“Hand Pattern Recognition for Orchestra Conductor using $P-Recognition”

GUIAS, KIRT JAN EAU, Ateneo de Davao University
SUMATRA, JOSE ANGELO, Ateneo de Davao University

Abstract: Technology in its rapid progression easily changes the people’s lives making them more convenient. In the music scene where technology has a big role, we can rarely see systems that can control and manipulate music contents, especially in conducting, where the conductor directs a musical performance by way of visible gestures, but to acquire or practice the skill of conducting can be extremely difficult. This paper sought to achieve a system where the user himself can interact with the system by conducting hand gestures; the system will then treat those movements as musical interpretation that can modify the music. There are different ways on how to convey musical decisions to the musicians, however this study is focused on the hand movements and because the style of conducting differs from one conductor to another, the system itself will not be limited to generic movements. The challenge therefore is how to utilize Leap Motion, a hand and finger motion detector with the use of $P-Recognition, an efficient method in developing recognition systems, to achieve the system. This paper will provide valuable information regarding the use of a controller, Leap motion’s hand detector with $P-Recognition in developing a virtual orchestra where the user can be a conductor that can direct a musical performance by using hand gestures that can be interpreted into a musical pieces that can alter the music performed on real time.

General Terms: Hand Gesture Recognition, $P, Orchestra Conductor

Additional Key Words and Phrases: Accelerometer Gesture Recognition, Feature sets, Leap motion, Pattern Recognition

1. INTRODUCTION

1.1 Background

With the rapid growth of today’s computing in recording and playing media contents, most of the system are just for recording and playing while controlling multimedia contents are rarely seen on mainstream systems, especially conducting. Conducting is one of the traditional ways of expressing music at the same time controlling it. Most of the people are also fascinated on conducting music, but conducting requires no less than one man for it, requires musicians to play instruments for you, there are also people who wants to practice conducting with their own style but man power is also an issue. With the use of $P-Recognition algorithm the proponents can achieve a system that could recognize/analyze hand gestures and patterns, and interpret those data as musical interpretation.

1.2 Problem Statement

The Study sought to identify the hand gestures of a conductor, use a sensor that captures hand and finger movements and follows it’s every gesture and translate it into machine language where it can alter the music. Moreover, the study sought to answer the following questions:

1. Is leap a viable device as the sensor for detecting hand gestures for orchestra conductors?
2. Is $P recognition a good algorithm for pattern recognition?
3. What is the software and language to use to be able to alter MIDI tempo?

1.3 Objectives

The Study sought to identify the hand gestures of a conductor, use a sensor that captures hand and finger movements and record its gesture pattern. Specifically, the study intended to accomplish the following objective:

1. To be able to Identify what are the generic hand gestures that are applicable in the software.
2. To be able to change the tempo of MIDI Real-time.
3. To be able to recognize the time-signature the user is conducting.

1.4 Significance

Just as every other branch of music, conducting is a skill that needs to be carefully cultivated. Unfortunately, it is a subject in which few college music directors in Asia have had any formal training. More often than not, they have simply learned on the job, gaining experience by playing under other conductors. The best way for a conductor to improve is in front of a live ensemble. The unfortunate reality, however, is that this is not always possible. Needless to say, attempting to improve one's conducting skills in such circumstances can be extremely difficult.

With these reasons, a study regarding Hand Pattern Recognition for an Orchestra Conductor will contribute in helping aspiring conductors to improve their skills and to attract interested parties to be a conductor, and would open more possibilities in the research in Hand Pattern Recognition.

1.5 Scope and Limitations

The study covers recognizing of the conductors hand gestures and analyze its patterns to change the tempo of the MIDI. A free-hand tracking with 3D Hand-based model and skeletal model will be used in the implementation of the study.

It includes manual hand gestures alone. Thus, considering the usage of baton, facial, and body expressions as well as eye gaze is beyond the scope of the study. Any hand movements that are not recognized nor do not exist in the training menu will be disregarded and will not affect the output.

The study will leap motion device as the device tool for detecting and recognizing 3D-Hand, and UNITY 3D is also use as the IDE to build the system, all the things that are to be accomplished are limited to what this devices and software is able to achieve.

The study will also cover the use of the algorithm $P recognition for pattern recognition for the time signature and C# MIDI toolkit for tempo change and MIDI playback.

2. REVIEW OF RELATED LITERATURE

2.1 ,A Pilot Study of Expressive Gesture Used by Classical Orchestra Conductors

According to the study by Braem and Bram (2000) comparing to the relatively large number of different hand-shapes that are phonological components of sign languages, the number of hand-shapes regularly used by conductors are limited, in respect, conducting gestures are similar to gestures used to accompany speech. The limited sets of hand-shapes include some of those found in most sign languages in the world, and those are used by young deaf children learning sign languages.

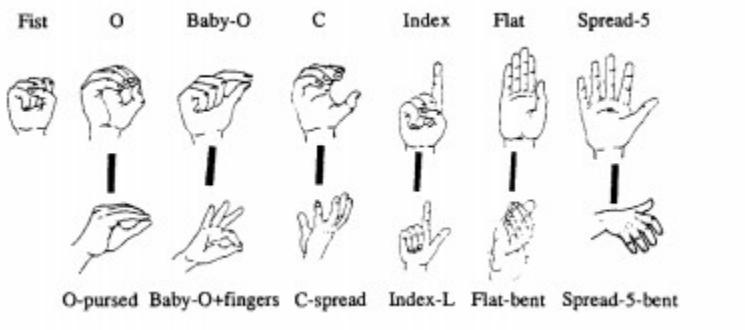
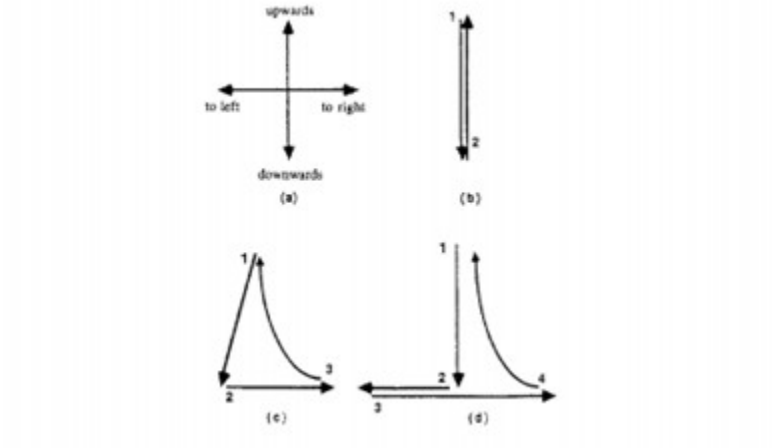


FIG. 1.0, The Limited set of Hand shapes used by the conductor in non-dominant hand-Gestures.

The dominant hand indicates the organization (the beginnings and ends), the tempo, and the rhythmic raster, or tact. The non-dominant hand shows special dynamics, sound colors, uniquely occurring events, entrances, and articulation. Naturally, all of these parameters influence each other and whether they are signaled by the dominant or the non-dominant hand is more of a general tendency than a firm rule. The dominant hand gestures are generally used to "direct the musical



traffic."

