5-1-2018

GLOVELET: A WEARABLE 3D AND 2D TRACKING GLOVE

Prasoon Gautam

GLOVELET: A WEARABLE 3D AND 2D TRACKING GLOVE

by

PRASOON GAUTAM

Presented to the Faculty of the Honors College of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

HONORS BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2018

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my supervising professor, Dr. Christopher Conly, for his continuous support and guidance of my Senior Design project, and for his patience, advice, and immense knowledge. I could not have imagined having a better advisor and mentor for the project.

Besides my advisor, I would like to thank my fellow Glovelet team members, Arnav Garg, Joseph Tompkins, Sushil Bista, and Ravindra Javadekar, who have helped me with my work on the wearable glove. I am also thankful to my parents for supporting me through my undergraduate studies.

Further, I thank the friends and faculty at the Honors College for having worked with me through the past four years.

Finally, I would like also like to thank all my friends for their constant support and motivation. They made feel like home even when I was thousands of mile away.

May 04, 2018

ABSTRACT

GLOVELET: A WEARABLE 3D AND 2D TRACKING GLOVE

Prasoon Gautam, B.S. Computer Science

The University of Texas at Arlington, 2018

Faculty Mentor: Christopher Conly

The traditional mouse as a user-interactive device has been around for quite some time but can be a hindrance when used to control three dimensional devices. Through Glovelet, my team and I seek to develop a wearable glove that can replace the traditional mouse with a more user-interactive and easy-to-use device. This would greatly enhance the user experience and would allow users to employ their hands to move objects, both two-dimensional and three-dimensional, on the computer screen, thereby making the experience more intuitive for younger and older adults. Glovelet will be designed from the beginning to be light-weight, comfortable, and able to be worn for long periods of time without any discomfort. Users will be able to freely switch between Glovelet and typing on a keyboard or even switching back to a mouse if they wish.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

1.1 Background

Glovelet is a wearable tracking glove that can replace the traditional mouse with a more user-interactive and easy-to-use device. The device would greatly enhance the user experience and would allow users to use their hands to move files and folders, both two-dimensional and three-dimensional, on the computer screen, thereby making the experience more intuitive for younger and older adults. The device provides natural-feeling gestures performed by your hand(s). The movement or rotation of the tracker for a three-dimensional object in the virtual environment is a one-to-one movement with the glove.

The task was accomplished by using a three-dimensional data collection from an inertial measurement unit (IMU) mounted at the back of the hand, as well as four flex sensors to detect finger motions of the thumb, index, and middle fingers. We utilized proven algorithms to reduce noise and recognize gestures, such as clicking or scrolling, or even manipulating virtual three-dimensional objects.

1.2 Motivation for Glovelet

The traditional mouse, which is widely used, is a comfortable choice but is less user-interactive and intuitive. It does the basic job for a user-interface device but faces a problem when used to control three-dimensional objects. The left and right gestures may feel like normal gestures but are not natural. Due to ongoing changes in gaming and design industries involving three-dimensional objects and rotations in real-time, the traditional

mouse lacks the natural movement of the hand and thus starts underperforming. There are a lot of wearable devices already available in the market that complete the job of a user interface device but are bulky and expensive. The devices also have limited gesture options and difficult to expand on them.

Glovelet is a more advanced and comfortable version of the existing wearable technologies. The device will be worn on the hand and will involve movements of thumb, index and the middle fingers, which will make wearable technology better for the user. Minimal computation will be used on board the actual glove unit. The majority of processing will be accomplished by the computer running the front-end application. The data between the glove and the computer application will be done via the Bluetooth wireless connection.

Users will be able to control user interface normally performed by a mouse, as well as provide new convenient use cases not previously possible with standard user interface devices. Glovelet is designed from the beginning to be lightweight, comfortable, and able to be worn for long periods of time without creating any discomfort. Users will be able to switch freely between using Glovelet and typing on a keyboard or even switching back to a computer mouse. This device is intended for any user who uses computer technology on a regular basis and wishes to increase productivity and workflow, as well as provide a foundation for a unique gaming interface device. The glove was designed to be affordable, useful, and usable for a wide range of applications.

