DJ Spatial Tracking and Gesture Recognition for Audio Effects and Mixing

Andrew Hamblin, Evan Leong, and Theo Wiersema
Adviser: Dr. José Sánchez

BRADLEY University

Department of Electrical and Computer Engineering
May 3, 2016

Abstract

In the music production world, there is a disconnect between a DJ and their software. A team was formed with the goal of bridging this gap through the creation of a DJ glove system that increases gesture recognition reliability in realistic settings. The system must use tri-color LEDs, a camera, and a glove that weighs less than 1 lb. The system must also operate in real time. The glove will have controllable modes, identified by the color of the LEDs. Based on the glove's mode and the gesture performed, the system will trigger an audio effect through DJ software. The DJ Glove should be easy to use and portable. The Trinket microcontroller will be used to control the LEDs as it provides the necessary number of PWM channels and is small enough to mount on a glove. The recognition system will acquire gesture information from the Pixy camera and process this information using the HMM running on a Raspberry Pi. This device will send the pre-mapped command to the open source DJ software, Mixxx, to trigger the audio effect.

TABLE OF CONTENTS

I. INTRODUCTION AND OVERVIEW

A. Problem Background

The following information has been drawn from [1]. Hand-based gesture recognition for computer control originated from the use of sign language for human communication. Early systems used a glove to track the position of the hand. The methods for tracking have varied widely over the years. The first glove systems were developed using technologies such as accelerometers and fiber-optic bend sensors. These could be programmed to record the motion of the hand. Because of limited processing power, these solutions were effective. Instead of having to perform intensive image processing, the motions could simply be mapped to gestures. The Sayre Glove [1] was one of the first gloves to perform gesture recognition. Light sensors were placed on the fingertips and were connected to light sources through a tube that ran down the finger. The user would control the amount of light hitting the sensors by bending their fingers. While the number of possible commands was limited, it was lightweight and inexpensive. Another early system, the Z Glove [1], used multiple sensors including touch, inertial, and tilt. This combination of sensors gave it the ability to recognize up to 80 unique gestures. However, the glove was tethered to the computing system, limiting mobility and was never released to the public. These required the sensor system to be attached to the glove. Newer systems have been developed that include the use of multiple sensors on the hand and wireless communication with the computing system. With the increase of processing power, however, some newer systems have been designed without sensors in the glove at all. Instead, the glove is equipped with lights or multicolored fingers. These offer the benefit of being wireless and remove the processing from the body to a stationary device. In the 1980's, the Massachusetts Institute of Technology (MIT) developed the MIT LED glove [1]. This included an image recognition system that could track LEDs placed on the fingertips of a glove. This however faced many issues due to overlap of the LEDs and the variability of gestures. Ten years later, work was begun on a glove system that used colored fingertips to identify gestures. This was the first successful image recognition system in use for gesture control and showed that it was potentially a better solution than wired sensors on a glove.

In the past 10 years, there have been several gesture recognition techniques. One of the most used techniques is the hidden Markov model (HMM) [2]. In this model, the system is assumed to be a Markov process, which means that the system can predict future performance based on its present state and history. This method was used by Hyeon-Kyu Lee and Jin H. Kim in their attempt to use gesture recognition to control a PowerPoint presentation [3]. Another commonly used method is dynamic time warping (DTW) which compares the similarity of two signals and has the advantage of dealing with differences in time or speed. G.A. ten Holt, M.J.T. Reinders and E.A. Hendriks worked on developing DTW for use in gesture recognition [4].

Currently, GECO has designed a gesture control system that works in conjunction with Leap Motion. GECO's system allows the user to manipulate music by tracking the position of their hands. No glove or sensors are needed on the hand, as the Leap Motion controller is able to accurately track the position of two hands in 3D space [5].

B. Problem Statement

The goal of the DJ glove is to provide an interactive system that allows for a seamless communication of expression between the performer and the DJ software. Non-ideal lighting environments hinder the ability of a device to recognize gestural input. The DJ glove bridges the gap between performance and technology by increasing gesture recognition reliability in realistic settings. A glove equipped with tri-color LEDs will be tracked by a single camera interfaced with a processing system. Operating in various lighting conditions, the system will acquire and recognize a user's gestures. These gestures, in combination with the LED colors, will indicate and trigger a specific audio effect in real-time.

C. Constraints

Table 1 lists the constraints for the DJ Glove. These are requirements that the DJ Glove must meet.

TABLE I. CONSTRAINTS

Must have tri-color LEDs
Must have real-time execution
Must have maximum weight of glove less than 0.45 kg (1 lb.)
Must use one camera

Note that real-time execution has been defined as 160 milliseconds: the response time of the human ear.

D. Scope

The scope for the DJ Glove is shown in Table 2. This is defined in order to establish boundaries.

TABLE II. SCOPE

Out of Scope	**In Scope**
3 Dimensional Image Acquisition	2 Dimensional Image Acquisition
User Gesture Training	Predefined Gestures

II. Statement of Work

E. Nonfunctional Requirements

The nonfunctional requirements are listed in Table 3. These requirements are included to minimize complexity and inconvenience for the DJ. The glove and gesture recognition system should be lightweight and small in form factor to ensure portability. The DJ should be able to quickly learn how to use the gesture and color combinations to carry out an audio effect. See Appendix C for metrics and comparison of nonfunctional requirements.

TABLE III. NONFUNCTIONAL REQUIREMENTS

The system must be easy to use
The system must be portable

F. Functional Requirements

The functions stated in Table 4 describe the operational requirements for the DJ glove. The column to the right of each function contains the unique specifications that determine how well the system will perform. These specifications are required for the DJ glove to be considered fully functional.

TABLE IV. FUNCTIONAL REQUIREMENTS

Subsystem	**Function**	**Specification**
Glove	The system shall display predefined color schemes through tri-color LEDs.	Embedded device will send the appropriate signals to LEDs in order to display selected color scheme.
Pixy Camera [6]	The system shall acquire DJ's gestures.	Acquires gesture at 10 frames/second at a 400 x 240 pixel resolution for an 85% success rate under ambient light up to 250 cd [7] [8] [9].
HMM	The system shall recognize DJ's gestures.	Maintains a 75% success rate while staying below 160 ms of latency [10].
Mixxx Software [11]	The system shall trigger sound effects specified by gesture and LED color combination.	The Raspberry Pi [12] will send a command to Mixxx that is dictated by the gesture and LED color combination.

G. Design Overview

1) System Block Diagram

Figure 1 is the system block diagram, which shows the inputs and output of the system. The first input is the gesture that the DJ will be performing. The second input is the mode select. This will be a button on the glove that allows the DJ to select which color the LEDs on the glove will emit. This color will be detected by the system and used to determine the output, which is the audio effect, executed through the DJ software.

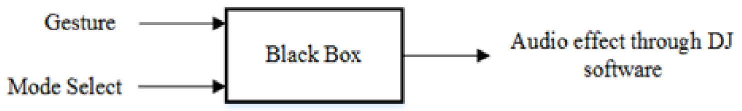


Fig. 1. System block diagram

2) *Subsystem Block Diagram*

Figure 2 is the subsystem block diagram, which includes the inputs and output of the system as well as those of the subsystems. The gesture performed by the DJ will be acquired by the camera system. The mode select will be read by an embedded device on the DJ glove, which will display the selected color on the glove's LEDs. The camera system will also acquire the color of the LEDs. The camera system will send this information to an embedded device, which will determine the gesture being performed. Based on the gesture detected, the embedded device will send a command (which had been previously mapped to the gesture) to the DJ software. The DJ software will then execute the effect that corresponds to the gesture performed.

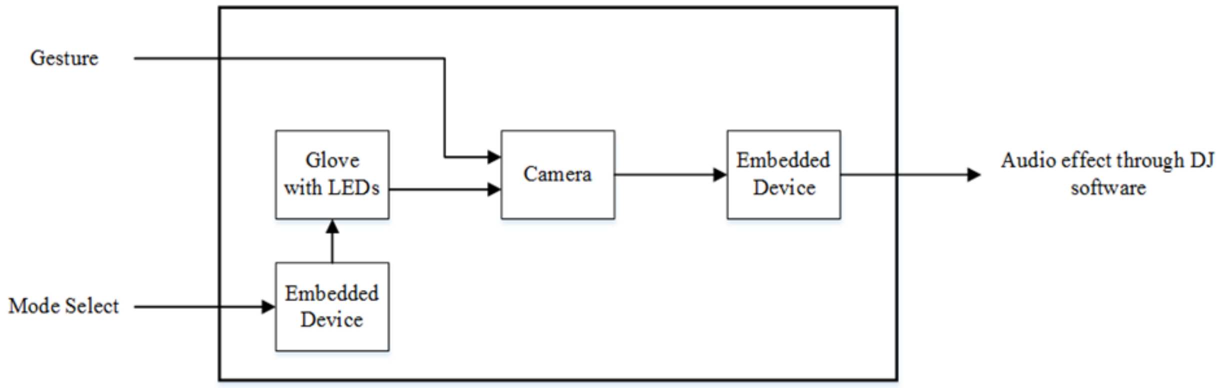


Fig. 2. Subsystem block diagram

3) *System State Diagram*

The top-level state diagram, Fig. 3, illustrates the overall flow of the system. The system will start with an initialization, and then begin to track the glove's LEDs in search for a predefined hand motion and color combination. When a gesture is recognized, the system then moves on to the next state where the audio effect is executed. Once the audio effect has been executed, the system will begin searching for another gesture.