FIG. 1.1, Dominant Hand (from the conductor's perspective): (a) traditional division of the conducting space, showing the temporal organization of the music; (b) 2 beats; (c) 3 beats; (d) 4 beats.

2.1.1 A repertoire of non-dominant hand gestures

Manipulating Objects = Sound Quality, Structure, Articulation, Musical Development, Psychology Motivation.

"Taking out view" – Another common left-hand gesture used by many conductors is based on the metaphor of "taking something away from the visual field" (see Fig. 10.5). The gesture is used at the end of the piece or section to indicate "Stop the production of sound."

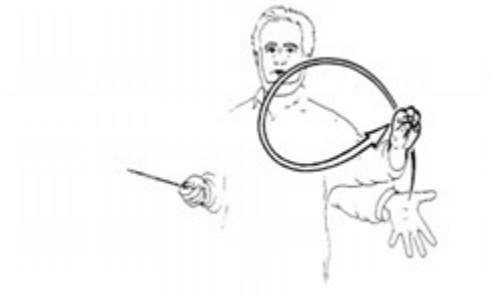
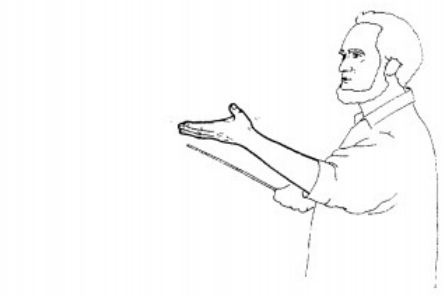


FIG 1.3 "Taking out view" = Stop Playing!



"Supporting an object" to sustain a solid sound quality (FIG. 1.4);

FIG. 1.5, “Touching a surface” = Sound quality (e.g., homogeneous sound quality).
"Touching a surface," this, depending on the type of movement and the hand shape can indicate, for example, a smooth, homogeneous sound with the full flat hand shape.

Showing the Path or Form of an Object = Structure

The manner of movement as the hand moves from group to group can be varied to indicate more details of the development: slow, brisk, abrupt change, and so forth (Fig. 1.6).

FIG. 1.6, “Path” = movement of musical material between instruments.

Vertical Direction = Dynamics

Vertical levels within the conducting space can indicate the dynamics of the music: high level = more = louder; low level = less = softer. These levels are indicated by a gesture with an open flat hand, moving upward or downward, palm held horizontally (FIG. 1.7a, and 1.7b).

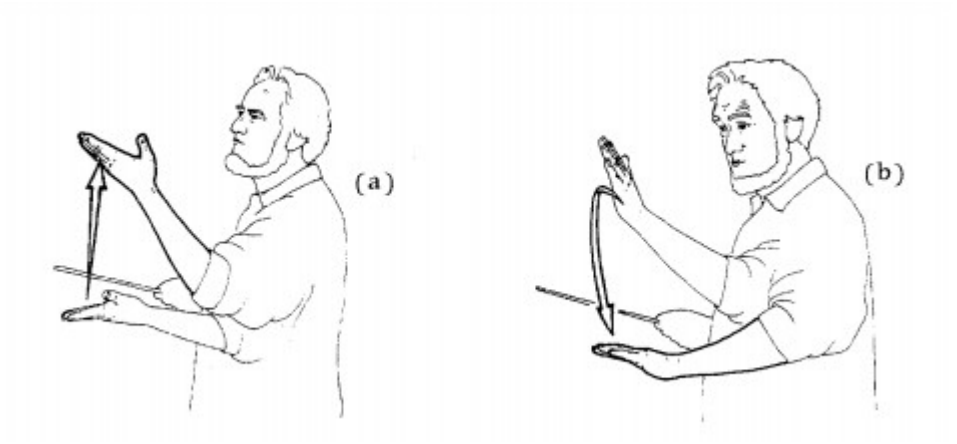


FIG. 1.7, (a) Upward = Louder; (b) Downward = softer

Portraying an Object = Sound Quality

A gesture in which a closed hand, palm oriented up, opens into a spread-5 hand shape is used for a particular timbre of the sound, a light, radiating quality (FIG. 1.8).

FIG. 1.8, "Rays" = Sound Quality (radiating, bright timbre).

"Holophrastic Interjections" = Tempo, Structure, Motivation

2.2 Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes

Rapid prototyping of gesture interaction for emerging touch platforms requires developers to have access to fast, simple and accurate gesture recognition approaches. $P considers gestures as

cloud of points. $P performs similar to $1 ($1, $N, $N-protractor, $family of recognizers) on uni-strokes and is superior to $N on multi-strokes, Specifically $P delivers > 99% accuracy in user-dependent testing with 5+ training samples per gesture type and stays above 99% for user-independent tests when using data from 10 participants. $P is an instance-based nearest-neighbor classifier with a Euclidean scoring function.

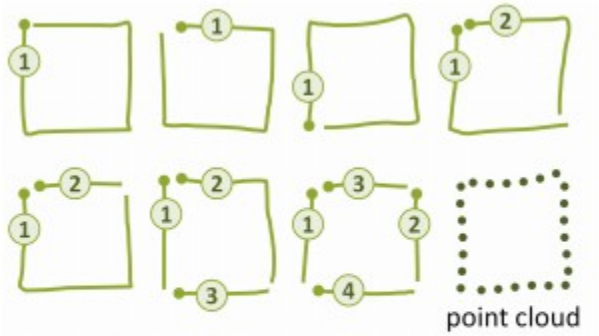


Figure 1: Even a simple square can be drawn using 1, 2, 3, or 4 strokes which can vary in order and direction (with a total of 442 possible cases). However, all the articulation details are ignored when looking at the square as a time-free cloud of points.

2.1.1 $P vs $N and Hungarian

First they conducted a recognition experiment in order to understand the performance of the Hungarian recognizer and to be able to compare it with its main competitor, $N.

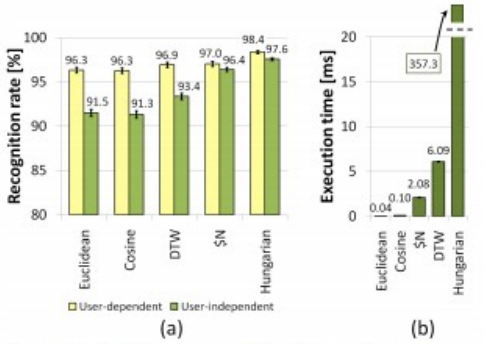


Figure 3: Performance of the Hungarian vs. competitor recognizers: (a) recognition rate; (b) execution time (for a training set of 16 gestures × 3 samples). Error bars show 95% CI.