1.3 Related Work

There are quite a few working models or devices already available in the market. One is Ultimate Mouse Control: This is a global script that lets you control your mouse

through the use of Myo device. The Myo armband is a wearable gesture control and motion control device that lets you take control of your phone, computer, and many other devices. The device is worn on the forearm and reads the muscle movements by sensing electrical activity along with other sensors to recognize gestures. [1]

Another is SignAloud: It is a pair of gloves that can recognize hand gestures that correspond to words and phrases in American Sign Language. Each glove contains sensors that record hand position and movement. The central computer matches the gesture data from the glove with the data in the system which, if successful, prompts the associated word or phrase to be spoken through a speaker. [2]

1.4 Flex Sensors

1.4.1 Resistance

The electrical resistance or just resistance of an electrical conductor is a measure of the difficulty of passing an electric current through that conductor. The SI unit of electrical resistance is the ohm (Ω). When using the hydraulic analogy, the current flowing through a wire is like water flowing through a pipe, and the resistance is proportional to how much pressure is required to achieve a given flow. A piece of conducting material of a particular resistance meant for use in a circuit is called a resistor.

1.4.2 Flex Sensor

The Flex Sensor is a unique component that changes resistance when bent. An unflexed sensor has a nominal resistance of 10,000 ohms (10 K). As the flex sensor is bent the resistance gradually increases. When the sensor is bent at 90 degrees, its resistance will range between 30-40 K ohms. [3]

Figure 1.1: A Flex Sensor [3]

1.4.3 How Do They Work?

The flex sensor perceives changes in bend angle as a linear proportional change in current or voltage. Different flex angles cause the area of the resistor to differ from its un-flexed 10K value, resulting in this characteristic. The bending of the sensor is registered as a change in resistance only when the sensor is flexed in one direction. The output analog data is converted to digital data by the A/D conversion module of the microcontroller (Arduino Feather).

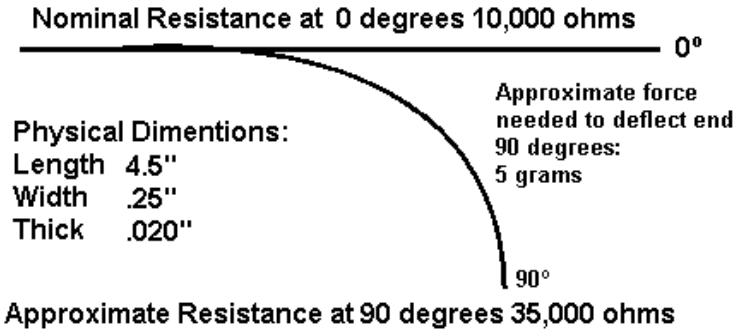

Figure 1.2: Flex Sensor Resistance [3]

The flex sensor is ideal for any project where precise measurement of the motion of a particular component is required. It is placed across the area on which flex is to be measured.

Figure 1.3: Bent Finger with a Flex Sensor [3]

CHAPTER 2

METHOD

2.1 Architecture

The high-level structure of the Glovelet software system is divided into four layers: The Hardware/Sensor Layer, the Computer Vision Layer, Event API Layer and Application Layer. The Hardware and Sensor Layer is the physical layer which will provide us with the IMU data and Flex Sensor data via the USB serial connection. The Computer Vision Layer is responsible for providing us with the user's hand tracking coordinates. The data from the Hardware and Sensor Layer and the Computer Vision Layer is processed in the Event API Layer where the Kalman Filter Algorithm is applied to compute the user's hand gestures and track their hand movement.

