclidean distances between fingertips and palm center - 5 features
2. Euclidean distances between two adjacent fingers - 4 features
3. Euclidean distances thumb and the other fingers of the hand - 4 features

Thus, a feature set of 13 features is built and fed into the machine learning model as the training data. A total of 5 gestures are used to interface with the program to create various MIDI effects.

Hence, each of the 5 gestures are individually trained by recording the shapes into the training pipeline. The machine learning model then trains on this set of data to recognize new incoming data as the instrument is played.

4.3.2 Why Support Vector Machines?

To recognize gestures, a classifier is required that can classify complexly related data. Also, the model needs to classify data into multiple features. For this experiment, a moderate number of 5 gestures was chosen. From the section 4.1.1, the feature set representing the data is 13 dimensional dataset, which creates a complex dataset. As the features are already reduced to represent gestures, simpler classification algorithms such as decision trees and logistic regression do a bad job in the classification problem.

SVMs are powerful algorithms that look at the edges or the extreme cases in each class. Thus, enabling the model to classify complex feature spaces. Also, as the number of features here is lesser than the number of samples, there is no efficiency problems in training the model. The algorithm is memory efficient as it only works on the extreme data points or support vectors in each class.

4.3.3 Training and mapping the model

As the experiment is set up with 5 gestures to be recognized, they can seen as 5 different classes. The training data was recorded into the program by performing each gesture for 10 seconds. All the training data was tagged with one of these 5 class labels before the data is sent to the training pipeline. All the training data is stored in a collection for further training pipelines.

Once the training data is set up, the hyper parameters for the SVM model are chosen. As the feature space is a complex 13-D set, the polynomial kernel was picked against the linear kernel as the hyperplane has to generalize efficiently. The SVM model is trained and later set into mapping mode. In this mode, the data recorded from the sensors comes into the model and at each frame, the SVM model classifies the gesture data into one of the 5 classes. These 5 class predictions are then sent into the MIDI processing module where the final mapping of the gestures into MIDI triggers takes place.

4.4 Creating MIDI triggers and Control Changes

Once the prediction is completed and the raw data from the Leap Motion and Wiimote are mapped to a MIDI note, the MIDI processing module in the program finally packs all the midi events into the MIDI out format that an external DAW can understand and produce sound.

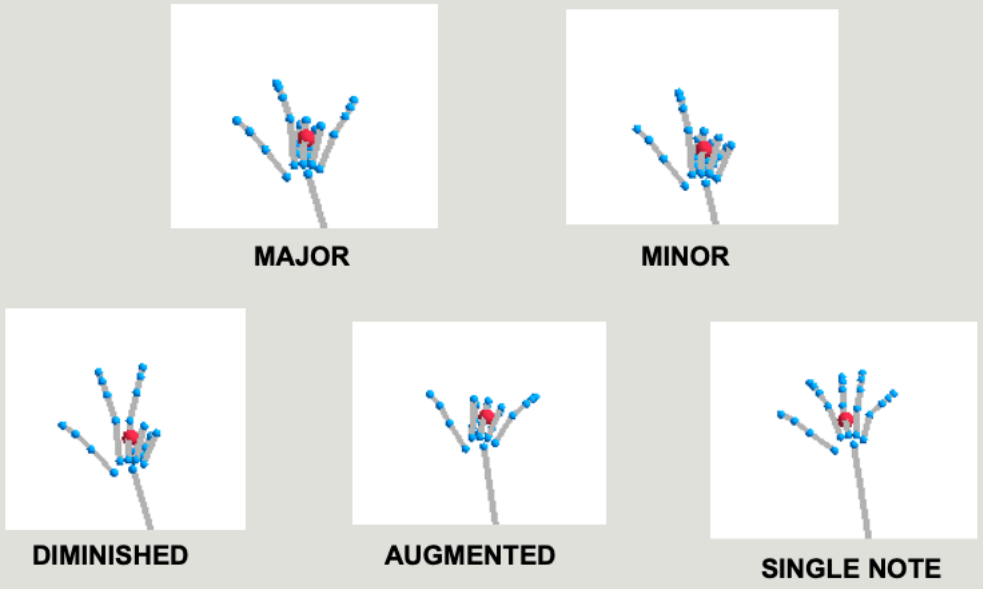

Figure 4.2 Gestures for triads visualized in Jitter

The five gestures recognized by the model can be used to serve a variety of purposes. This experiment chose to use the 5 gestures to convert a played note into either a