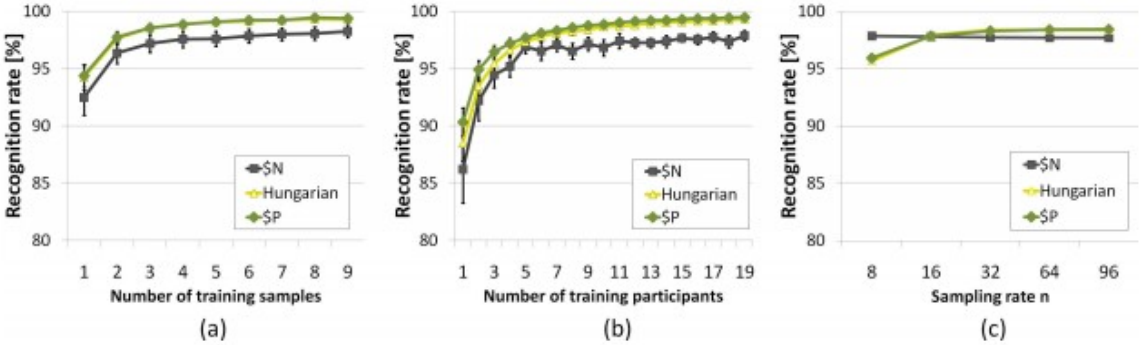


Figure 5: Recognition performance of $P vs. $N [2, 3] and Hungarian [15].

Results obtained $P was not significantly different from Hungarian for user dependent tests. However significant differences were detected between the two recognizers for user-independent testing, exhibiting better performances

2.3 Approach to Hand Tracking and Gesture Recognition Based on Depth-Sensing Cameras and EMG Monitoring

Gestures arise from person's mental concept and are expressed through motion of arms and hands. These expressions are observed and recognize by a spectator, in their case the spectator is a computational device that is able to recognize gestures through specific pre-learnt models.

There are two conventional approaches towards gestures recognition such as *Free-hand* tracking and *Glove based-tracking*. Both of the approaches have a need for specific mathematical model. This kind of tracking is not affected by surrounding and allows the usage of multiple sensors such as force sensors, accelerometer, gyroscope or bend sensors.

As for the *Free-hand tracking* Sharma & Huang (1997) further define two main models. First is 3D *Hand-based models* and *Skeletal Models*, 3D *Hand-based Models* are based on 3D description of the hand and skeletal model is reduced to set of equivalent joint angle.

Parameters with segment lengths, second is *Appearance-based model* are not directly like 3D that compromises the 3D description of the hand but instead it's based on the display of hands in sequence of images.(Ondrej Kainz1 , František Jakab1, 2014)

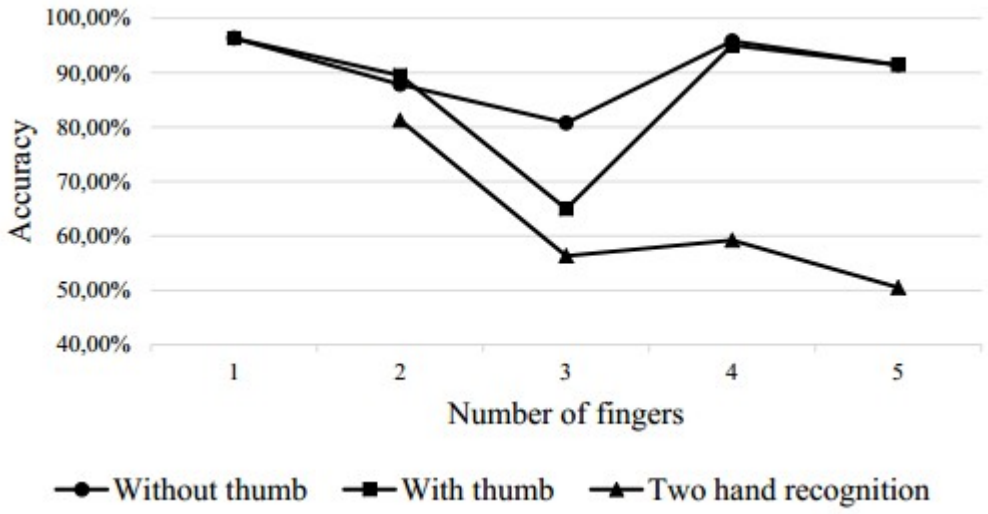
2.3.1 Hand Tracking Using Leap Motion

Mobility price and distance interaction range were the factors that lead them to choose leap motion as the device to be tested for hand tracking purposes.

	Microsoft Kinect	Asus Xtion PRO	Leap Motion
Distance (cm)	40 - 350	80 - 350	7 - 25
Field of view (horizontal, vertical, diagonal)	57°, 43°	58°, 45°, 70°	140°
Sensor	RGB, Depth, Microphone	RGB, Depth, Microphone	Depth
Depth image size	VGA (640x480)	VGA (640x480) : 30 fps, QVGA (320x240): 60 fps	VGA (640x480)
Resolution	SXGA− (1280 x 960)	SXGA (1280x1024)	N/A
Programming language support	C++, C#	C++, C#, Java	C++, C#, Java, JavaScript, Python
Dimensions (cm)	28 x 8 x 8	18 x 3.5 x 5	8 x 3 x 1

The table above shows the comparison of the depth sensing cameras

Their focus primarily on the accuracy of the leap motion device, First for one hand, next to both hands, Tests were conducted in a laboratory environment under regular lightning conditions, No smudges or any other obstacles that could disrupt the measurements were observed. Hands were placed in horizontal position with an approximate height of 16 cm from the device, with interaction height was set to just declare value. The goal of the testing was to estimate the accuracy of the device. Results were collected for over 1500 samples per measurement.



The figure above show the best accuracy achieve when one figure was used in tracking with an accuracy rate of 96.34%. Minimum accuracy was observed when other factors were included.

Compared to their study, their study needs EMG device to detect skeletal muscles to detect the electrical activity produced by the skeletal muscles on forearm. The purpose of such combination is to enhance the gesture recognition rate. And Leap motion as the depth sensing device. However leap motion alone has a great deal of accuracy rate in terms of depth sensing with a proper tuned ANN the leap motion alone is enough without the use of EMG to detect hand gesture and motion. (Vatavu, Anthony, Wobbrock, 2012)

2.4 GESTURE RECOGNITION LIBRARY FOR LEAP MOTION CONTROLLER

This thesis study by Nowicki, Pilarczyk, .et al (2014) studies the new possibilities to gesture interfaces that emerged with a leap motion sensor. The Leap Motion is an innovative, 3D motion capturing device designed especially for hands and fingers tracking with precision up to 0.01mm. The outcome of this thesis is the leapGesture library dedicated to the developers for leap motion controller that contains algorithms allowing learning and recognizing gestures.

2.4.1 Controller

Leap Motion is an usb sensor device by Leap motion inc. designed to provide real time tracking of hands and fingers in three-dimensional space with 0.01 millimeter accuracy. It allows user to obtain information about objects located in device's field of view (about 150 degree with distance not exceeding 1 meter).

FIGURE 3.1: Leap Motion controller with Micro–USB plug

2.4.2 Proposed Methods

The static gesture recognition problems can be stated as a problem invariant to the time. This means that for each detected hand, its position and orientation can be treated as a new data point, uncorrelated to previously shown gestures. With these assumptions, one can easily generate multiple samples from the sensors in short period of time, at the same time it also gives an opportunity to look at the static gesture recognition problem as a problem of classification.