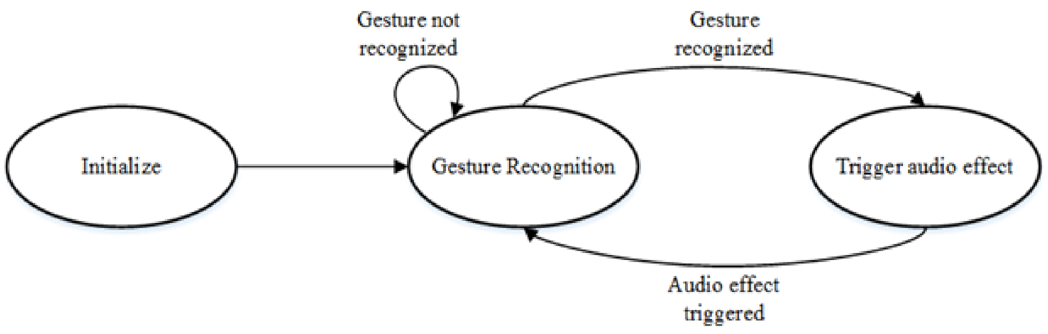


Fig. 3. Top-level state diagram

The glove component introduces a separate process to be considered; therefore, the glove component has its own state diagram illustrated in Fig. 4. Upon initialization, the DJ can select a predefined LED mode. At any time, the DJ can select another mode. The mode will select the color of the LEDs displayed by the glove.

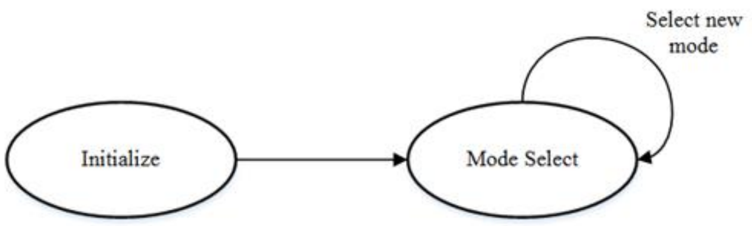


Fig. 4. Glove state diagram

4) *Division of Labor*

The division of labor was divided evenly between the group members; however, the responsibilities were determined to not be solely on one group member. For example, one member was in charge of developing the HMM and was responsible for communicating with the other group members about changes in understanding or implementation of the algorithm. Every member had responsibility for some aspect within each element of the design. Table 5 shows an overview of the assignment of responsibility.

TABLE V. HIGH LEVEL DIVISION OF LABOR

Andrew	**Evan**	**Theo**
MIDI communication	Glove circuit design	HMM
Mixxx command mapping	Glove integration	Glove embedded device code
Pixy object detection	Pixy-Raspberry Pi communication	Gesture trajectory calculation

See Appendix E for a detailed division of labor.

5) *Design Methods*

1. *Glove*

The first major component for the entire system is the glove. The glove system can be divided into four key hardware components: a Trinket Pro [13], the tri-color LEDs, 2N2222A BJT transistor, and the circuit box. The Trinket is a small microcontroller board made by Adafruit. This board generates three PWM signals that are used to control the colors of the tri-color LEDs. The transistor behaves as a switch between the three signals and the tri-color LEDs emits different color light based on signal sent from the Trinket. The entire circuit is contained in the 3D printed circuit box with the LED wire leads emerging from the box and connecting the LEDs at the fingertips. This can be seen in Fig. 5. A detailed circuit schematic is found in Appendix XIV.

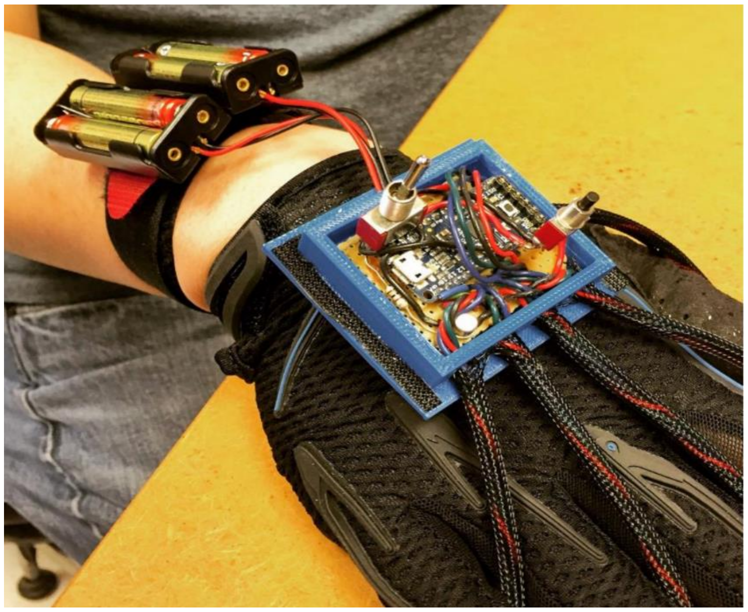


Fig. 5. Glove design

The Trinket Pro embedded device executes C code that generates PWM signals and waits for a switch toggle signal. The Trinket Pro has three pins that send PWM signals. A pin is used to compare voltage levels to determine a command to switch between PWM signals. A push button is used to switch between the 5V source and the Trinket Pro pin. Also used was a switch to toggle the power supply. Four AAA batteries are used to power the glove. These are attached to a wrist strap with Velcro.

2. *Pixy camera*

The second major hardware component is the Pixy camera. The Pixy camera is used to detect the LEDs as objects with color, coordinate position, and object size. The Pixy also includes a digital signal processor (DSP) that processes the images acquired from the Pixy and determines the Cartesian coordinates, size, and color signature of all objects detected. This information is sent to an embedded device for additional processing.

Pixy is controlled through Pixymon, which is software that is executed on a computer. Pixymon allows the user to adjust the properties of the detection process. Properties of the detection process that can be adjusted are the maximum number of blocks, color signature sensitivities, brightness, and auto white light exposure correction.

3. Raspberry Pi

The third major hardware component is the Raspberry Pi. The Raspberry Pi will receive the object information from the Pixy camera and calculate the trajectory of the LEDs. This will be done by first calculating the center of the hand in each frame.

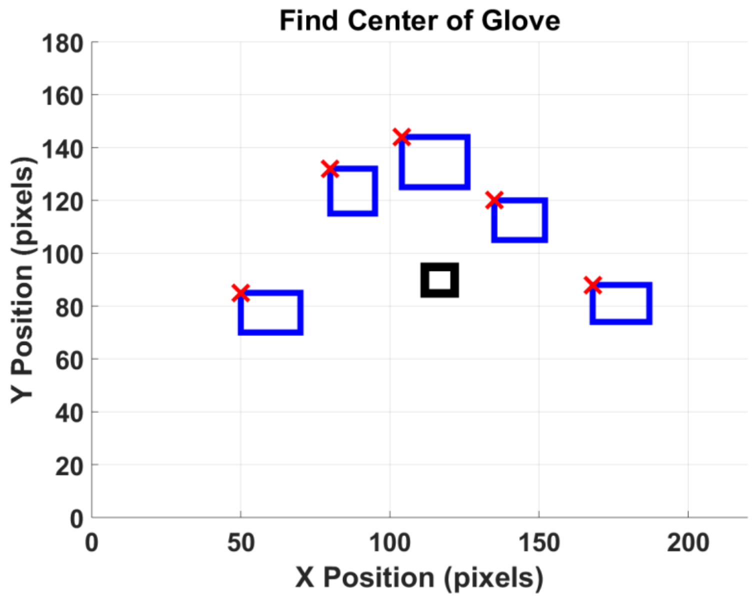


Fig. 6. Finding the center of the glove

In Fig. 6, a frame generated from the Pixy camera is shown. The red “X” indicates the Cartesian coordinate of each object detected and the blue boxes indicates the size of each object that is also sent. The objects detected are the LEDs on the fingertips of the DJ’s glove. From these detected objects, the center of the hand can be determined by finding the average of their Cartesian coordinates. The position of the center of the hand is saved for each frame.

A sequence of frames, in which the center of the hand is found, is saved to the Raspberry Pi. This sequence of Cartesian points will model the path of the gesture performed. To eliminate occasional inconsistencies in the gesture acquisition, the Cartesian coordinates of each frame are filtered using the following formulas.

$$x(i) = \frac{x(i) + x(i-1)}{2} \tag{1}$$

$$y(i) = \frac{y(i) + y(i-1)}{2} \tag{2}$$

The following plot shows a gesture that was acquired by the Pixy camera. The data was saved to MATLAB and plotted in Fig. 7. This was a left to right loop with the red LEDs on.

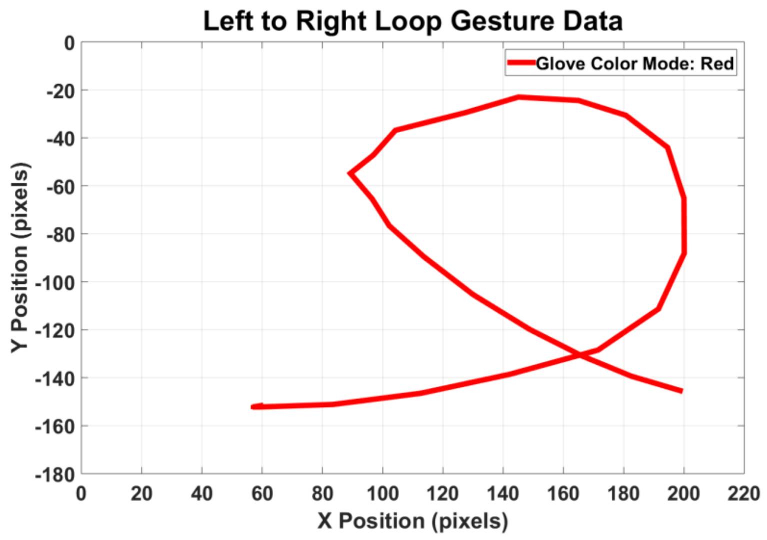


Fig. 7. Left to right loop gesture data

The angles between each sequential data point in Fig. 7 will be used by to determine which gesture was performed.