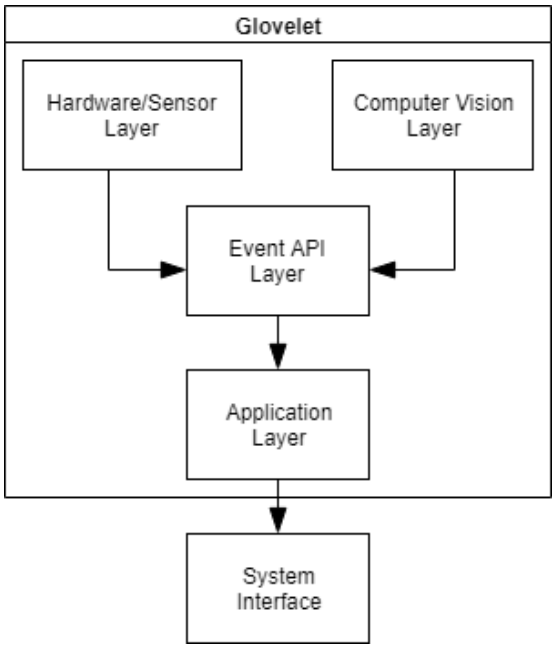

Figure 2.1: System Architecture Overview

2.1.1 Hardware/Sensor Layer

GloveLet uses two types of sensors to capture user hand gesture and motion data. For finger gestures, flex sensors are used to capture the degree of bend in each of the fingers. For rotational data, an IMU (Inertial Measurement Unit) with accelerometer, magnetometer, and gyroscope and on-board AHRS (Attitude and Heading Reference System) is used.

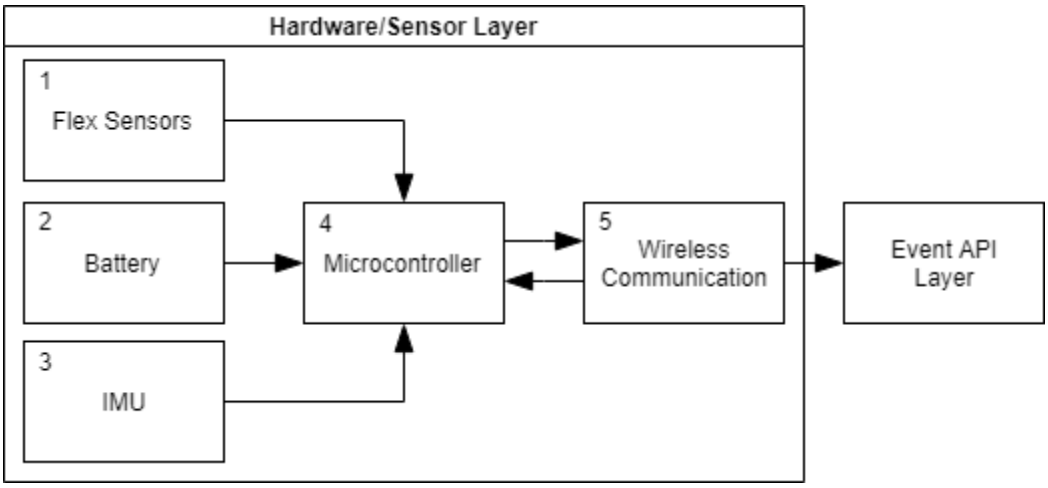

Figure 2.2: Hardware/Sensor Layer Architecture

2.1.1.1 Flex Sensors

GloveLet uses a total of four flex sensors to capture data regarding the degree of bend of each finger. The index and middle fingers each have a 4-inch flex sensor. Two 2-inch flex sensors are attached to the thumb, one on the outer thumb and one on the inner thumb. This is due to the opposable nature of the thumb, which has a higher degree of freedom in bend motion. An analog signal is received from each of the flex sensors, and the microcontroller then executes an algorithm to clamp and normalize the readings before outputting each data sample. Flex sensor values read from the microcontroller are thus

floating point numbers between zero and one, where zero is no or minimal bend and one is full or maximum bend.