In application of the library presented in this thesis, it is assumed that the learning process can be done offline, while strict online requirements has to be met for the recognition part. To meet listed requirements, the Support Vector Machines (SVMs) were used as a robust classification algorithm. The SVMs were chosen as there exist a solid mathematical background supporting the simple idea of maximizing the margin between classes. The SVMs are popular in the recent research trends and are commonly used in biology, robotics or IT for solving data classification problems. Another advantage is the existence of the open-source library libSVM, which contains an efficient implementation of the SVM in many programming languages.

2.4.3 Evaluation Methodology

With samples of static gestures without labels, For each sample the sets of features are computed and then given as input to the trained classifier. The classifier provides the information of the predicted gesture's class membership (label) for each input sample.

2.5 C# MIDI ToolKit

A MIDI toolkit by Leslie Sanford, is a .NET MIDI toolkit for creating MIDI applications, this toolkit takes a more traditional C#/.NET Approach to flow-based programming.

2.5.1 Flow-Base Programming

In computer programming flow based programming or FBP is a programming is a programming paradigm that uses a “data factory” a metaphor for designing and building applications. FBP defines applications as networks of “black box” processes in which it exchanges data across predefined connections by message passing. FBP is naturally component-oriented.

C# MIDI toolkit uses flow-based programming as an approach, it uses this “Black box” processes as methods on passing and exchanging predefined connections through passing of message. C# MIDI toolkit has MIDI messages, these are the messages that are being passed and exchange.

2.5.2 MIDI Time-Signature

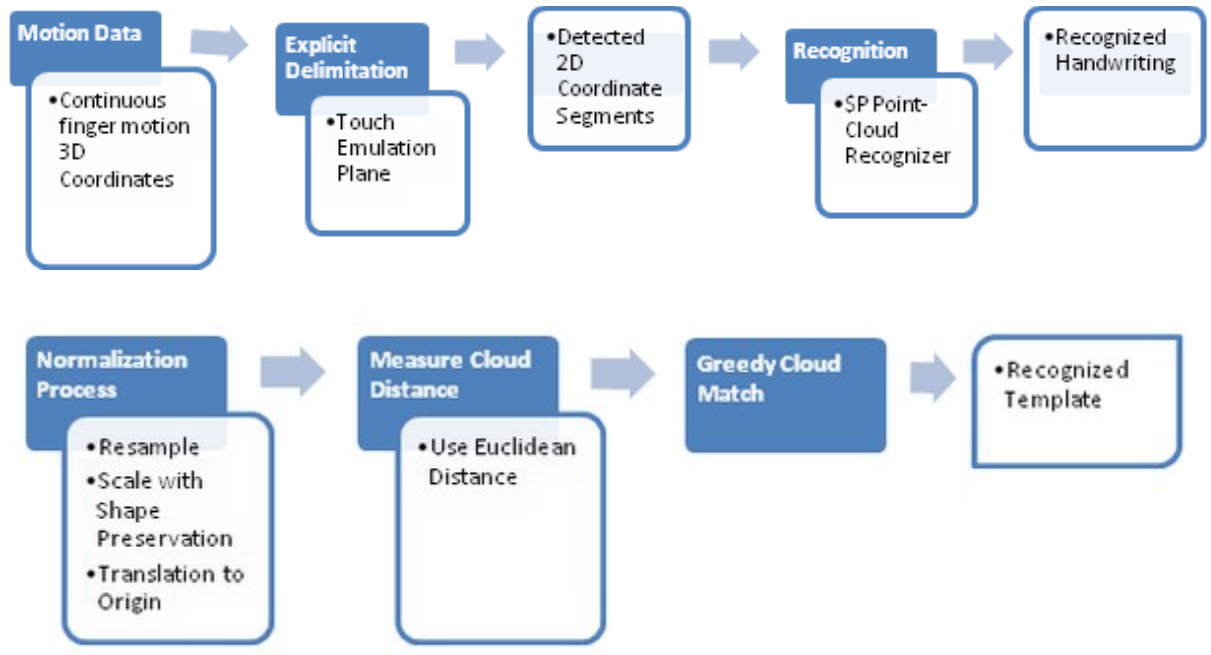
In conducting time signature is important, firstly it is vital to properly notate a piece of music. Reading MIDI will give you hex values that represent many events and messages in a MIDI such as obtaining the Time-Signature of a piece it is identified that the format of time signature in a MIDI file is “FF 58 04 nn dd cc bb” where nn is the numerator, dd is

2.6 Process Designing of Virtual Valipilia

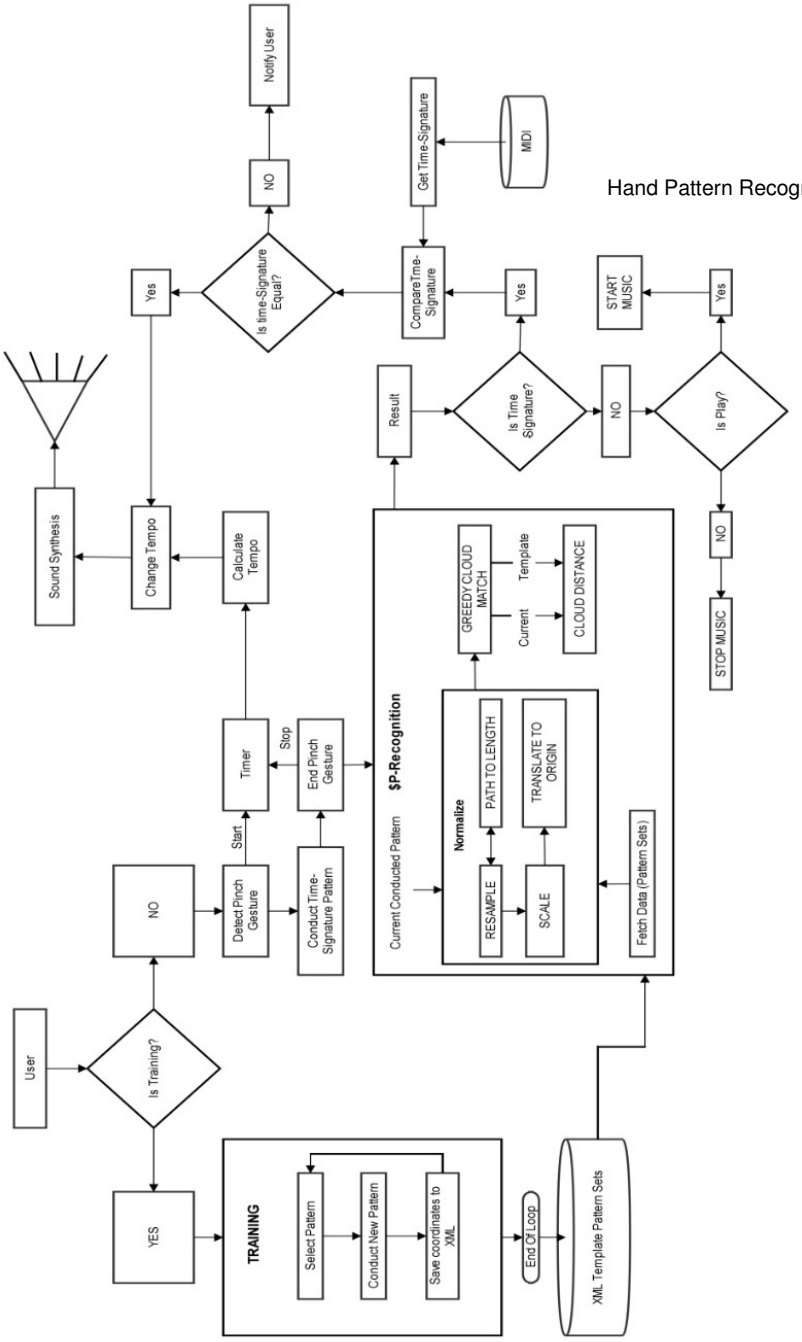
Air-writing recognition allows user to input text by simply writing in the air using your fingers. The air finger writing renders characters that are written with the tip of a finger or a stylus pen, on a virtual plane. It involves no physical plane to write on, and doesn’t have direct pen up/down information. In other words, air writing is a uni-stroke and different from the ordinary and traditional way of handwriting. This type of research deals with interpreting data’s while it is generated and is much more different from the process of OCR systems (Optical Character Recognition). So most of the researchers that carried out in this area is to find a better recognition techniques and algorithms to recognize uni-stroke writing in virtual plane. To put it simply it is like building gesture based writing tool.