A HMM will be used to recognize the gesture being performed. This algorithm will be run on the Raspberry Pi. The HMM uses the angles from the gesture performed as observations. Using predefined transition and emission probabilities, the most likely gesture is predicted. For more information on the HMM, see Appendix D. Results of the HMM can be seen in Appendix E.

Once a gesture is recognized, the Raspberry Pi sends a MIDI command to the DJ's computer to execute the audio effect that corresponds with the performed gesture.

4. *Audio effect execution*

The final hardware component is a computer. In the implementation of the design, a Macintosh computer is used. The computer will receive a MIDI command from the Raspberry Pi and map that MIDI command to an audio effect. The computer will execute that audio effect, and the results are observed through the computer's speakers.

i. MIDI communication

In the music production world, musical instrument digital interface (MIDI) signal communication is the industry standard for instrument or audio effect command transmission. A MIDI signal is a collection of three bytes. The first of the three is the status byte, which contains the op-code and a channel number. The op-code provides the receiver with information about which type of music command is being sent. A channel number determines the MIDI channel that will receive the information. The two bytes that follow contain data information. For example, Fig. 8 illustrates a sample MIDI signal that controls a digital piano with the first data byte establishing the note and the second determining the velocity at which it will be played. The DJ software that receives the MIDI command operates similarly with data bytes that determine an effect followed by the degree of change within its parameter.

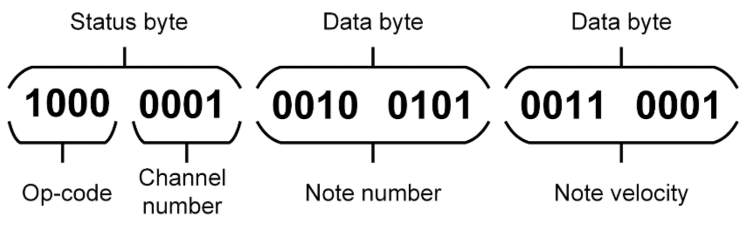


Fig. 8. MIDI protocol

ii. MIDI learning wizard

MIXXX is an open source DJ software that receives and performs audio effect commands within the system. This software is equipped with a "MIDI Learning Wizard" that allows the user to map any MIDI signal to their desired audio effect. In this way, a custom MIDI command, received by the computer's serial port, is "learned," and then mapped to the audio effect selected by the user.

5. Interfacing

There are two communication channels between three major elements within the system. The Pixy camera will use serial peripheral interface (SPI) protocol to communicate with the Raspberry Pi. The second interfacing channel connects the Raspberry Pi to the DJ software within a computer. This channel uses universal asynchronous receiver transmitter (UART) communication protocol to transmit custom MIDI signals to be received by the serial port of the DJ's computer.

The first of the communication channels connects the Pixy camera to the Raspberry Pi. Charmed Labs provides several libraries to facilitate transmission of data to the Raspberry Pi through SPI communication [14] [15]. SPI uses a synchronous protocol that involves a master-slave architecture. The Raspberry Pi, providing the clock signal, or timing information, is the master device. The Pixy Camera serves as the slave device and transmits object information for use in trajectory calculations.

Communication was also established between the Raspberry Pi and the DJ's computer. MIDI signal communication is a standard method for transmitting instrument or audio effect commands. The MIDI communication protocol is a unique subset of UART communication. It requires that the baud rate be set at 31,250 bits per second. The steady state signal is at a constant logic high. Furthermore, MIDI communication uses a '0' start bit and a '1' stop bit to parse the information being sent.

The DJ's computer uses a universal serial bus (USB) port to accept serial data. Therefore, UART communication is established through a MIDI-to-USB conversion cable. The general-purpose input output (GPIO) pins on the Raspberry Pi relay binary information to initiate this UART protocol. Three wires are required to attach the GPIO pins to the cable. The ground, 3.3-volt, and transmission (TX) pins on the Raspberry Pi are connected to the corresponding MIDI pins. The MIDI-to-USB cable, shown in Fig. 9, then converts these signals into information packages accepted by the USB port.

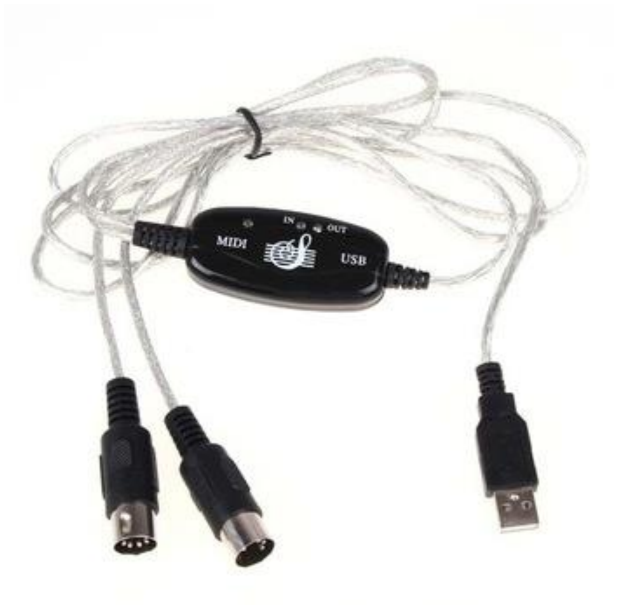


Fig. 9. MIDI-to-USB cable

H. Economic Analysis

An exact list of parts and their prices can be seen in Appendix D. The cost of the DJ Glove is approximately $180. The DJ Glove will be funded by the Electrical and Computer Engineering Department of Bradley University.

III. Design Testing and Validation

The testing of the design is founded upon the functional requirements of the system. The functional requirements are shown in Table 4.

For the glove system, the goal is to display predefined color schemes through tri-color LEDs. The method for testing that function requires PWM signal measurements. An oscilloscope reading will confirm a change in the PWM signal sent. From the implementation, the results show a successful switch between three emitted colors and a successful switch between three 5V PWM signals.

For the Pixy camera system, the goal is to acquire the user's gestures. There are four specifications that guide the testing procedure. Frames per second, pixel resolution, success rate, and lighting condition are the properties to test and confirm. Measuring the period of image acquisition, acquiring test gestures, and confirming the image properties in an environment of 250 candela are the methods to test the achievement of the goal. The specification for the image is a 400 x 240-pixel resolution, so retrieving a saved image's properties will determine the pixel resolution. Ten frames/second is another specification, so measuring the period with software is the method used. The frames/second is acquired and displayed by a Raspberry Pi. Finally, in order to test success rate, test gestures are performed. The data from Pixy is acquired and saved to a computer, and MATLAB is used to process the information and plot the results of the acquired gesture.

The goal of the HMM algorithm is to recognize user's gestures. The specifications to test the achievement of this goal are a 75% success rate with a latency of 160 milliseconds. This latency will be defined as the time from the end of gesture to when the audio effect is executed.

The goal of the Mixxx software is to trigger an audio effect specified by the gesture and LED color combination. The method of testing the achievement of this goal is to view the MIDI signal being sent from the Raspberry Pi with an oscilloscope to confirm that the corresponding signal has been sent as well as aural confirmation of the audio effect being executed. A MIDI signal is designed and sent via Raspberry Pi through a MIDI to USB converter into a computer. Mixxx software maps a MIDI command with an audio effect specified by the user. Validation of mapping is shown with the user interface of Mixxx.

IV. Acknowledgments

This project was funded by the Electrical Engineering Department of Bradley University and advised by Dr. José Sánchez.

V. Conclusion

Over the last several decades, music technology has shifted toward the digital realm, bringing the tools required for music creation and manipulation within the reach of more artists than ever before. DJs, in particular, rely on computer software to carry out audio effects that enhance their performances. The DJ Glove aims to integrate the natural expressions and movements of an artist with the software used during a performance. A glove equipped with tri-color LEDs not only adds to the artistic side of the DJ's performance but also expands the range of possible audio effects by combining the gestures with color information. Implementation of the HMM algorithm will connect gesture with MIDI commands, completing a gesturally controlled DJ system.