For use as a mouse-type interface device, thresholds for bends are set and control flow statements are used to determine which action is being performed. For example, if the index finger exceeds a predetermined threshold, this is considered equivalent to the left mouse button being pressed down and the left-button down state is entered. Once the bend of the index finger decreases below a second predetermined threshold, this is considered equivalent to releasing the mouse button, and the left-button down state is exited.

As a part of the group, I was asked to implement the flex sensors into the glove with the onboard Arduino to get the data from the sensors when they are bent. I developed a code for the Arduino which takes in the resistance from the flex sensors and outputting the result as time series data for the thumb, index, and the middle finger. The graph obtained gave a resistance increasing and decreasing as the fingers are flexed or relaxed (see Appendix A).

2.1.2 Computer Vision Layer

The Computer Vision layer handles raw video stream data and performs hand tracking algorithms. Coordinates of the hand relative to the area of the captured video are its outputs. The layer uses computer vision to track the position of the hand relative to the camera frame. It uses a color-based tracking algorithm to track the position of the hand. The tracked color is then segregated from the background.

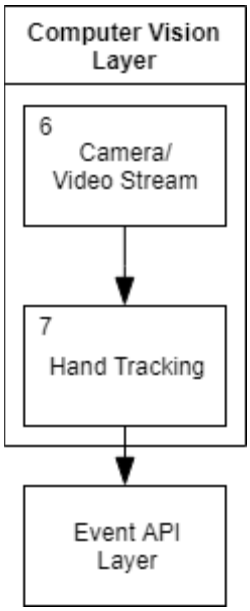

Figure 2.3: Computer Vision Architecture

2.1.3 Event API Layer

The Event API layer will be responsible for all pre-processing of the data it receives from the Hardware/Sensor layer and Computer Vision layer. In addition, it will need to establish connections with the Hardware/Sensor layer components as well as the Computer Vision layer components. This layer is where any filters, parsing, and structuring will be applied to the data (i.e. low-pass filter). From here, the pre-processed data will be provided to the Application layer via a forward-facing API, which will function similar to mouse or keyboard input event APIs.

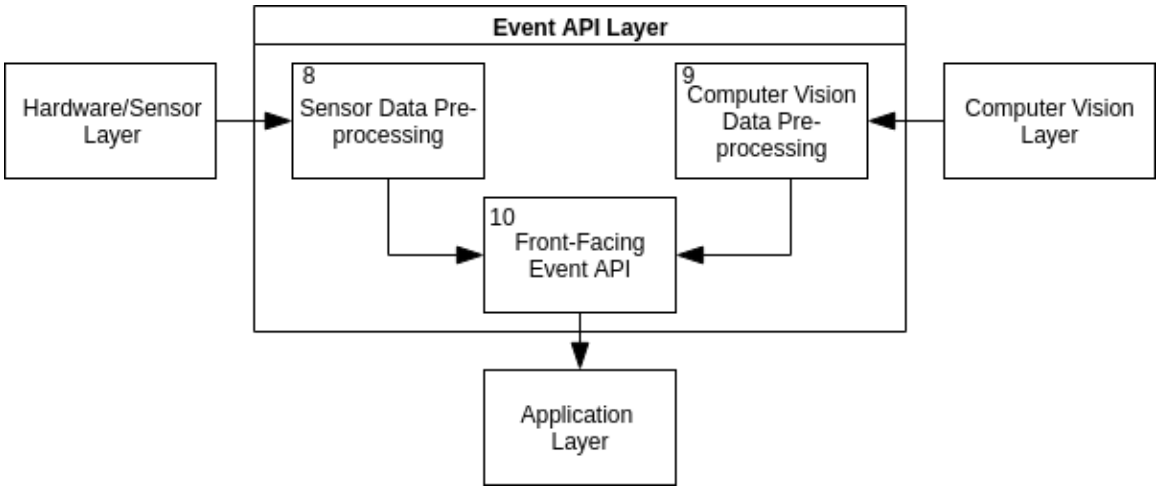

Figure 2.4: Event API Layer

2.1.4 Application Layer

The Application layer will be responsible for observing and handling events issued by the Event API layer, as well as gesture-recognition logic processing, and the software application runtimes executed by the host system. Components include event handling, gesture-recognition logic, and applications. Applications may consist of anything from drivers to GUIs, which in-turn, interface with the host system.