- Major triad
- Minor triad
- Diminished triad
- Augmented triad

- Single note

The prediction along with the midi note and the Wiimote note hold produces a MIDI event to play a note in the virtual instrument sitting inside a DAW. However, there are a number of other ways to use this data. For example, the gestures could be mapped in a synthesizer to generate different wave forms while holding a note.

4.4.1 Alternative uses of gestural interfaces.

The usage of the instrument or interface is totally based on the musician and the sole idea came from using MIDI interface boards such as AKAI or Novation which have a keyboard, a pad board, faders, knobs and other similar interfaces. The keyboard and pad boards are essentially simple MIDI note triggers, however the faders, knobs are control changes in MIDI between 0 – 127. These CCs can be used to mix, master and manipulate parameters inside the DAW. An alternate mode for the instrument was set up to play pads and move faders using a specific gesture (a pinch gesture to zoom in) to hold the fader and move it up or down.

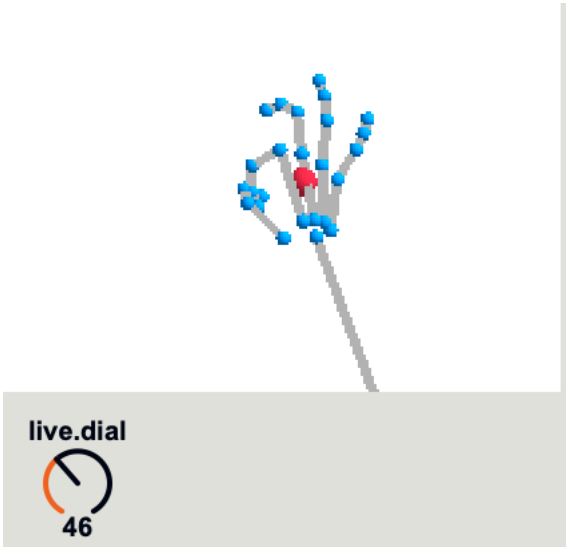

Figure 4.3 Use pinch gesture to create control changes

The interface is very flexible and mappable according to the user's requirements and the idea was to design the device that can be taught how to interpret gestures.

Chapter 5 - Results and Future Work

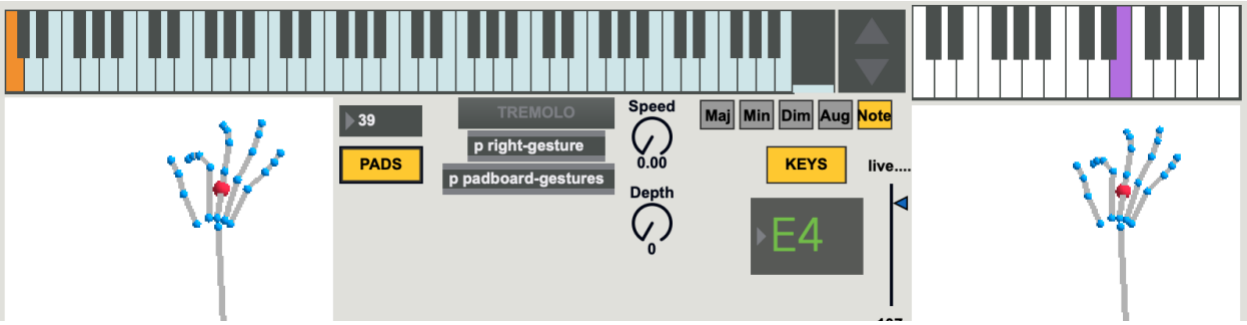

Figure 5.1 Completed gestural interface UI

The completed design has 2 jitter visualizations for a keyboard and a pad surface. There is a tuner, tremolo and a variable vibrato and gestures displayed as triads. Also, there is an octave control and the machine learning modules are packed into sub patches for the interested technical user.