VI. References

[1] P. Premaratne, Human computer interaction using hand gestures. New York: Springer-Verlag, 2014.
[2] 2015. [Online]. Available: http://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf. [Accessed: 14- Oct- 2015].
[3] Hyeon-Kyu Lee and J. Kim, "An HMM-based threshold model approach for gesture recognition", IEEE Trans. Pattern Anal. Machine Intell., vol. 21, no. 10, pp. 961-973, 1999.
[4] G. ten Holt, M. Reinders and E. Hendriks, "Multi-Dimensional Dynamic Time Warping for Gesture Re".
[5] Uwyn.com, "GECO - Music and sound through hand gestures", 2015. [Online]. Available: http://uwyn.com/geco/. [Accessed: 07- Apr- 2015].
[6] J. French, R. LeGrand, 'blog', Charmed Labs, 2015. [Online]. Available: http://charmedlabs.com/default/. [Accessed: 14- Oct- 2015].
[7] Americandj.eu, "Snap Shot LED"., 2015. Web. 13 Apr. 2015.
[8] Avagotech.com, "High Brightness Tricolor PLCC-6 Black Body LED"., 2015. Web. 13 Apr. 2015.
[9] Charmed Labs, "Pixy (Cmucam5)"., 2015. Web. 13 Apr. 2015.
[10] Biology.clemson.edu, "Literature Review on Reaction Time", 2015. [Online]. Available: http://biology.clemson.edu/bpc/bp/Lab/110/reaction.htm. [Accessed: 07- Apr- 2015].
[11] Mixxx DJ Software, 'Mixxx - Free MP3 DJ Mixing Software', 2015. [Online]. Available: http://mixxx.org. [Accessed: 14- Oct- 2015].
[12] Raspberry Pi, 'Raspberry Pi - Teach, Learn, and Make with Raspberry Pi', 2015. [Online]. Available: https://www.raspberrypi.org/. [Accessed: 14- Oct- 2015].
[13] T. others, 'Adafruit Pro Trinket - 3V 12MHz ID: 2010 - $9.95: Adafruit Industries, Unique & fun DIY electronics and kits', Adafruit.com, 2015. [Online]. Available: https://www.adafruit.com/products/2010. [Accessed: 14- Oct- 2015].
[14] GitHub, 'omwah/pixy_rpi', 2014. [Online]. Available: https://github.com/omwah/pixy_rpi. [Accessed: 14- Oct- 2015].
[15] Wiringpi.com, 'WiringPi', 2015. [Online]. Available: http://wiringpi.com/. [Accessed: 14- Oct- 2015].
[16] Lee, Hyeon-Kyu, and Jin Kim. A HMM-Based Threshold Model Approach for Gesture Recognition. 1st ed. 2015. Web. 15 Oct. 2015.
[17] "HMM Calculations", Indiana State University, 2016. [Online]. Available: http://www.indiana.edu/~iulg/moss/hmmcalculations.pdf. [Accessed: 16- Dec- 2015].
[18] "Types of Markov models", Sheffield University. [Online]. Available: http://mas115.group.shef.ac.uk/projects/project2a/group23/typesofmarkov.html. [Accessed: 16- Dec- 2015].
[18] Mixxx.org, '4. Configuring Mixxx — Mixxx User Manual', 2015. [Online]. Available: http://mixxx.org/manual/latest/chapters/configuration.html#latency. [Accessed: 15- Oct- 2015].

VII. Appendix

A. Experimental Results

A. Overview

The objective for the DJ Spatial Tracking and Gesture Recognition for Audio Effects and Mixing project was to create a glove that allows DJs to gesturally control a musical performance. The system designed to achieve this goal considers four main system sections: the glove, gesture acquisition, gesture recognition, and audio effect execution. The experimental results from the implementation and integration of these subsystems are outlined below.

B. Glove

Constraints were placed on the design of the glove to ensure that it would not hinder the DJ's ability to perform. Emphasis was also placed on the importance of aesthetics. Thus, a sleek, black, and form-fitting glove was chosen as the base for the design. With three PWM signals being routed to each fingertip, a total of four stranded wires are guided down the finger with two Velcro straps to secure their position. A three-dimensionally printed circuit case was designed to house the embedded device, button, and switch circuitry.

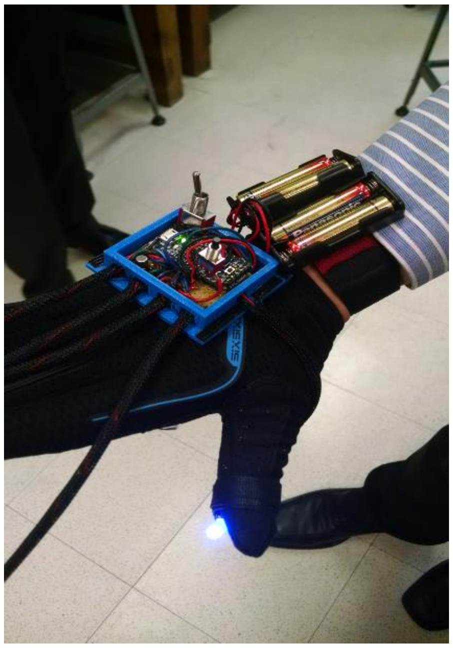


Fig. 10. Glove on hand

The final glove result weighed 6 ounces, including the batteries, and is worn by the user with extremely little hindrance to the hand's mobility. With 100% reliability, the DJ can press a button cycle through the three color scheme modes. Oscilloscope readings confirm the circuit's ability to send 25%, 50%, 75%, and 100% duty cycles. Each color scheme usage has a slightly different current draw, resulting in three different battery life calculations. The measurements below are performed with 100% duty cycle PWM signals:

TABLE VI. BATTERY LIFE CALCULATIONS

Red	794 mAh ÷ 74 mA = 10.7 hours
Blue	794 mAh ÷ 62 mA = 12.8 hours
Green	794 mAh ÷ 61 mA = 13.0 hours
Average	**12.2 hours**

A DJ performance is usually less than three hours. A battery life of 12.2 hours is more than sufficient for the average DJ's requirements. With aesthetics and gesture acquisition in mind, the design calls for sufficiently bright LED performance. The transistor circuit is carefully designed to minimize power usage to ensure that components do not overheat. Detailed circuit measurements are provided below.

TABLE VII. CIRCUIT CALCULATIONS

Color	Diode voltage (V)	CE voltage (mV)	Base voltage (V)	Base current (mA)	Collector current (mA)
Red	2.037	108.8	0.771	5.200	67.6
Blue	3.101	71.70	0.761	5.200	46.50
Green	3.166	73.02	0.759	5.200	47.05

C. *Gesture Acquisition*

To acquire a gesture and color signature, the Pixy camera, and image processing system uses color-based object detection. The LEDs on the fingertips are recognized as an object of interest within the frame. The system was able to acquire these images and recognize objects in an environment that exceeds ambient lighting of over 250 candela. A virtual box is placed over the objects of interest providing the system with the location, size, and color of the LEDs within the frame. Example frames from the object detection process are provided in the figures below.

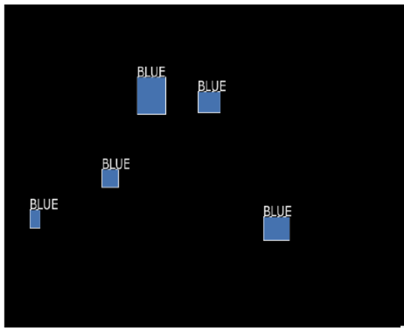


Fig. 11. Blue LEDs

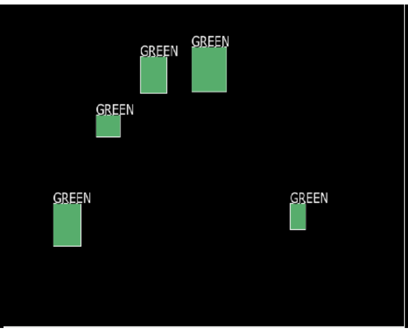


Fig. 12. Green LEDs

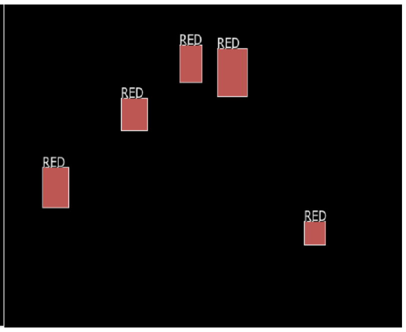


Fig. 13. Red LEDs

From the data provided by these images, the system can calculate the average center of the glove within each frame. From frame to frame, as the DJ performs a gesture, trajectory information is compiled. The angles between the glove's locations in a series of frames are quantized in predefined angle bins. The quantized angles are then provided to the HMM for processing. Illustration of the glove center calculation process and angle bins are provided in Fig. 6 and Fig. 19 respectively.

To validate and observe the gesture acquisition, trajectory information is uploaded into MATLAB. The program plots the gesture data using the line color that corresponds with the color signature detected. With 20 trials performed, 100% of the gestures and corresponding color signatures were acquired. Examples of these plots are provided in Appendix J.

D. Gesture Recognition

Please refer to hidden Markov model test results in Appendix E.

E. Audio Effect Execution

To validate and test the MIDI signals, as well as the software's ability to recognize them, a program was written on the Raspberry Pi to communicate with Mixxx. Based on a keyboard button press, the program transmits a MIDI signal from the GPIO pins to the MIDI-to-USB cable. The "MIDI Learning Wizard" is trained to recognize a custom MIDI command and map it to the audio effect chosen by the user. MIDI commands were created, transmitted, received, recognized, and executed. The effects performed were play/pause, jump-to-start, increase beats per minute, decrease beats per minute, and a "kill the mids" equalizer effect. These effects were validated through visual assessment within the graphical user interface of the software as well as through the audio from the speakers.

B. Code

The following code was used to control the color of the LEDs on the glove. It reads the state of a pin that is connected to a button. When the pin reads high, the code cycles to the next color mode. Three color modes are possible. A debouncer is also included.

```
const int buttonPin = 4;    // the pin that the push button is attached to
const int pwmPin1 = 9;       // pwm outputs
const int pwmPin2 = 10;
const int pwmPin3 = 11;

int counter = 0;   // count # times button is pressed
int state = 0;         // current state of the button
int lastState = 0;     // previous state of the button

// initialization
void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(pwmPin1, OUTPUT);
}

void loop() {

  // read current state of button
  state = digitalRead(buttonPin);
  // if there is a change in states (button was pressed)
  if (state != lastState) {
    // only on button press (not release)
    if (state == HIGH) {
      counter++;
      // restart counter since there are only 3 states
      if (counter == 3) {
      counter = 0;
      }
    }
    // debouncer
    delay(50);
  }
  // prepare for next loop
  lastState = state;

  // define states and what PWMs will run in each state
  if (counter % 3 == 0) {
    analogWrite(pwmPin1, 255*1);
    analogWrite(pwmPin2, 255*0);
    analogWrite(pwmPin3, 255*0);
  } else if (counter % 2 == 0) {
    analogWrite(pwmPin1, 255*0);
    analogWrite(pwmPin2, 255*1);
    analogWrite(pwmPin3, 255*0);
  } else {
    analogWrite(pwmPin1, 255*0);
    analogWrite(pwmPin2, 255*0);
    analogWrite(pwmPin3, 255*1);
  }

}
```

The code for the HMM MATLAB simulation is extensive and thus can be accessed in a .zip format at the following link. http://cegt201.bradley.edu/projects/proj2016/djglove/deliverables/hmm_simulation.zip

C. Recommendations for Future Work

There are two main recommendations that should be taken into consideration if future work is going to be attempted on this project.