In this study the researchers develop a system that will recognize text in Air-writing from a virtual 2D plane. The researchers chose $P as the algorithm for recognition phase.

1 Theoretical Frameworks



(Framework from Process Designing of Virtual Valipilia)



3. RESEARCH DESIGN AND METHODOLOGY

3.1 Conceptual Framework

3.1.1 Data Gathering and Preparation

The first phase is for the user to select either to train for a gesture or start conducting.

3.1.2a Training

The proposed training will just let the user create training samples for a pattern and save it to an XML file associated with the filename base on what is the pattern name, that to be use by the algorithm later on.

3.1.2b Start Conducting

This is when the user select to play instead of training, In this phase the system will wait for the user to perform a certain gesture before it records the pattern done by the user, when the user starts conducting a timer will start and will time how long the user creates the pattern.

3.1.3 $P Recognition Algorithm

After the user finish conducting, the coordinate data points and training set samples with the user conducts will be fed in the algorithm and will be process through series of fixations and processes in proper order; Normalization that contains Resampling, Path to length, Scale and translate to Origin after that data will be pass on Greedy Cloud matching

where it calls the cloud matching where it matches data until it find any data's that has the closest distance towards each points.

3.1.5 Results

After the algorithm identified a pattern the system will check on what to do on that pattern, there are classification of certain pattern either it is a time signature pattern or not, if it is not a time signature pattern the system will consider that as a START, STOP, RESUME patterns that is responsible for playing, stopping and resuming the music.

3.1.7 Time Signature Pattern

If the system identified it as a time signature pattern then it will check if the pattern matches the time signature of the MIDI file, if it doesn't match it will tell the user to conduct the proper pattern and ignore it, and if the user conduct correct pattern it will then go to the next phase changing tempo.

3.1.8 Tempo Change

After the system identified the pattern as the correct time signature, the system will now calculate the tempo in terms to the time, the timer that is used a while ago will be used to compute the bpm then will change the tempo of the MIDI.

3.2 Methodology

3.2.1 3D hand Detection

The proponents will use leap motion device to gather hand data's, data's x and y coordinates data, z data's aren't needed so the proponents neglect the usage of z.

3.2.2 Gesture Recognition

In order for the system to know when and where to start getting the data's from the hand, It is needed to detect a certain gesture to start recording data's from the hand, thus the proponents Identified two key pinch gesture that the conductor must do.

The two pinch gesture that are identified are thumb distal phalange, index metacarpal and thumb distal phalange and index point when the users thumb distal phalange and index metacarpal touches or thumb distal phalange and index point touches it signifies start recording, in recording the system will gather coordinates data per frame.

In order to identify that those points touches each other then use this formula

iM or iC = The position of the Index Point or Index metacarpal; x,y,z
tP = the position of the thumb Point; x,y,z
Distance = the distance from two points.
normalizeDistance = the normalize distance in terms of pinch min and max distance.

m_pinchMinDistance = A threshold for the minimum distance
m_pinchMaxDistance = A threshold for the maximum distance

Thumb distal phalange and Index metacarpal: *Distance = magnitude of (iM – tP);*

Thumb Point and Index Point: *Distance = magnitude of (iC – tP);*

Magnitude is computed as the |distance| = sqrt of (iC – tP) or sqrt of (iM – tP)

normalizeDistance = (distance – m_pinchMinDistance) / (m_pinchMaxDistance – m_pinchMin Distance)

Subtract the minimum pinch distance to the distance and divide it by the difference between the max distance and minimum distance thus, get the normalized distance.

The numbers that are obtaining when doing a pinch is 0 or less and when not pinching the values are above 0 and 1. So in order to identify, it is need to limit those numbers into numbers ranging 0 to 1. So using Mathf function “CLAMP”, what clamp do is that when a number goes beyond or lower to the specified lowest and highest value, it automatically clamps the value into the highest or lowest value specified.

normalizeDistance = Mathf.Clamp(normalizeDistance, 0.0f, 1.0f)

What this do is, when pinching the value of the normalize distance will be 0 and if not is above 0 and does not exceed 1, but it is really awkward especially when doing an if else that choses 0 as true and 1 as false, thus the normalize distance is subtracted to 1, so that we get 1 as the value when pinching and when not its value is less than 1 but not lesser than 0.

normalizeDistance = 1 – normalizeDistance

3.2.3 $P Recognition for pattern Recognition

$P recognition is an instance base nearest neighbor algorithm. What it does is that it matches the current pattern to all of the training set available. The algorithm converts all coordinate points into point clouds.

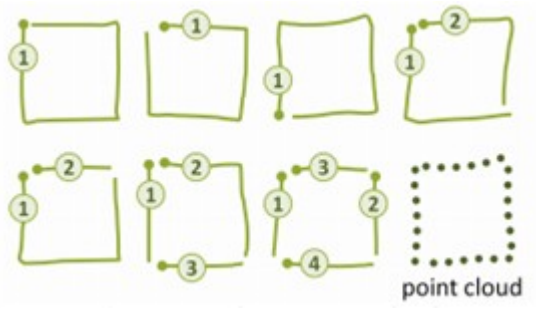


FIG 3.0: The square pattern and how it looks in point cloud.

In FIG 3.0 it could be seen in 7 ways to create a square and the variation of how can a square be drawn using 1, 2, 3, or 4 strokes which can also vary in order. However the algorithm when seeing the square as point clouds it ignores the articulation details, the algorithm doesn't care whether the square is drawn from top to bottom, left to right or of how many strokes the user drew the square.

$P seeing patterns as point cloud recognize pattern by aligning each points to all of the points of the training sets, achieving that point cloud is considered as List of points that has values x and y, and the list is pass through out two classes such as Normalize and Greedy Cloud Match.