5.1 Results

This sub-section presents the results of the experiment described in the previous chapter. The instrument was evaluated for playability, irregular behavior, ease of navigation and the prediction accuracy of the gesture recognition section.

The addition of the Wiimote to trigger note on and note off makes the system a robust and reliable instrument getting rid of unintended notes. The UI for the system is simple and self-explanatory. Hand movements and gestures are visualized using Jitter, a visual programming module within Max/MSP that stores data and displays the pixel data on a window. Clear and concise note display and the current triad playing defines detailed user experience.

The machine learning algorithm was tested by generating a confusion matrix. Initially the model was trained to recognize the 5 gestures and set in 'Map' mode. In the map mode, the model was tested against the actual gestures being played of a note sheet for simple chord progressions or melodies.

5.1.1 SVM Model Performance

The corresponding confusion matrix as programmed within Max/MSP is shown below in Figure 5.1. The test framework is built in as a module within the program to help users evaluate results. As the problem is multi-classification, the performance measures were adapted by calculating precision, recall and F-score. Refer to Chapter 2.3 for performance metrics used.

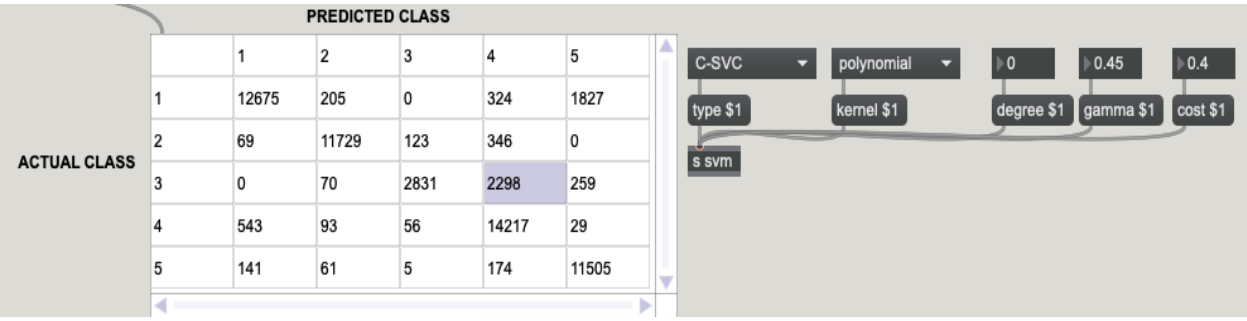

Figure 5.1 Confusion matrix calculated for the SVM gesture recognition model

Micro Averaging:

Precision (micro) = 0.97

Recall (micro) = 0.90

F1- measure (micro) = 0.933

The performance metrics for the micro averaging show the prediction of the SVM model is reliable. (F-1 measure is greater than 0.9, a promising value.)

Macro Averaging:

Table 5.1 Performance metrics for macro averaging

Class	Precision	Recall	F1-measure
1	0.94	0.843	0.884
2	0.964	0.956	0.958
3	0.938	0.518	0.66
4	0.818	0.951	0.876
5	0.844	0.96	0.898

Precision (macro) = 0.90

Recall (macro) = 0.84

F1-measure (macro) = 0.87

Apart from the general results, the experiment posed problems navigating between the black keys on the keyboard as they are placed narrowly on the keyboard laid out horizontally across the width of the Leap Motion sensor. After researching, this was found to be a tracking limitation on the Leap Motion V2.3 and seemed to be fixed in latter versions of the Leap Motion. However, adding the Wiimote to control note on and note offs showed great results in creating expressive music when combined with the tremolo and pitch bend. The combination of these 2 gestures along with DSP plugins within Ableton Live creates amazing sounds ideal for expressive film scoring or instrumental pieces or writing complex music in general.