Custom image acquisition should first be considered. The Pixy camera was limiting in the fact that it did not allow for much customization of how objects were detected. Pixy is optimized to detect objects based on their color. However, it struggled when it came to detecting differences in the color of light emitted from the tri-color LEDs. Due to the camera sensor's low dynamic range, the color of the LEDs would typically be recognized as white light. Even with the LEDs diffused with several layers of masking tape, the relative brightness was almost too much for the camera to handle. A custom image acquisition system may be able to overcome this limitation.

Additionally, more emphasis should be placed on understanding the HMM algorithms. While the HMM has been proven to work with gesture acquisition before, it will take significant effort to create a working simulation. Several of the algorithms have been successfully code already, however implementation of all HMM algorithms together should be a major focus.

D. HMM Explanation

The HMM is a complex subject. To introduce the topic, an example has been created.

Suppose there is a large gumball factory as seen in Fig. 14. Inside the gumball factory are three large gumball machines, each with three different colors of gumballs: blue, green, and purple. Every second one of these machines drops a gumball.

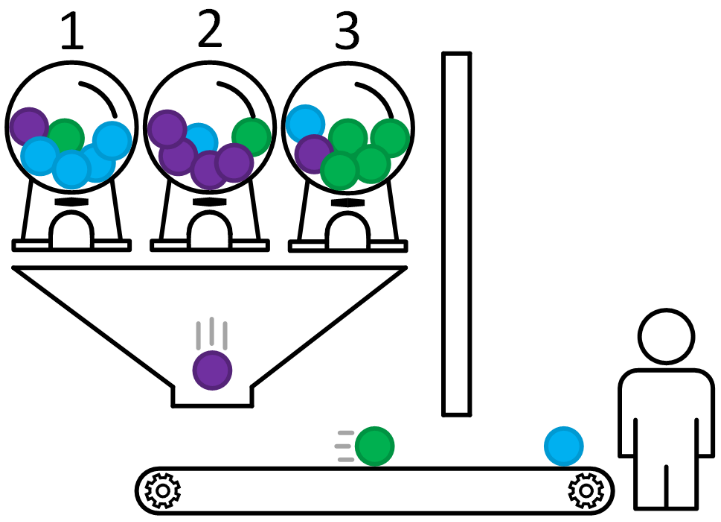


Fig. 14. Gumball factory

Dwight, the friendly gumball factory worker, stands outside the factory and watches as gumballs roll out the factory door. However, he can only see the gumballs on the conveyer belt – he cannot see the machines dropping the gumballs since the gumballs are behind the factory wall.

Dwight's goal is to guess the sequence of machines that dropped the sequence of gumballs in front of him. He can make an intelligent guess about this because he knows two important sets of probabilities concerning the gumball machines.

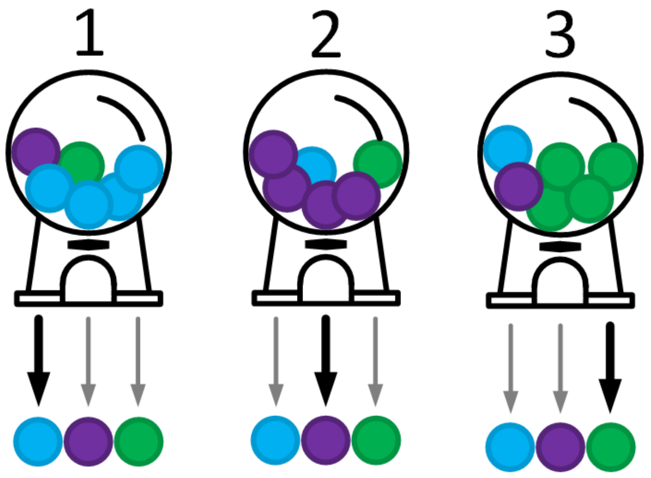


Fig. 15. Gumball machine emission probabilities

He knows that each machine has a probability of dropping a certain color gumball. In Fig. 15, the probabilities of the three gumball machines are illustrated by the thickness of the lines. For example, machine 1 has the highest probability of dropping a blue gumball; machine 2, purple; and machine 3, green. Each machine will have its own unique set of output probabilities. These are commonly called the emission probabilities.

There is also another set of probabilities that Dwight is aware of.

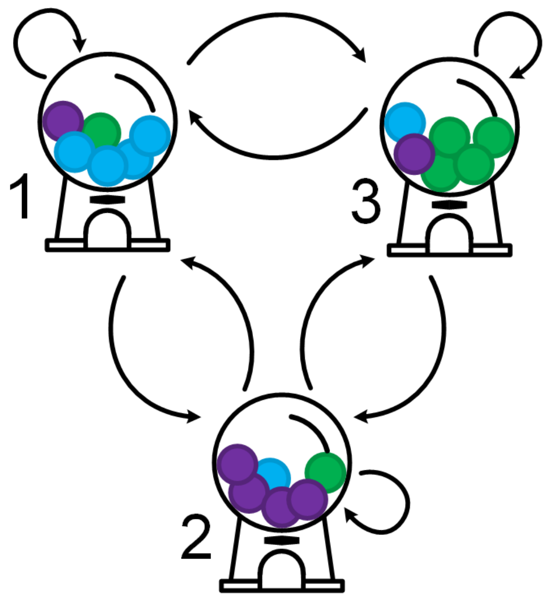


Fig. 16. Gumball machine transition probabilities

Once a machine drops a gumball, there are certain probabilities for where the next gumball will come from. It could come from any of the three machines, including the machine that dropped the previous gumball. These are called the transition probabilities.

Dwight knows both the emission and transition probabilities, and he can use these to determine the sequence of machines that dropped the sequence of gumballs in front of him. However, remember that he still cannot see the machines dropping the gumballs. It is important to him that he determines the sequence of machines since this tells him about an important part of his day – it tells him what he will have for lunch.

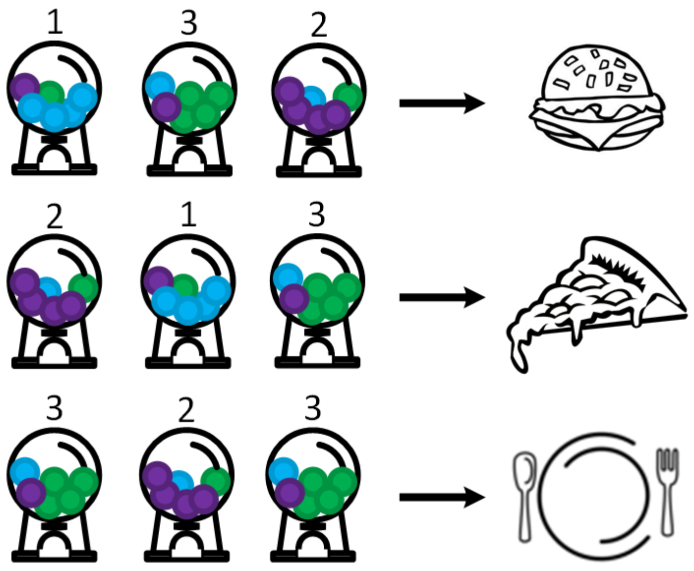


Fig. 17. Possible outcomes

For example, if Dwight finds that the sequence of machines is “1 → 3 → 2”, then this means he will be getting hamburgers for lunch. If he finds the sequence of to be “2 → 1 → 3”, then he will have pizza for lunch, and if he finds “3 → 2 → 3”, then he will be having nothing for lunch. While there are other sequences of machines that could be observed, these other sequences have no meaning to Dwight.

However, this project is about gesture recognition, not gumballs. In order to connect the HMM with gesture recognition, some definitions must first be made. Gumballs can be called observations. The gumballs on the conveyor belt can be thought of as the observation sequence. Gumballs machines are states and the succession of machines dropping gumballs can be thought of as a sequence of states. Finally, food is the result of the observations.

With regard to gesture recognition, the DJ will perform a gesture, and throughout the gesture the image recognition system will be collecting trajectory information. This information, in the form of a sequence of angles, will serve as the observations.

The states are more difficult to understand but can be thought of as hidden or abstract representations of angles. The sequence of states would therefore be an abstract representation of an entire gesture. Finally, the result of the observation would be the audio effect applied by the DJ software.

There are several important things to note about the HMM.

- **States transition with time.** With the gumball machines, once a gumball had been dropped, there is a certain probability that determines which machine would drop the next gumball. Since the gumball machines represented states, these can be thought of as transitioning with time.
- **The goal of the HMM is to estimate the state sequence.** Remember that Dwight was trying to guess the sequence of machines that dropped the sequence of gumball on the conveyer belt in front of him.
- **States are always hidden.** Dwight could not see inside the factory where the machines were located.
- **The HMM correlates observations with a state sequence.** Dwight was trying to correlate the sequence of gumballs in front of him with the sequence of machines that dropped them.