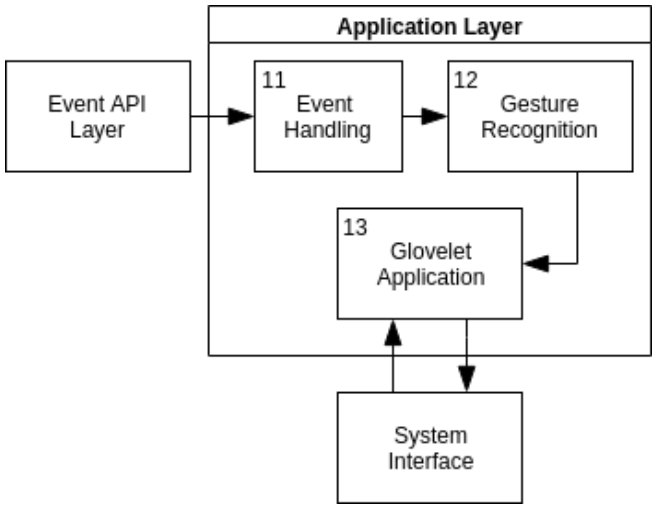

Figure 2.5: Application Layer

2.2 Data Integration

The integration of computer vision and sensor data was one of the primary challenges of this project. Data pre-processing methods for CV and sensor data were developed separately as submodules. For this reason, an event-based data flow method was designed and implemented. The concept of this system was derived from commonly used mouse event APIs, where applications implement their own listener interface which receives mouse event information via callback functions.

To increase the efficiency and performance of computation of both CV and sensor data, each is done on a separate process so that computation is done in parallel on the host device. This means that GloveLet applications have a minimum of three processes running in parallel at a given time. Using multiprocessing techniques, the CV and sensor processes dispatch events in a constant stream, which are then distributed to listeners that execute a callback subprocess on the main program process.

Once the GloveLet application starts, the CV and sensor sub-processes begin pre-processing data and dispatching events in parallel with the main application process. The application will then enter the main loop, where on each execution of the loop the listeners will respond to any dispatched events. Within the listener, callbacks are where mouse functionality or any other functionality is implemented.

CHAPTER 3

DISCUSSION

After the completing the project, the designed glove met our expectations. We learned a lot of skills, for example sewing (to place the flex sensors).The image below has the detailed picture of the glove with all the sensors.