5.2 Future Work

Applying machine learning in real time programming to generate predictions is not a novel idea, known to researchers. Gesture recognition adds a layer of application to the existing frameworks and makes the field of human-computer interaction futuristic and continuously evolving. However, the applications extend far beyond computers and music to neuroscience in studying embodied music cognition and substitutive senses.

Improve model prediction by using a centralized database:

A low recall and f1-measure can be observed for class 3 as the gesture 3 is very similar to gesture 2. This is a possible area of improvement for the algorithm. After discussion with my masters committee, a deep learning model can be used to improve the accuracy of the model by using data collected from different users. This can be a different design on the whole that collects data from the users over time and trains one major model with more gestures and pre-sets. This is a very interesting idea to focus on in future work.

Going beyond the result oriented research, I came across Marc Leman's book about embodied music cognition. The book inspired ideas to pursue the field of neuroscience and psychoacoustics to understand perception of sound and how we can develop and design musical experiences beyond commercial music. Touch free music instruments and interfaces can improve our tools of expression. Using piezo sensors, force sensitive resistors, heart beat sensors to define rhythms and patterns in music can be used to understand the embodied music cognition. Also, much more expressive

sensors such as the Microsoft Kinect is known for it's motion sensing capabilities more than the gaming console. These tools give us access to data beyond just hands. The Kinect reads the joints of the human body through the IR camera, and the joint data can be used to train and understand complex gestures involving dynamic movements controlling advance sound and light parameters.

Also, there is active current research in substitutive senses i.e. translating sound, light into alternative senses such as haptic touch, or other forms of sensory inputs. This could potentially help the deaf understand sounds. Combined with the right devices, the deaf could listen, perceive and write music. Another application of embodied gestural interfaces is creating the ability for the visually impaired to feel and create music through hand gestures and movements. Proprioception is the ability to sense the orientation of your body in your environment without consciously thinking about your spatialization within the environment. Empowering the disabled to interact and create music or any other form of art is my long term goals in studying and bringing machine learning, human-computer interaction and music technology together.

Chapter 6 - Conclusion

Like any art form, music technology is also subjective. While there are growing arguments about the contemporary and electronic music lacking feeling or expression, any technology behaves the way it is used. Traditionally, orchestras are led or controlled by the conductor. The best performances or pieces are all heavily dependent on the conductor's performance. The gestural interface can be viewed as a tool for the conductor or the modern music producer to control and express himself while also playing sections of the music himself. The instrument was built to allow flexibility of mapping and creating new sounds or even teaching the framework gestures that you want to incorporate in your playing. Experimentation and expression come hand in hand, and are subjective to each musician or artist. The instrument can be used to write musical pieces, score motion pictures or be used art installations to interact sound, light and bodily movements.

The results from Chapter 5 represent the accuracy and reliability of the model. The design along with directly interfacing a physical controller keeps the perfect balance for the user. The machine learning model is set up in a way to allow the user to teach new gestures to the instrument relatively easily or change the mapping in the software. So, a set of repetitive gestures will create different sounds based on the section of the song (verse, chorus or intro).

While there is still the problem of misclassification in rapid movements and the problem of occlusion i.e. Leap Motion does not recognize the hand when it is obstructed partially or the hand goes outside the region, they can be viewed as

physical limitations of the sensors and how expensive the sensors are. Expensive alternatives such as Microsoft Kinect or the MI.MU gloves add much more robust and reliable data. There are other relatively inexpensive and easy-to-setup options such as the source audio Hot Hand which are limited in their functionality. There is always a tradeoff between the cost and the functionality you would want to achieve in technology and the area it will be used as with the Wiimote.

However, the research in this field can be positively expanded to control drones, machines or anything that we can think of controlling with gestures. From the perspective of neuroscience, it can be used to study motor relationships with sound, light and other senses.

Substitutive senses could potentially help physically impaired people to hear, perceive and play music or converse.

Quoting the composer Louis Spohr who watched Beethoven in a rehearsal in 1814:

"In *forte* passages the poor deaf man pounded on the keys until the strings jangled, and in *piano* he played so softly that whole groups of notes were omitted, so that the music was unintelligible unless one could look into the pianoforte part. I was deeply saddened at so hard a fate."