The HMM consist of three primary matrices. The transition matrix: *A*, which contains the state transition probabilities; the emission matrix: *B*, which contains the output probabilities; and the initial condition matrix: π, which contains the initial state distribution. These three matrices need to be trained beforehand for the HMM to perform optimally.

On a more theoretical basis, the HMM can be illustrated as shown in Fig. 18.

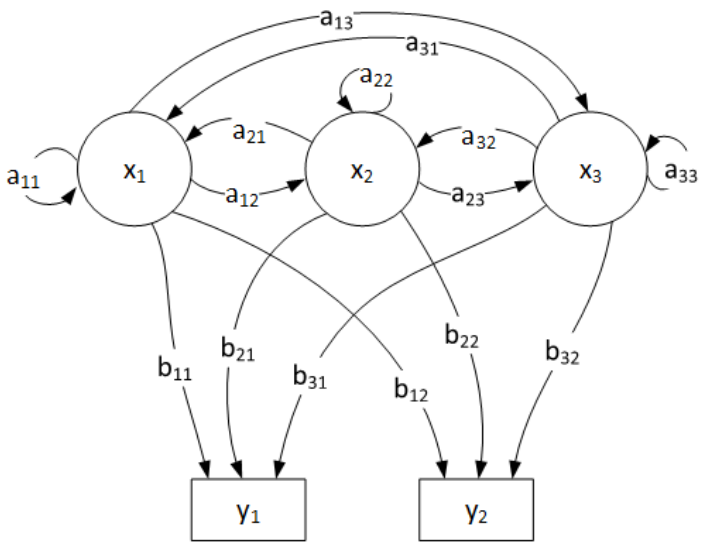


Fig. 18. HMM example

The states are the circles labeled as x_1, x_2, and x_3. The observations are the squares labeled as y_1 and y_2. The state transition probabilities are the arrows going between the circles (states) and the emission probabilities are the lines going from the circles (states) to the squares (observations). The image recognition system will be recording the position of the hand. The angle between every two frames captured will be used as the observation data. These angles will need to be quantized for use for the HMM. The following figure illustrates how angles will be quantized.

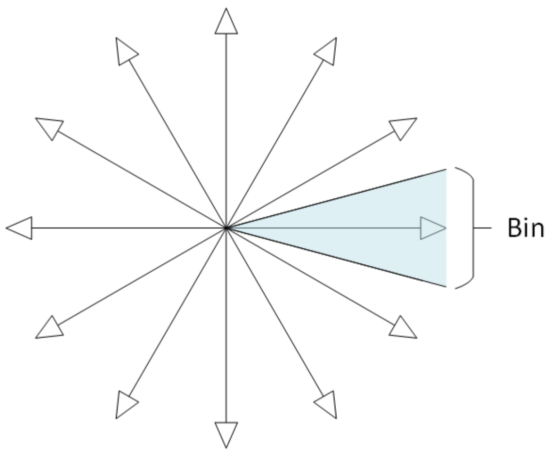


Fig. 19. Quantized angle bins

Angles will be rounded into "bins" with bins being equally divided among 360°. To demonstrate this, test gesture data was entered into MATLAB – three loop gestures and three left-to-right swipes.

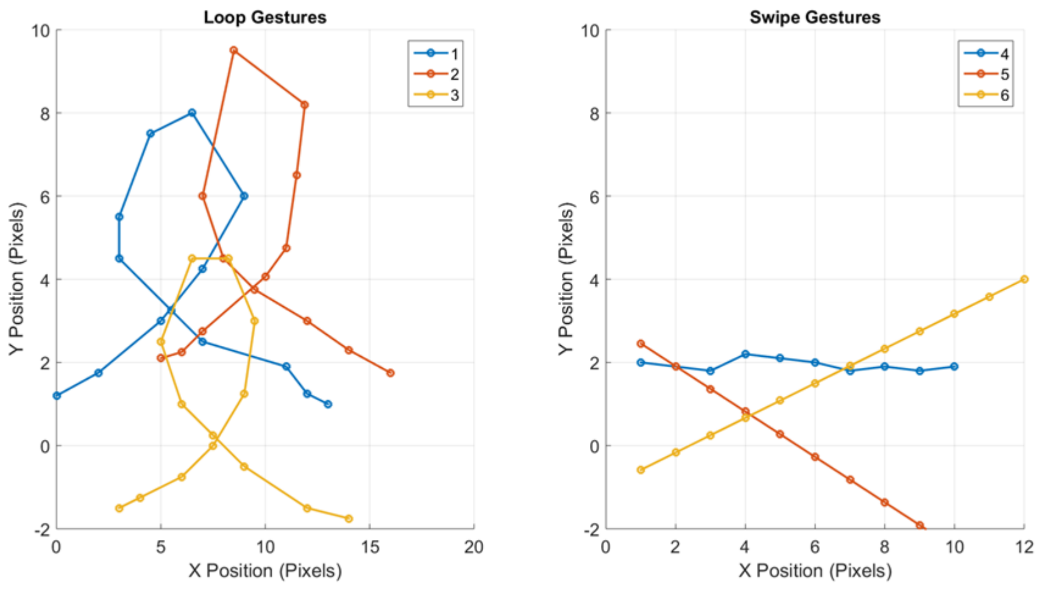


Fig. 20. Test gestures

From these gestures, the angles between every two points were extracted.

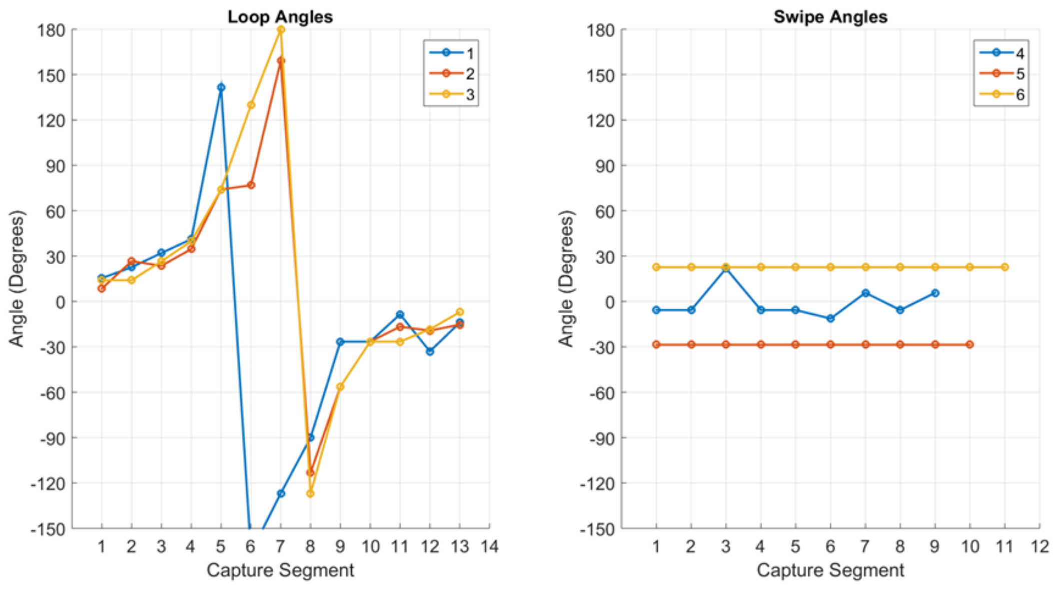


Fig. 21. Gesture angles

The angles were then quantized using 12 bins. This means that the angles were rounded to the nearest 30 degrees. These angles can now be used by the HMM to determine the gesture performed.

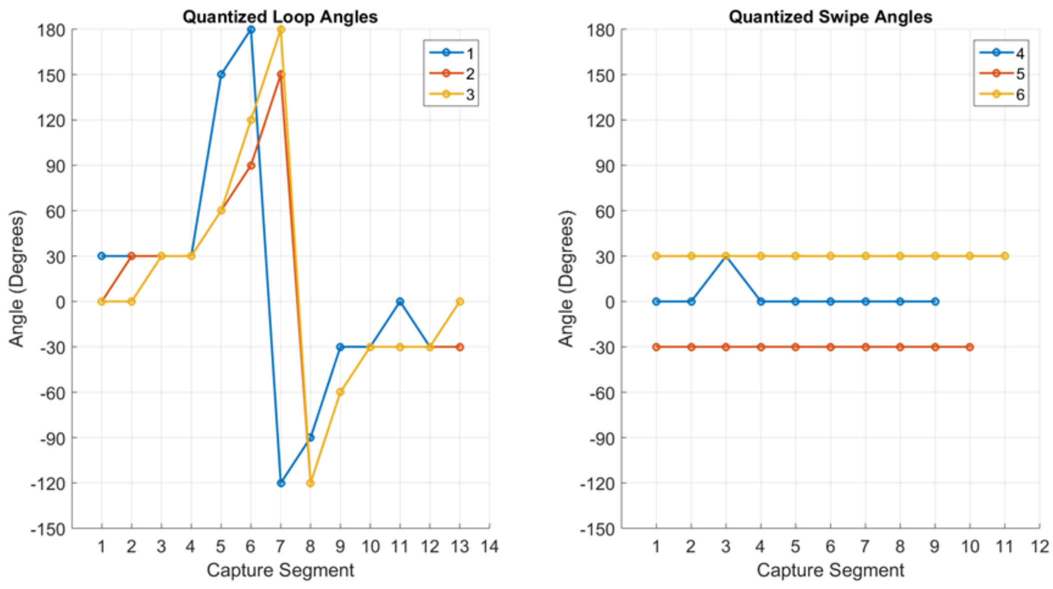


Fig. 22. Quantized gesture angles

There are three problems that have to be solved when working with HMMs. They are as follows.