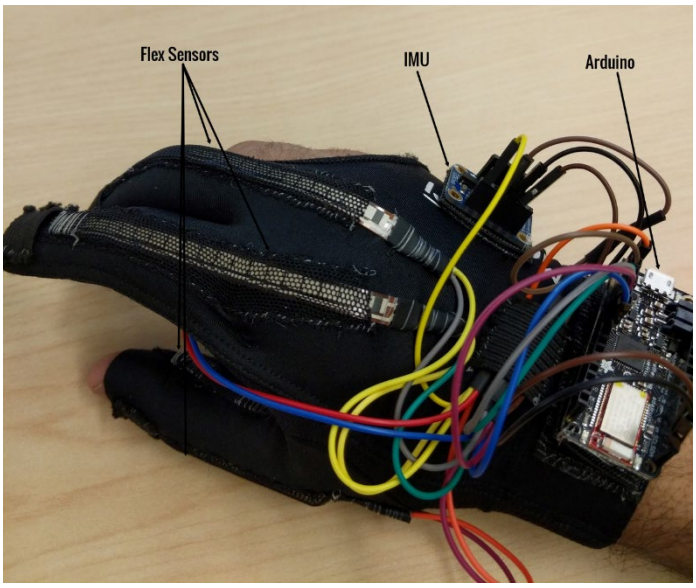

Figure 3.1: Completed Glove

During the course of the two semesters on the project, I faced a lot of difficulties, as there were a lot of factors and components involved in the project that I had no clue about and the project had a massive learning curve. I had to learn about new concepts and techniques to successfully work on the project. As a computer science student with no background in electronics, I had to learn about flex sensors and how they work. Learning to use them and then creating programs to use the data from the sensors took several weeks.

We also faced a lot of problems during the course of the project. Everyone had to learn something new as we lacked the knowledge to successfully work on the project. The learning phase took a lot of time. I believe if we had more knowledge about the concepts involved, we would have added more functionalities to the device. As of right now, we have left click, right click, scroll up and scroll down functionalities.

CHAPTER 4

CONCLUSION

Glovelet is designed to be a wireless two-dimensional and three-dimensional object tracking glove. The completed device was a success and performed all the basic functions. The rotation on the glove needed minimum computation. The device included several layers in its architecture. These layers comprised of Hardware/Sensor layer, Computer vision layer, Event API layer, and Application layer. The position of the hand in the three-dimensional space is derived through multiple sensors installed on the glove while the hand movement is tracked, using a camera which involves the computer vision layer, relative to the camera frame. The most difficult part of the project was the integration as all the subsystems were developed separately so fusing all the systems to make them work in sync took a lot of effort.