Despite their disabilities, Beethoven and many other artists continued to express themselves and make art. The tools to express ourselves will always continue to evolve technologically and artistically.

References

1. Marc Leman. 2007. Embodied Music Cognition and Mediation Technology. The MIT Press.

2. Freed, Adrian & Schmeder, Andy & Zbyszynski, Michael. (2018). Open Sound Control (OSC) is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology. Compared to protocols such as MIDI, OSC's advantages include interoperability, accuracy, flexibility, and enhanced organization and documentation.

3. The Complete MIDI 1.0 Detailed Specification:

https://www.midi.org/specifications-old/item/the-midi-1-0-specification

4. Miller Puckette. 2002. Max at Seventeen. Comput. Music J. 26, 4 (December 2002), 31-43. DOI=http://dx.doi.org/10.1162/014892602320991356

5. Miller Puckette. 2007. The Theory and Technique of Electronic Music. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

6. Jamie Bullock and Ali Momeni. 2015. ml.lib: Robust, Cross-platform, Open-source Machine Learning for Max and Pure Data. In Proceedings of the international conference on New Interfaces for Musical Expression (NIME 2015). The School of Music and the Center for Computation and Technology (CCT), Louisiana State University, Baton Rouge, Louisiana, USA, 265-270.

7. Nicholas Gillian and Joseph A. Paradiso. 2014. The gesture recognition toolkit. J. Mach. Learn. Res. 15, 1 (January 2014), 3483-3487.

8. Ingo Steinwart and Andreas Christmann. 2008. Support Vector Machines (1st ed.). Springer Publishing Company, Incorporated.

9. Paradiso, Joseph. (1999). The Brain Opera Technology: New Instruments and Gestural Sensors for Musical Interaction and Performance. Journal of New Music Research. 28. 10.1076/jnmr.28.2.130.3119.

10. Claudia Villalonga, Hector Pomares, Ignacio Rojas, and Oresti Banos. 2017. MIMU-Wear. Neurocomput. 250, C (August 2017), 76-100. DOI: https://doi.org/10.1016/j.neucom.2016.09.125

11. Ameur, Safa & Ben Khalifa, Anouar & Bouhlel, Med. (2016). A Comprehensive Leap Motion Database for Hand Gesture Recognition. 10.1109/SETIT.2016.7939924.

12. Matthew Wright, Adrian Freed, and Ali Momeni. 2003. OpenSound Control: state of the art 2003. In Proceedings of the 2003 conference on New interfaces for musical expression (NIME '03). National University of Singapore, Singapore, Singapore, 153-160.

13. Tobias Grosshauser and Gerhand Tröster. 2014. Musical instrument interaction: development of a sensor fingerboard for string instruments. In Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction (TEI '14). ACM, New York, NY, USA, 177-180. DOI: https://doi.org/10.1145/2540930.2540956

14. Maes, Pieter-Jan & Lorenzoni, Valerio & Moens, Bart & Six, Joren & Bressan, Federica & Schepers, Ivan & Leman, Marc. (2018). Embodied, Participatory Sense-Making in Digitally-Augmented Music Practices: Theoretical Principles and the Artistic Case “SoundBikes”. Critical Arts. 32. 1-18. 10.1080/02560046.2018.1447594.

15. Maes, Pieter-Jan & Nijs, Luc & Leman, Marc. (2018). A Conceptual Framework for Music-Based Interaction Systems. 793-804. 10.1007/978-3-662-55004-5_37.

16. Caruso, Giusy & Coorevits, Esther & Nijs, Luc & Leman, Marc. (2016). Gestures in Contemporary Music Performance: A Method to Assist the Performer’s Artistic Process. Contemporary Music Review. 1-21. 10.1080/07494467.2016.1257292.

17. Naidoo, S & Omlin, Christian & Glaser, Meryl. (1999). Vision-Based Static Hand Gesture Recognition using Support Vector Machines.