1) **Classifying**: finding the probability of observing a sequence of observations
2) **Decoding**: finding the best sequence of states that explains the observed sequence of observations
3) **Training**: training parameters from the observations

Descriptions of the algorithms that make up these steps are as follows.

The Baum-Welch algorithm [16] is used to find the unknown parameters of a HMM. It uses the forward and backward algorithms to do this. These work from opposite ends of the gesture to find the probability of seeing a sequence of observations and being in some state at a certain time. Using both these algorithms, update transition and emission matrices can be generated.

The second algorithm used is the Viterbi algorithm [16]. This is used to find the most probable sequence of states based on the sequence of observations.

The final algorithm used is the expectation maximization algorithm [16]. This is used to find the values of the transition and emission matrices that make the observation data most likely. This is used in conjunction with the threshold model. [16] Demonstrating the threshold model is Fig. 23.

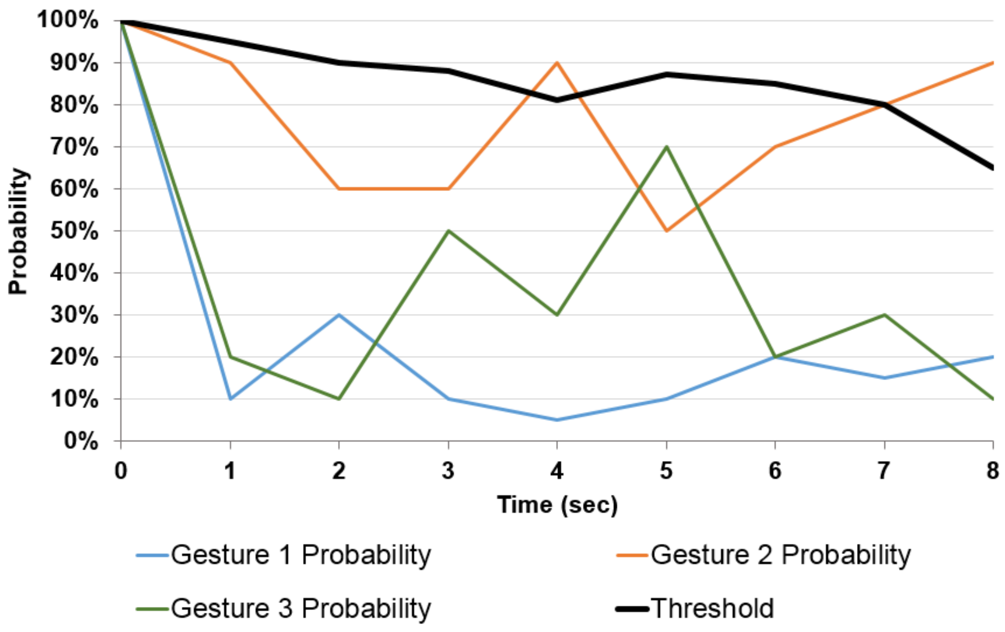


Fig. 23. Threshold model

The black line is the threshold for determining the end of a gesture. The other three lines are the probabilities of the gestures. These are all changing as the actual gesture is performed. Note that at time “4”, the probability of gesture 2 rises of the threshold. This means that the HMM can begin looking for the end point of this gesture.

E. HMM Results

The first algorithm in the HMM is the Forward-Backward algorithm. This generates the α and ß matrix which is used by other algorithms later in the HMM. We tested the HMM simulation results against those of Indiana State University [17]. The following initial conditions were used.

Initial Condition Matrix (π)

- s = 0.85
- t = 0.15

States: *s,t*
Observations: *A,B*

Fig. 24 shows the transition and emissions probabilities used in the HMM simulation.

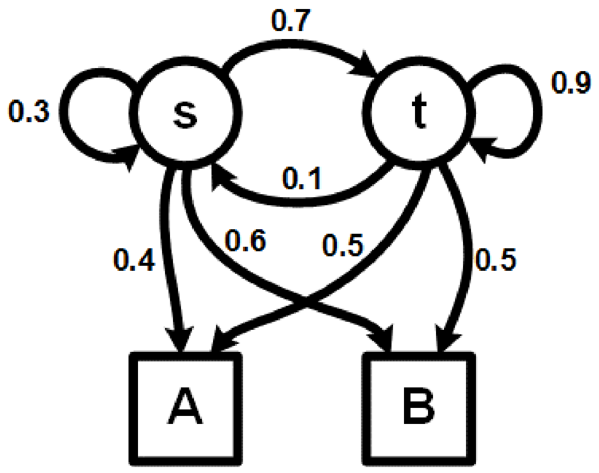


Fig. 24. Initial conditions for HMM simulation

As seen in the following figures, the HMM simulation generated very similar results to Indiana State indicating that the Forward and Backward algorithms had been implemented correctly.

TABLE VIII. α MATRIX FROM INDIANA STATE UNIVERSITY

Iteration	**s**	**t**
1	0.3400	0.08000
2	0.0660	0.1550
3	0.0212	0.0929
4	0.0063	0.0492

Total probability = 0.00625 + 0.04919 = **0.05544**

TABLE IX. α MATRIX FROM HMM SIMULATION

Iteration	**s**	**t**
1	0.3400	0.0750
2	0.0657	0.1528
3	0.0210	0.0917
4	0.0062	0.0486

Total probability = 0.0062 + 0.0486 = **0.0548**

TABLE X. ß MATRIX FROM INDIANA STATE UNIVERSITY

Iteration	**s**	**t**
4	0.1331	0.1273
3	0.2561	0.2487
2	0.4700	0.4900
1	1.0000	1.0000

Total probability = 0.1331 + 0.1273 = **0.2604**

TABLE XI. ß MATRIX FROM HMM SIMULATION

Iteration	s	t
4	0.1332	0.1273
3	0.2561	0.2487
2	0.4700	0.4900
1	1.0000	1.0000

Total probability = 0.13315 + 0.12729 = **0.26044**

A similar test procedure was performed for the Viterbi algorithm, which generates the γ matrix. The HMM simulation results were compared to Sheffield University [18]. The initial conditions for the test are stated below.

Initial condition matrix (π)

- s is 0.6
- t is 0.4

States: *s*,*t*
Observations: *A*,*B*,*C*

Fig. 25 shows the transition and emissions probabilities used in the HMM simulation.

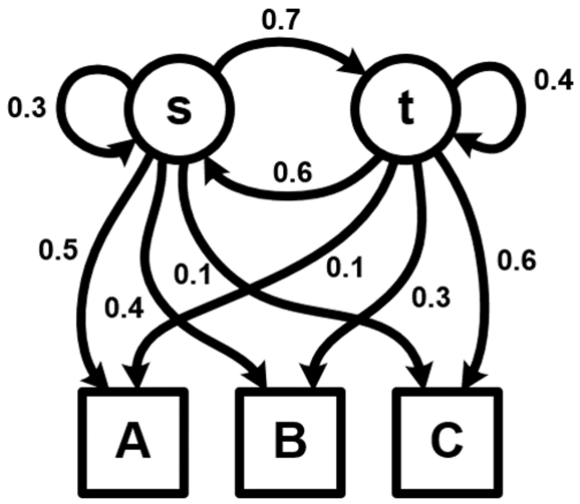
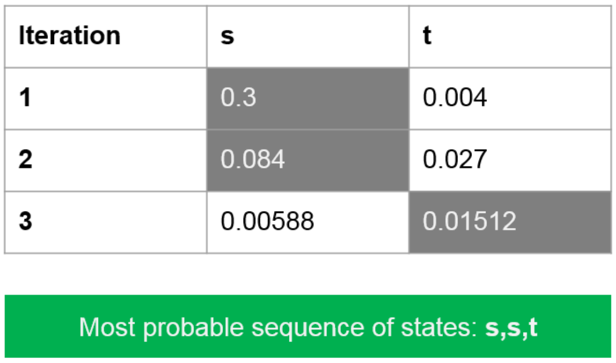


Fig. 25. Initial conditions for HMM simulation

As seen in the following figures, the HMM simulation generated very similar results to Sheffield University indicating that the Viterbi algorithm had been implemented correctly.

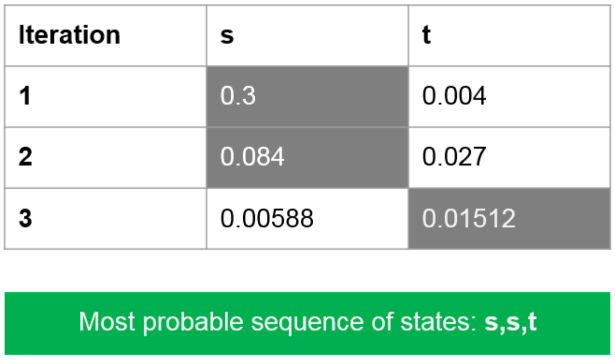
TABLE XII. γ MATRIX FROM SHEFFIELD UNIVERSITY



Iteration	s	t
1	0.3	0.004
2	0.084	0.027
3	0.00588	0.01512

Most probable sequence of states: **s,s,t**

TABLE XIII. γ MATRIX FROM HMM SIMULATION



Iteration	s	t
1	0.3	0.004
2	0.084	0.027
3	0.00588	0.01512

Most probable sequence of states: **s,s,t**

F. Nonfunctional Requirements

Table 14 compares the two nonfunctional requirements for this project. Note how the easy-to-use objective has a higher score than portable, indicating that the easy-to-use objective has the highest priority.

TABLE XIV: COMPARISON OF OBJECTIVES

	Easy-to-Use	Portable	Score
Easy-to-Use		1	1
Portable	0		0

Each nonfunctional requirement has been given a metric below along with a corresponding quantifiable measurement.