3.2.3.1 Normalize

Before comparing the points, it is needed to normalize those points and it is solve by this series of methods; Resample, scale, and Translate-To-Origin, point cloud as a list is also considered as a point's path or an array of points.

3.2.3.1.3 Resample

This part of the process of normalizing those points is Resample, in this phase the algorithm resamples a points path into *n* evenly spaced points, using n = 32, it means that all the points in a point cloud or points path are narrowed down/up into 32 number of evenly spaced points. Also in order to make those points evenly matched and call the method Path-Length.

3.2.3.1.3.1 Path-Length

Path-Length gets the Euclidean distance of two consecutive points.

Distance += EuclideanDistance(Pi – 1, Pi) // i is the array index of the point.

Distance = Distance / n // n = 32, Here it makes the interval spaces for 32 points

A = Pi – 1; B = Pi;

EuclideanDistance = sqrt of (a.X – b.X) * (a.X – b.X) + (a.Y – b.Y) * (a.Y – b.Y);

The resulting distance will be the interval spaces between points

After getting the distance from the method path length, the algorithm will check each distance consecutive points if those points are greater than or equal to “distance”, if it is greater than then the algorithm will append the point to the new list of points. If the distance is less than “distance” then the third distance will add be added till then that the distance of consecutive points will be greater than the “distance” that is taken from PathLength. This part of the algorithm will just create a new list of points with the length of n = 32 with equal distance.

3.2.3.1.2 Scale

In this phase of Normalization the algorithm scale the list of points with self-preservation into a [0..1] x [0..1] bounding box. To do that first take the maximum and minimum X and Y distance of the pattern.

First, creating a bounding box with the value a constant float with the largest and smallest single value if the point(x,y) is larger than the largest value then point(x,y) will be the largest value and if it’s smaller than the smallest value then it will be the smallest value.

Loop all points:

If Min.X > points[i].X then Min.X = points[i].X;
If Min.Y > points[i].Y then Min.Y = points[i].Y;
If Max.X < points[i].X then Max.X = points[i].X;
If Max.Y < points[i].Y then Min.Y = points[i].Y;

After getting the proper bounding box in terms to the Max and Min X and Y, A Scale is then created. Scale = MAX (max.X – min.X, max.Y – max.Y). This scale variable is to be divided later on in order for us to scale the points in a square bounding box, and then the algorithm will add points to the new bounding box.

Loop all points:

newPoints[i] = (points[i].X – minx) / scale, (points[i].Y – minY) / scale)

3.2.3.1.3 Translate to Origin

After scaling the points that is set on to the new bounding box is not centered it would be hard to compare distance later on if it is not centered so this method will center the point cloud into the origin. X = 0, y = 0. The bounding box is [0..1]x[0..1] so what will happened is that the half of the point cloud is inside the bounding box while the half is outside, it can be achieve by using this formula.

First take the centroid of the point cloud, the centroid is the center of a two dimensional region (x and y).

Loop points

Cx += points[i].X
Cy += points [i].Y

After, divide both Cx and Cy by the total length of the points.

Cx = Cx / points.length()
Cy = Cy / points.length()

After getting the centroid, then continue on getting the point cloud to the origin.

newPoint = (points[i].X – Cx, points[i].Y – Cy)

subtract the centroid to the current points so that the resulting coordinates of the points will be move down to the center origin.

Now normalization is done, Note the process of normalization does not limit to the pattern that has been done by the user, training sets will also undergo normalization, so that all of the pattern; current or training samples will all have the same number of points with even spaces and is located in the center origin so that it would be easier and viable to be compared with the next phase of the algorithm.

3.2.3.2 Greedy Cloud Match

This part of $P recognition is called Greedy Cloud Match from the name itself Greedy it matches the current pattern by the user to *all* pattern in the training set it's not just but during the process it gets the minimum cost of alignment for each point, it manually calculates all thus it is greedy.

The Greedy Cloud Match accepts two parameters first is the current pattern done by the user and the next is a pattern in training set.

Loop in terms of the length of the index of the points in this case all of the points now has the same number of length.

Inside the loop the algorithm will call two cloud distance function, the first cloud distance function will match the current pattern by the user to a training pattern, and the second cloud distance function will match the training pattern to the current pattern, and then get the minimum distance on both.

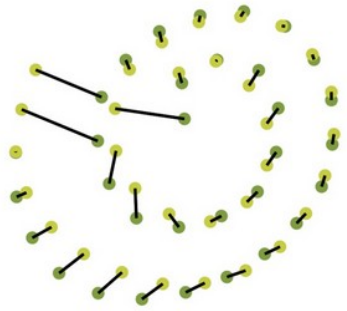


FIG 3.2.3.2.1, Visualization of how Greedy Cloud Match, aligns and math points

3.2.3.2.1 Cloud Distance

In Cloud Distance it computes the minimum weight of the current pattern and the training pattern from point to point starting from the first index of each point in point cloud.

3.2.4 Integrate Pattern Recognition and Gesture Recognition

In this phase in the methodology this integrate the gesture recognition and $P recognition. Each time a Gesture is done (see 3.2.2 for the two key gestures) the system records every frame coordinates of the hand where it moves until the pinching is done. After recording the coordinates of the points, the system then passes those points to the Pattern Recognition Algorithm (see 3.2.3 for the flow of the algorithm).

3.2.5 Load / Save XML Pattern Data's

This phase in the methodology is the training and fetching of XML pattern Data's, The data that is being fetch and is fed to the algorithm are XML Data that contains the coordinates of each points. When training user creates or improve a pattern, user will literally draw pattern on the screen and save it as XML. Also the proponent decided in the training part that the training will use mouse instead of your hand.

The reasons why instead of hand, mouse pointer is selected as the mode of drawing pattern is that, using hand gives stress to the user to add up the device leap motion has limited area and angle range where in fact gives stress to the hand of the user, also most of the time hand detection fluctuates does data being gathered using the hand are not good data's.

3.2.6 Audio Control: Load Midi

In this phase of the methodology user can load Midi files into the system, the proponents use C# midi toolkit in order to achieve this. The toolkit uses a sequencer class and sequence class. The Sequence Class represents the collections of tracks that also provide functionality for loading and saving MIDI files. The Sequencer class in this sense is a lightweight class for playing back Sequences.

3.2.7 Audio Control: Change Tempo in Real-Time

In this phase of the methodology after loading the file using the C# midi toolkit you can now have the option of using its classes, and one of those classes is the tempoChangeBuilder this class allows you to be able to create a meta message that has a type of tempoChangeBuilder, this message will be added to a track and send to the sequence class in order for the MIDI to change its tempo.

How to compute the tempo? The tempoChangeBuilder class accepts a parameter of microseconds and has a default tempo of 120bpm or 500000 microseconds. In order for us to compute the the tempo in terms of bpm to microseconds with this formula

Bpm = Beats per minute
Bps = Beats per second

Bps = Bpm / 60; //to get the beats per second, divide the bpm by 60, 1 minute = 60s

Sec = 1 / bps; //to get the second from every beat.

Millisecond = sec * 1000; // to get the millisecond. 1000 millisecond = 1 sec.
Microsecond = Millisecond * 1000 // to get the Microsecond.
1000 microsecond = 1 millisecond.