APPENDIX A

SIGNALVIEWERUTIL.PY

```
import numpy as np
import matplotlib.pyplot as plt
from glovelet.sensorapi.sensorstream import SensorStream
from glovelet.sensorapi.glovelet_sensormonitor import
GloveletBNO055IMUSensorMonitor, GloveletFlexSensorMonitor
from glovelet.utility.timeseries import DataSequence
from multiprocessing import Process, Queue
from serial.serialutil import SerialException
from scipy.signal import butter, filtfilt


np.set_printoptions(precision=4, suppress=True)


_PORT = '/dev/ttyACM0'
_BAUD = 115200


class ImuEvent:
    def __init__(self, velocity, acceleration, rotation, timestamp,
tdelta):
        self.velocity = velocity
        self.acceleration = acceleration
        self.quaternion = rotation
        self.timestamp = timestamp
        self.tdelta = tdelta


class ImuPlotEvent:
    def __init__(self, acc_timeseries, vel_timeseries, rot_timeseries,
dt_seq, t_seq):
        self.t_seq = np.flip(t_seq[:], 0)
        self.accel_x = np.flip(acc_timeseries[:, 0], 0)
        self.accel_y = np.flip(acc_timeseries[:, 1], 0)
        self.accel_z = np.flip(acc_timeseries[:, 2], 0)
        self.accel_max = max([max(acc_timeseries[:, i]) for i in
range(3)])
        self.accel_min = min([min(acc_timeseries[:, i]) for i in
range(3)])
        self.vel_x = np.flip(vel_timeseries[:, 0], 0)
        self.vel_y = np.flip(vel_timeseries[:, 1], 0)
        self.vel_z = np.flip(vel_timeseries[:, 2], 0)
        self.vel_max = max([max(vel_timeseries[:, i]) for i in
range(3)])
        self.vel_min = min([min(vel_timeseries[:, i]) for i in
range(3)])
        self.orient_x = np.flip(rot_timeseries[:, 0], 0)
        self.orient_y = np.flip(rot_timeseries[:, 1], 0)
```

```
        self.orient_z = np.flip(rot_timeseries[:, 2], 0)
        self.orient_w = np.flip(rot_timeseries[:, 3], 0)


class FlexPlotEvent:
    def __init__(self, flex_data, t_elapsed):
        self.index = np.flip(flex_data[:, 0], 0)
        self.middle = np.flip(flex_data[:, 1], 0)
        self.thumb0 = np.flip(flex_data[:, 2], 0)
        self.thumb1 = np.flip(flex_data[:, 3], 0)
        self.t_elapsed = np.flip(t_elapsed[:], 0)


def imu_stream(stream, monitor, q):
    try:
        stream.open()
        while stream.is_open():
            stream.update()
            imu_pltevnt = ImuPlotEvent(monitor.acceleration_sequence,
                                       monitor.velocity_sequence,

monitor.orientation_timeseries,
                                       monitor.accel_tdelta,
monitor.accel_time_elapsed)
            if q.full():
                q.get()
            q.put(imu_pltevnt)
            # print(' '.join([str(x) for x in
monitor.orientation_timeseries[0]]))
    except SerialException as e:
        print(e)
        stream.close()


def flex_stream(stream, monitor, q):
    try:
        stream.open()
        while stream.is_open():
            stream.update()
            flex_pltevnt = FlexPlotEvent(monitor.data_sequence,
monitor.time_elapsed)
            if q.full():
                q.get()
            q.put(flex_pltevnt)
    except SerialException as e:
        print(e)
    finally:
        stream.close()
```

```
def plot_imu_data(nsamples=None):
    if nsamples is None:
        nsamples = 500000
    s = SensorStream(_PORT, _BAUD, do_success_check=False,
conn_delay=500)
    m = GloveletBNO055IMUSensorMonitor(100)
    s.register_monitor(m)
    pltevntqueue = Queue(5)
    stream_proc = Process(target=imu_stream, args=(s, m, pltevntqueue))
    stream_proc.start()
    plt.ion()
    fig = plt.figure(figsize=(8, 10))
    shape_orient = m.orientation_timeseries.shape
    acc_ax = fig.add_subplot(311)
    vel_ax = fig.add_subplot(312)
    quat_ax = fig.add_subplot(313)
    while pltevntqueue.empty():
        continue
    pltevnt = pltevntqueue.get()
    t_seq = pltevnt.t_seq
    line1, = acc_ax.plot(t_seq, pltevnt.accel_x, 'r-', label='x')
    line2, = acc_ax.plot(t_seq, pltevnt.accel_y, 'g-', label='y')
    line3, = acc_ax.plot(t_seq, pltevnt.accel_z, 'b-', label='z')
    vel_x, = vel_ax.plot(t_seq[:], pltevnt.vel_x, 'r-', label='x')
    vel_y, = vel_ax.plot(t_seq[:], pltevnt.vel_y, 'g-', label='y')
    vel_z, = vel_ax.plot(t_seq[:], pltevnt.vel_z, 'b-', label='z')
    t_seq_quat = t_seq[:shape_orient[0]]
    quat_line1, quat_line2,\
    quat_line3, quat_line4 = quat_ax.plot(t_seq_quat, pltevnt.orient_w,
'k-',
                                          t_seq_quat, pltevnt.orient_x, 'c-
',
                                          t_seq_quat, pltevnt.orient_y, 'm-
',
                                          t_seq_quat, pltevnt.orient_z, 'y-
')
    quat_ax.set_ylim(-1, 1)
    plt.show()
    for i in range(nsamples):
        pltevnt = pltevntqueue.get()
        if isinstance(pltevnt, ImuPlotEvent):
            t_seq = pltevnt.t_seq
            line1.set_data(t_seq, pltevnt.accel_x)
            line2.set_data(t_seq, pltevnt.accel_y)
            line3.set_data(t_seq, pltevnt.accel_z)
            acc_ax.set_ylim((min(pltevnt.accel_min, -5),
max(pltevnt.accel_max, 5)))
            acc_ax.set_xlim((min(t_seq), max(t_seq)))
```

```
            vel_x.set_data(t_seq[:], pltevnt.vel_x)
            vel_y.set_data(t_seq[:], pltevnt.vel_y)
            vel_z.set_data(t_seq[:], pltevnt.vel_z)
            vel_ax.set_ylim((min(pltevnt.vel_min, -2),
max(pltevnt.vel_max, 2)))
            vel_ax.set_xlim((min(t_seq[:]), max(t_seq[:])))
            t_seq_quat = t_seq[:shape_orient[0]]
            quat_line1.set_data(t_seq_quat, pltevnt.orient_w)
            quat_line2.set_data(t_seq_quat, pltevnt.orient_x)
            quat_line3.set_data(t_seq_quat, pltevnt.orient_y)
            quat_line4.set_data(t_seq_quat, pltevnt.orient_z)
            quat_ax.set_xlim((min(t_seq_quat), max(t_seq_quat)))
            fig.canvas.draw()


def plot_flex_data(nsamples=None):
    if nsamples is None:
        nsamples = 500000
    s = SensorStream(_PORT, _BAUD, do_success_check=False,
conn_delay=500)
    m = GloveletFlexSensorMonitor(100)
    s.register_monitor(m)
    pltevntqueue = Queue(5)
    stream_proc = Process(target=flex_stream, args=(s, m,
pltevntqueue))
    stream_proc.start()
    plt.ion()
    fig = plt.figure(figsize=(8, 10))
    flex_plt = fig.add_subplot(111)
    while pltevntqueue.empty():
        continue
    pltevnt = pltevntqueue.get()
    t_seq = pltevnt.t_elapsed
    line1, = flex_plt.plot(t_seq, pltevnt.index, 'r-', label='index')
    line2, = flex_plt.plot(t_seq, pltevnt.middle, 'g-', label='middle')
    line3, = flex_plt.plot(t_seq, pltevnt.thumb0, 'b-', label='thumb0')
    line4, = flex_plt.plot(t_seq, pltevnt.thumb1, 'y-', label='thumb1')
    plt.show()
    for i in range(nsamples):
        pltevnt = pltevntqueue.get()
        if isinstance(pltevnt, FlexPlotEvent):
            t_seq = pltevnt.t_elapsed
            line1.set_data(t_seq, pltevnt.index)
            line2.set_data(t_seq, pltevnt.middle)
            line3.set_data(t_seq, pltevnt.thumb0)
            line4.set_data(t_seq, pltevnt.thumb1)
            flex_plt.set_ylim(-0.25, 1.25)
            flex_plt.set_xlim((min(t_seq), max(t_seq)))
```

```
            fig.canvas.draw()


def main():
    # plot_flex_data()
    plot_imu_data()


if __name__ == '__main__':
    main()
```