Objective: Easy to Use
Units: Ratings of design team's assessment of ease of use
Metric: Assign points according to the following scale

● Very easy to use (greater than 75% gesture recognition success)	5 points
● Easy to use (between 65% to 75% gesture recognition success)	4 points
● Can be used (between 50% to 65% gesture recognition success)	3 points
● Hard to use (between 20% to 50% gesture recognition success)	2 points
● Very hard to use (less than 20% gesture recognition success)	1 point

Objective: Portable
Units: Ratings of design team's assessment of portability
Metric: Assign points according to the following scale

● Very portable (between 0 lb to 3 lb)	5 points
● Portable (between 3 lb to 5 lb)	4 points
● Semi-portable (between 5 lb to 7 lb)	3 points
● Little-portability (between 7 lb to 10 lb)	2 points
● Not portable (greater than 10 lb)	1 point

G. Parts List and Cost Analysis

Table 15 lists the components of the primary solution for the project and their individual costs.

TABLE XV. PARTS LIST AND COST FOR PRIMARY SOLUTION

Device	**Item Cost**
Adafruit Pro Trinket	$9.95
10 Red-Green-Blue (RGB) LEDs	$14.90
Glove	$17.98
Miscellaneous	$40.00
Pixy	$69.00
Raspberry Pi	$35.00
Mixxx Software	$0
2N2222A Transistor	$0.20
Velcro	$5
Wire Sheaths	$7
Total Cost	**$171.93**

H. Division of Labor

Table 16 shows the division of labor.

TABLE XVI. DIVISION OF LABOR

TW: Theo Wiersema *AH: Andrew Hamblin* *EL: Evan Leong*

Create HMM in MATLAB (forward algorithm)	AH
Create HMM in MATLAB (backward algorithm)	TW
Create HMM in MATLAB (Viterbi algorithm)	TW
Create HMM in MATLAB (combine all previous work)	TW
Initialize HMM in MATLAB	TW
Test/Debug HMM in MATLAB with recognition success rate	All
Test/Debug HMM in MATLAB with garbage data	All
General testing/debugging of HMM in MATLAB	All
Transfer MATLAB algorithm to C (Forward)	TW
Transfer MATLAB algorithm to C (Backward)	TW
Transfer MATLAB algorithm to C (Viterbi)	TW
Transfer MATLAB algorithm to C (Misc)	All
Test/debug gesture recognition	All
Research communication Methods	All
Send block information from Pixy	EL
Receive block information on Raspberry Pi	TW
Send MIDI command from Raspberry Pi	AH
Research Mixxx MIDI Scripting	AH
Writing JavaScript plugin for Mixxx	AH
Test JavaScript plugin for Mixxx	AH
Receive MIDI command in Mixxx	AH
Construct LED circuitry	EL
Design current amplification circuitry	AH
Construct current amplification circuitry	EL
Test/Debug LED drive circuit	EL

Write code for Trinket	TW
Test code for Trinket	TW
Build glove System	EL
Test glove System	EL
Research Pixy object detection	AH
Train Pixy to recognize LEDs	AH

I. *Test Procedures*

1) *Glove*

- Oscilloscope PWM analysis
  - A. Send 100% duty cycle PWM signal from pin 9 of the Adafruit Trinket Pro
  - B. Connect oscilloscope lead to the wire that transfers the signal
  - C. Ground the oscilloscope
  - D. Assess the oscilloscope reading to ensure correct voltage level and duty cycle
  - E. Repeat this step for pins 10 and 11 of the Adafruit Trinket Pro
  - F. Repeat entire test procedure for 25%, 75%, and 100% duty cycle signals
- Battery life
  1. Measure current through the collector junction of the transistor
  2. Measure current of the three PWM signals from the Adafruit Trinket Pro
  3. Calculate power draw of the entire circuit
  4. Calculate battery life to ensure a life suitable for a DJ performance of approximately two hours
- Power rating
  1. Calculate power draw of the transistor to ensure power draw of less than 500 mW (max rating of transistor)
- Mode select
  1. Begin in mode one
  2. Switch to mode two
  3. Switch to mode one
  4. Repeat steps 2 and 3 for 20 iterations
  5. If all iterations are successful, continue test
  6. Repeat steps 4 and 5 for switching between mode one and three and mode two and three

2) *Pixy Camera*

- Ambient light setup:
  1. Turn on lights
  2. Measure lux with phone app on same surface that camera is placed on
  3. Measure distance to lights (the table to ceiling distance)
  4. Determine candela reading (http://www.rapidtables.com/calc/light/lux-to-candela-calculator.htm)
  5. Ensure reading > 250 cd
- LED color signature detection:
  1. Complete ambient light setup procedure
  2. Connect Pixy to computer
  3. Open Pixymon application
  4. Turn on glove circuit
  5. Emit one color from the LEDs
  6. Use Pixymon train function to train color signature
  7. Assess detection of color signature
  8. Repeat procedure for light environment measuring > 250 cd

- LED trajectory tracking:
  1. Complete LED color signature detection test procedure
  2. Connect Pixy camera to the Raspberry Pi and
  3. Perform gesture with glove in front of camera
  4. Take gesture data from Raspberry Pi and copy it to MATLAB HMM simulation
  5. Ensure that the gesture plotted by MATLAB matches the gesture performed

3) Raspberry Pi

- Evaluate computation time:
  1. Connect oscilloscope lead to data-received flag pin of Raspberry Pi
  2. Connect oscilloscope lead to MIDI-signal-sent flag pin of Raspberry Pi
  3. Execute test script for computation time
  4. Record when trajectory data is received from Pixy
  5. Record when MIDI signal is sent from Raspberry Pi
  6. Calculate time difference
- Analyze MIDI signal output
  1. Connect oscilloscope to MIDI signal output pin
  2. Send MIDI command
  3. Evaluate MIDI signal voltage, baud rate, and data bits
  4. Compare recorded experimental signal with theoretical signal

4) HMM Algorithm

- Gesture recognition rate:
  1. Connect oscilloscope lead to MIDI signal output pin
  2. Perform gesture one
  3. Compare captured signal to theoretical signal
  4. Repeat steps 2 and 3 for 50 iterations
  5. Repeat steps 2-4 for gestures two and three
  6. Evaluate if 75% gesture recognition rate is met
- Latency:
  1. Connect oscilloscope lead to data-received flag pin of Raspberry Pi
  2. Connect oscilloscope lead to MIDI signal output TTL pin
  3. Perform gesture one
  4. Evaluate time difference between data input and output
  5. Repeat steps 3 and 4 for gestures two and three
  6. Evaluate if 160 ms latency is met

5) Mixxx Software

- Confirm effect execution
  1. Complete Analyze MIDI signal output test procedures from Raspberry Pi
  2. With captured MIDI signal, validate mapping of specified signal to audio effect
  3. Observe audio effect applied and assess whether audio effect is correct

- Evaluate processing time
  4. Complete Confirm effect execution test procedures
  5. Record latency of effect execution through Mixxx
  6. Assess whether latency is in the range of 36-64 milliseconds [19]

J. *Acquired Gestures*

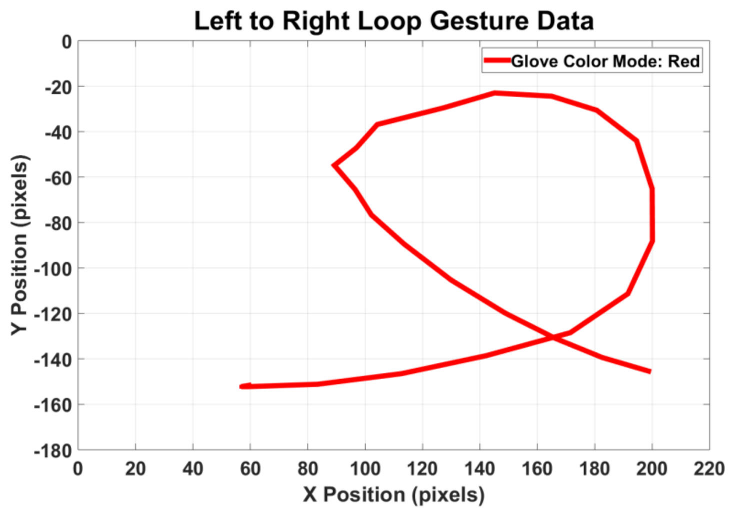


Fig. 26. Acquired loop gesture

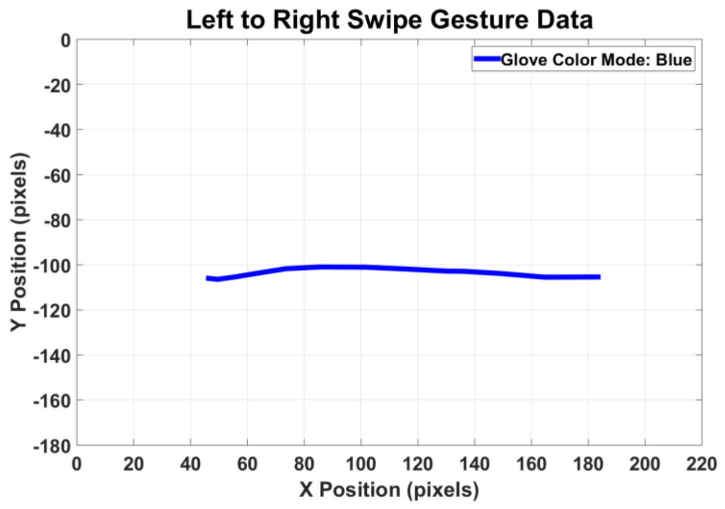


Fig. 27. Acquired swipe gesture