After getting the Microsecond it is then fed to the tempoChangeBuilder, and then afterwards build. tempoChangeBuilder has a method build that builds a meta message after building the meta message, the method result is then called this returns the meta message the has been build. It is then added to a track class to the sequence and then it changes the tempo, also note the limitation of the changes in tempo (see 1.5 for limitations).

3.2.8 Integrate Pattern Recognition and Audio Control

The last part of the methodology, the proponents integrate the pattern recognition and audio control: Change Tempo in Real-time. As discussed in 3.2.7 in order to compute the tempo the parameter that is required is microsecond. So the proponents added a timer class during the conducting. By the time a Pinch Gesture is done, a timer starts and ends until the pinch gesture is finished, then the system checks whether the time signature of the MIDI is the same as the conducting pattern that is made by the user. If it is the same the system can calculate the tempo in terms with time and time-signature, with the help of the panelist Fr. Carlos G Cenzon, formulated a formula to get the tempo relative to time and time-signature;

Time = total time spent from conducting a time-signature pattern.
Stroke = total number of stroke of a time signature.

bpm = (stroke / time) * 60; // it is multiplied by 60 because time is interms of seconds
So to get bpm, multiply it by 60; 1 minute = 60seconds.

Every MIDI file has time-signature data's or if it doesn't have it is automatically set to 4/4, the toolkit doesn't have any way of recognizing the time-signature of a certain MIDI file so in order to solve that problem after loading the file, the system will also read the MIDI as an array of bytes, the system will then loop the array in terms of its length. In those hundreds of bytes the system can find the time signature of the MIDI piece, a time signature of a certain midi file has a format;

FF 58 04 nn dd cc bb
FF 58 04 03 02 // this is an example of a 3/4 time signature

In the example above it is a 3/4 time signature but dd = 2. Note that dd in this sense is a power. In a formula;

Time signature denominator is = 2 ^ dd
FF 58 04 03 02 // 3 / (2 ^2) = 3 / 4

4. RESULTS AND DISCUSSION

4.1 Results and Evaluation

4.1.2 Audio Control Pattern Testing

Start		
n=40	Predicted: **NO**	Predicted: **YES**
Actual: **NO**	20	0
Actual: **YES**	0	20
	Accuracy:	1
	Precision:	1

Pause		
n=40	Predicted: **NO**	Predicted: **YES**
Actual: **NO**	20	0
Actual: **YES**	0	20
	Accuracy:	1
	Precision:	1

Resume		
n=40	Predicted: **NO**	Pre
Actual: **NO**	20	
Actual: **YES**	0	
	Accuracy:	
	Precision:	

4.1.2 Time Signature Pattern Testing

2/4 Time Signature		
n=40	Predicted: **NO**	Predicted: **YES**
Actual: **NO**	18	2
Actual: **YES**	1	19
	Accuracy:	0.925
	Precision:	0.95

3/4 Time Signature		
n=40	Predicted: **NO**	Predicted: **YES**
Actual: **NO**	18	2
Actual: **YES**	2	18
	Accuracy:	0.9
	Precision:	0.9

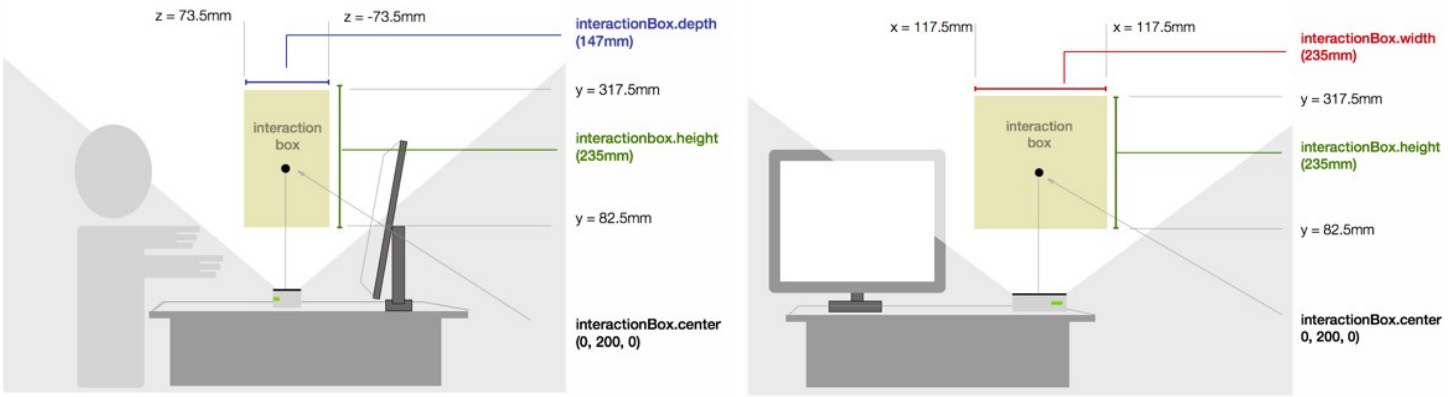
4/4 Time Signature		
n=40	Predicted: **NO**	Pre
Actual: **NO**	19	
Actual: **YES**	1	
	Accuracy:	
	Precision:	

Each of the patterns has 20 training samples, using the confusion matrix to get the accuracy and precision of the algorithm averaging of Accuracy: 0.9625(96.25%), precision: 0.9666(97%) for the time-signature and control pattern.

4.2 Discussion

4.2.1 Limitations

The device that has been use for detecting hand model and movement LEAP has an area range limitation.



(Coordinate Systems — Leap Motion C# SDK v2.3 documentation, https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Coordinate_Mapping.html)

The algorithm $P recognition recognize pattern who has the highest score from all the training sets (scores from 0 to 1). The algorithm will not be able to detect unknown pattern because it will scan the entire pattern in the training set and will identify pattern even though it has low scores.

As the proponents do the study, it have been clarified that the algorithm is a good approach for pattern recognition but is lacking at some point to be of use for pattern recognition for orchestra conductors, the algorithm will recognize as soon as the pattern is done in other words it will only start recognizing after the user is finish doing the pattern the orchestra in this sense do strokes and implies beat so whether the entire pattern is finish or not it should already affect the music.

The use of artificial neural networks is a good choice for this kind of problem the $P is facing and the first approach of this study includes the use of neural networks but then it is later on realize that $P can recognize pattern alone, but using artificial neural networks has its own perks, but the problem arises when the chosen platform UNITY 3D don't have a way an option on

creating a network, the proponents found an asset for UNITY for creating networks for artificial networks but it's not free it roughly cost $90, the other choice is using ENCOG its language is C# and UNITY is C# too, but the problem is the .NET framework used by ENCOG and UNITY is different. ENCOG in this sense uses native .NET DLL frameworks while UNITY uses the mono version of this DLL's although there are version of this DLL's in mono, the support on this DLL is not enough and most of the unsupported DLL's is needed for ENCOG.

The MIDI toolkit that has been used by the proponents to read and create MIDI events has a limitation on changing the tempo, when passing a meta message that signifies a change of tempo the proponents need to use the continue command in which the next note will have the new tempo but the problem is that it skips to the next note, when playing a long note and invoke a continue command it will not finish the playing of the note instead it jumps to the next note.