REFERENCES

[1] "Ultimate Mouse Control." *Myo Market*, www.market.myo.com/app/557af681e4b0910e1dcf8482/ultimate-mouse-control.

[2] "*UW undergraduate team wins $10,000 Lemelson-MIT Student Prize for gloves that translate signlanguage*." *UW News*, 12 Apr. 2016, www.washington.edu/news/2016/04/12/uw-undergraduate-team-wins-10000-lemelson-mit-student-prize-for-gloves-that-translate-sign-language/.

[3] Mah, Michael, et al. "Utilizing Flex Sensors within a design." *Ualberta*, http://www.ece.ualberta.ca/~elliott/ee552/studentAppNotes/2003_w/interfacing/ADC0809_flexsensors/Flex_sensor.htm. Accessed 3 May 2018

BIOGRAPHICAL INFORMATION

Prasoon Gautam graduated from the University of Texas at Arlington in May 2018 with an Honors Bachelor of Science in Computer Science with Latin Honors. He also has a minor in Mathematics. He has completed an Unmanned Vehicle System certificate from the University of Texas at Arlington. He is a member of the Tau Beta Pi Engineering Honors Society and also Upsilon Pi Epsilon. Prasoon plans to continue his education and go to graduate school to get a Master's degree in Computer Science with a focus in Robotics and Artificial Intelligence.