The study also covers only quarter notes, it means that MIDI's with time-signature that has a denominator less or more than 4 is not part of the study.

The panels also suggested that the suggested not to include the change in volume, although the proponents tried to do a thing on that approach a problem arises regarding the toolkit and the IDE (unity) they used. Unity can change its volume if the audio is from an audio source and the audio source don't accept MIDI, although this problem is solve using the toolkit as a host for playing the audio then the problem on changing its volume came up the toolkit doesn't have any way of changing the volume of MIDI so the only solution left is using the unity to change the System volume, but then unity cannot change volume aside from its audio source.

5. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The system that has been built for this study has a good accuracy rate for recognizing time-signature and gestures, the system could also alter the tempo real-time via the time spent conducting a time signature, the system could also read the time-signature of a MIDI file through the method of opening the MIDI as an array of byte and check the hex code FF 58 04.

In this study, using $P recognition to recognize pattern with proper training set is achieved, the only problem that has come up is that it isn't enough for an orchestra, $P again (found in 4.2.2 limitation) has limitations that are not suited for orchestra. Strokes made by the conductor has musical interpretation, each strokes signifies beats and the conductor also needs to conduct the proper time-signature or else there is a note that will be lost, the study achieve on recognizing the proper time-signature, the problem is that the algorithm cannot be able to determine strokes and require the pattern to be finish throughout so that it can recognize, so the strokes are ignored. Even so the algorithm $P have above 95% up to 99% accuracy rate in recognizing pattern without any problems with the hand detection from the leap motion.

5.1.1 Contribution to $P recognition

In this study $P recognition has done its job in recognizing pattern, the only thing that is missing in this algorithm is that it doesn't recognize unknown gesture instead it recognize patterns who has the nearest distance towards each of its points, however this study is a good eye-opener for students and researcher who wants to try and experiment $P recognition for Pattern recognition, also like any other pattern recognition algorithm each of it has their own strengths and weaknesses and in this study the proponents found out the weakness of $P recognition in the study of Orchestra Conductor .

5.1.2 Contribution to Leap Motion

In this study the proponents use Leap Motion Device as a device for the study, as the proponents search and experiment with the device, the device is good for building blocks applications, applications that have to do with the hands. As for the study Leap Motion prove to have a say for it, it's just that a proper algorithm is in need for it to be utilize properly, as for the study orchestra Leap motion controller has so many limitations that is required for orchestra conductors, Conductors require a large area of space in conducting while leap has a limited, conducting requires not just the hands but also other body parts. As said earlier leap is good for building block applications it is good if it is used for building applications that have more object interaction rather than data presentations, like any other device, devices have their own strength and weaknesses and in this study the proponents found the weakness of Leap Motion in the study of Orchestra Conductor.

5.1.3 Contribution to MIDI study

In this study the proponents used C# midi toolkit from manipulating MIDI files, this toolkit is existed for some time but still much more to be explore in this area, perhaps a proper documentation on how to use the toolkit will help students and researcher to use the toolkit proper and efficiently, the toolkit is good when writing MIDI applications the only thing that is missing is that It doesn't have a smooth and a proper way of changing tempo, the toolkit is designed for input devices such as Piano, keyboard to be able to play musicality in it but for the study of orchestra conductor it has proven that it lacks quality over its design. Also the toolkit doesn't have in it a way of recognizing the time signature of a certain midi but it has timeSignatureBuilder to build time signature but for a MIDI that has already have a time-signature in it, a method is required to be able to recognize the time-signature or to simply open the midi file as an array of byte and loop each byte and find the time signature, like any other way, C# midi toolkit is not the only MIDI toolkit that can be used for MIDI applications like any other applications they have their own advantages and disadvantages and in this study C# midi toolkit prove to be lacking in some area.

5.1.1 Contribution to the study for pattern recognition for orchestra conductor

In this study, the proponents use a pattern recognition algorithm, device, a toolkit used in the study, the study prove that these methods prove to have their own achievements, problems and limitations, like any other methods on answering this problems it is concluded that this approach

lacks some aspects in itself thus a further study and other methods or methods combine with these methods are the light for the continuing this study..

5.2 Recommendations

First and foremost, the proponents recommended to future Computer Science student researchers to consider this study as a study that is open for improvements.

The study has achieved a good performance in terms of pattern recognition and Real-time Tempo change, but the study for orchestra requires more than pattern, stroke detection with musical interpretation is recommended for further improvements.

Second, usage of another algorithm aside from $P recognition, before the study also have neural networks as a part of it, due to some circumstances that neural networks is not available because the neural networks framework is not FREE in UNITY, it is a recommendation for the future student researchers to use another methods and algorithm to achieve more on this study

Third, the study proved that using Leap motion for the study for conducting is not enough as a device tool for the study of orchestra, It is recommended that another device or tool is to be used future study and kinect device is recommended; unlike leap motion that only detects the hand, kinect detects the entire body of the user thus the limited area range of Leap motion can be solve with its wide range coverage. But if the researcher still want to use Leap motion as the device tool for this study it is recommended to level up the approach with the usage of Oculust Rift for Virtual Reality, although Leap motion has a limited area range coverage, it can be overcome if it is embedded onto an Oculust Rift in that case the area will be widen by the movement of the Oculust Rift.

Lastly, the proponents also recommend to future researchers to consider the musical interpretation per stroke of the conductor, instead of just sticking with a MIDI playback and like this study alter the tempo in terms of the time when a time signature is conducted, it is highly recommended to include in the future study to play notes per stroke of the conductor, Volume Control and to be more promising to convey emotional and other musical interpretation in the future study.

REFERENCES

Radu-Daniel Vatavu, Lisa Anthony, Jacob O. Wobbrock. Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes, ICMI'12, October 22–26, 2012, Santa Monica, California, USA.

Michal Nowicki, Olgierd Pilarczyk, , Jakub Wąsikowski, Katarzyna Zjawin. GESTURE RECOGNITION LIBRARY FOR LEAP MOTION CONTROLLER. Poznan University of Technology Faculty of Computing Institute of Computing Science, 2014

Emmorey, Karen & Lane, Harlan. (eds.) . .2000. The Signs o~ Language Revisited. Mahwah, New Jersey: Lawrenc-e Erlbaum Associates.

Ondrej Kainz, František Jakab. Approach to Hand Tracking and Gesture Recognition Based on Depth-Sensing Cameras and EMG Monitoring. Computer Networks Laboratory at DCI FEEI TU. 2014

Leslie Sanford, C# MIDI Toolkit, CodeProject, April 2007, http://www.codeproject.com/Articles/6228/C-MIDI-Toolkit.

Jpaul Morrison, Flow-based-programming, http://www.jpaulmorrison.com/fbp/

Ben Smiley Andrew, MIDI time signature and BPM, deluge.co

Coordinate Systems — Leap Motion C# SDK v2.3 documentation,
https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Coordinate_Mapping.html

Upekha Vandebona, July 28, http://upekhavandebona.blogspot.com/2015/07/process-designing-of-virtual-valipilla.html,