Smart Pen: Final Project for ECE5725

Author: Ziqi Yang (zy259), Haowen Tao (ht398)

Introduction

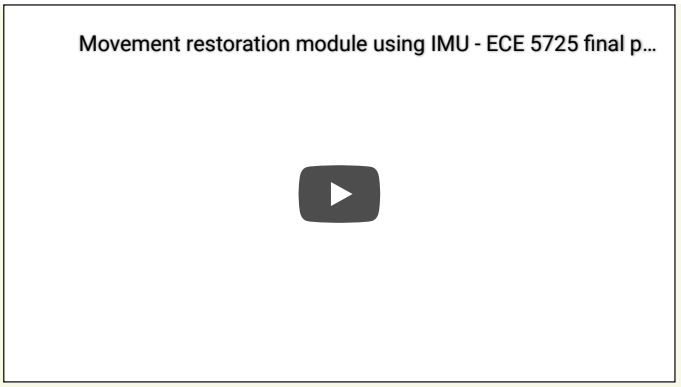
Nowadays, in order to gain the handwriting results, people often rely on the capacitive touch screen, stylus pen, or other similar devices. They are functional and precise, but not always flexible. Our project combines Raspberry Pi with IMU to build a product that can keep tracking of the movement and generate the trajectory very quickly. We designed a system that does not need any specific surface or other input tools for support and it only reply on IMU and its sensors. By connecting IMU with Raspberry Pi, when IMU device moves, the sensor data will be transmitted from IMU to Raspberry Pi. Then using our program and algorithm, restore the IMU movement. The movement will be recorded and stored in Raspberry Pi. With PyGame, the movement also can be displayed on TFT screen of Raspberry Pi. With a small and convenient button, the recording function can start and finish very flexibly. Users also can attach our device on other things to track their movements, too. Therefore, our device provides a workable solution for handwriting tracking, movement tracking and etc.

System Setup

Objective

The goal of the project is to design a separate module that can be put on a pen, robot or even people to track and record the object movement. Users can use this module and attach it with their other devices, so when the object is moved, our device can restore and track their trajectory and display the movement in the horizontal plane in the world frame on the screen. A very classic application is that users can use it as a pen, so their writings will be recorded and saved as an image. A special feature about this project is that the tracking does not reply on any specific plane and it can be applied on every plane, even in the air. The placement or tilt of the device also will not affect the final result.

We use Raspberry Pi and Inertial Measurement Unit to be the main components for our project. Also, PyGame is used to display the trajectory on Raspberry Pi.



Smart Pen: Final Project for ECE5725

Author: Ziqi Yang (zy259), Haowen Tao (ht398)

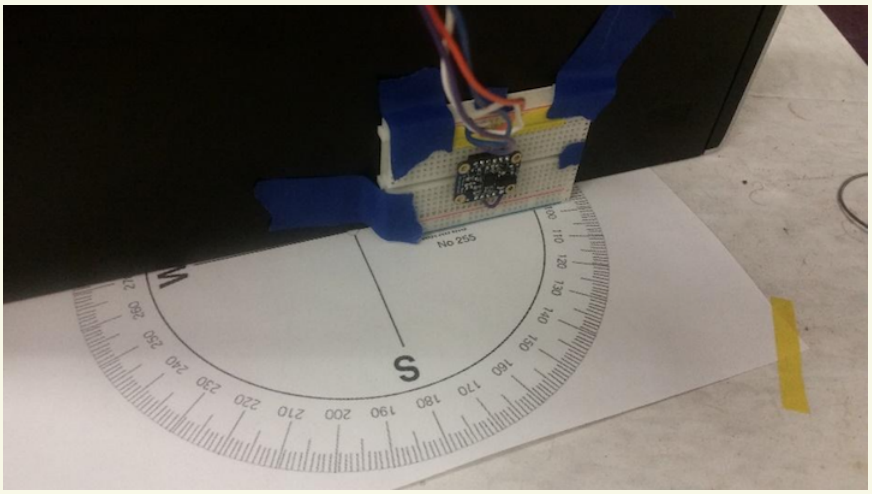
IMU Calibration

Due to fabrication inaccuracy, the 3 axes of the accelerometers and the 3 axes gyroscopes are usually misaligned, causing an error between the Euler Angle of two coordinates. Calibration is needed. Using the 6-pos calibration technique, we setup the 6 calibration position as follows:

	Axes orientation			Gravity Components		
Pos	X	Y	Z	X/g	Y/g	Z/g
1	east	sky	south	0	-1	0
2	east	north	sky	0	0	-1
3	ground	east	south	1	0	0
4	west	ground	south	0	1	0
5	sky	west	south	-1	0	0
6	south	west	ground	0	0	1

Calibration Position and Gravity Components

Using a self-made calibration table (Figure XX) we fix the imu to the 6 positions and record the reading, the calibration data gathered by these six positions are:



Self-made Calibration Station

Pos	X axis acceleration/g	Y axis acceleration/g	Z axis acceleration/g
1	0.396139678	9.717861488	0.059217435
2	0.127414167	0.0800275	9.8051675
3	-9.814006667	0.138280833	0.320044167
4	-0.365053333	-9.618991667	-0.074034167
5	9.756785833	-0.229565833	-0.349639167
6	-0.110838333	-0.223583333	-9.869680833

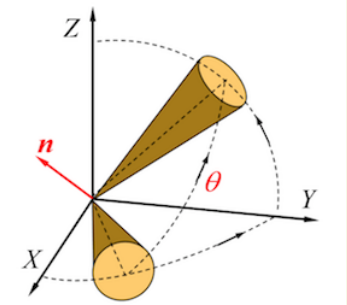
Calibration Data

The acceleration data after calibration is (Ax, Ay, Az are data after calibration and ax, ay, az are raw data):

$$\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} = \begin{bmatrix} 0.00121521 \\ -0.00598759 \\ -0.00113234 \end{bmatrix} + \begin{bmatrix} -0.99796807 & -0.03881531 & -0.01214914 \\ 0.0187575 & -0.98603886 & -0.01548194 \\ 0.03414898 & -0.00679486 & -1.00327416 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} -0.00413306 & 0. & 0. \\ 0. & 0.01102923 & 0. \\ 0. & 0. & -0.00413306 \end{bmatrix} \begin{bmatrix} {a_x}^2 \\ {a_y}^2 \\ {a_z}^2 \end{bmatrix}$$

Sensor Fusion For Raw Data

Quaternion calculation:



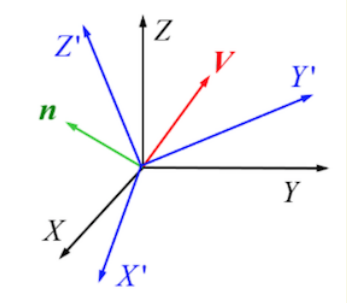
Quaternion

Quaternion is a representation of the orientation and rotation of the object, and it is easier to compute the rotation of vector than the Euler Angles. The transaction between the quaternion and the Euler Angle is shown below:

$$\mathbf{q} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\cos\alpha \cdot \mathbf{i} + \sin\frac{\theta}{2}\cos\frac{\beta}{2} \cdot \mathbf{j} + \sin\frac{\theta}{2}\cos\gamma \cdot \mathbf{k}$$

$$\mathbf{q} = \lambda + P_1\mathbf{i} + P_2\mathbf{j} + P_3\mathbf{k}$$

Quaternion rotation:



Frame Rotation transform Using Quaternion

For a fixed vector V coordinated in frame XYZ, it could be represented in quaternion:

$$V = 0 + V_x i + V_y j + V_z k$$

If the frame rotates for q, become X'Y'Z', V coordinated in X'Y'Z' could be represented as:

$$V' = 0 + V_x' i' + V_y' j' + V_z' k'$$

Then,

$$V = q \circ V' \circ q^{-1}$$

However, using the raw data generated by the gyroscope is still not enough, because of the misalignment mentioned in calibration part, the gravity vector obtained by the accelerometer need to be considered. The function 'UpdateIMU' is used

to calculate the error between the gravity vector calculated by gyroscope and the gravity vector measured by the accelerometer. This function was run 2000 times at the beginning stationary stage with no movements nor rotations, and using a feedback to calculate the error. The function is shown below:

```
def UpdateIMU(self, Gyr, Acc):
        if np.linalg.norm(Acc) == 0:
            warnings.warn("Accelerometer magnitude is zero.  Algorithm update aborted.")
            return
        else:
            Acc = np.array(Acc / np.linalg.norm(Acc))
        v = np.array([[2*(self.q[1]*self.q[3] - self.q[0]*self.q[2])],
                [2*(self.q[0]*self.q[1] + self.q[2]*self.q[3])],
                [self.q[0]**2 - self.q[1]**2 - self.q[2]**2 + self.q[3]**2]])
```

Variable “Acc” and ‘v’ are both normalized gravity vector calculated by the accelerometer and the gyroscope, then the angular deviation (error) between the two vector could be represented using their cross product:

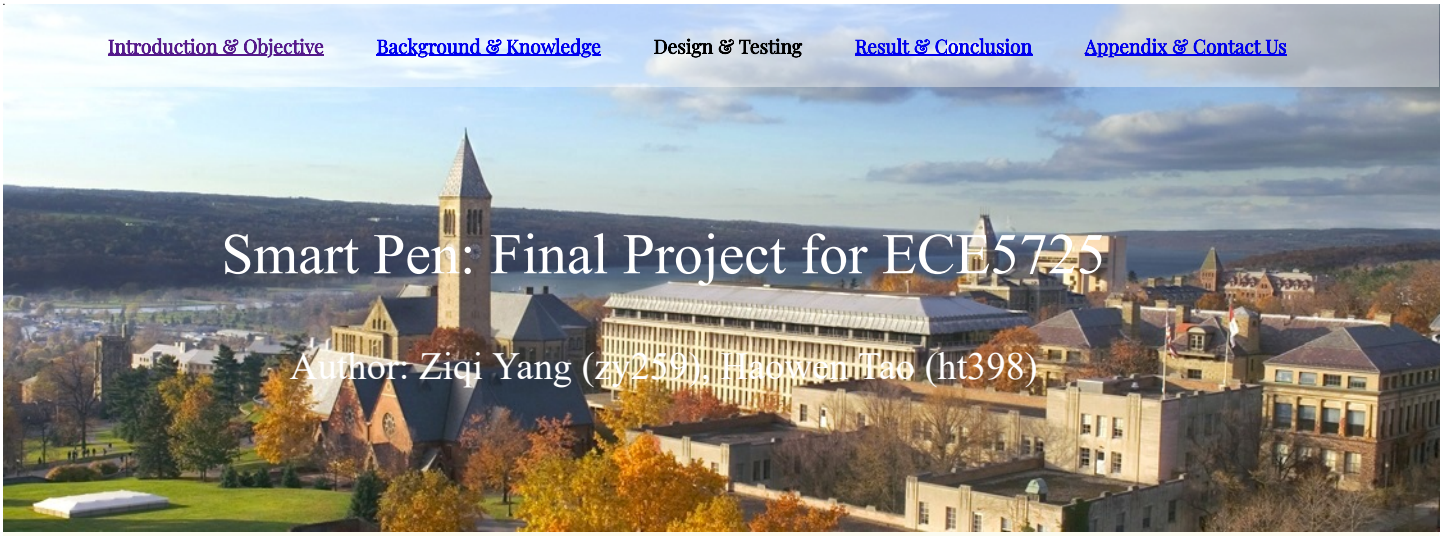
```
error = np.cross(v,np.transpose([Acc]),axis = 0)
```

The calculated error could also be integrated to update the gyroscope reading using PI negative feedback loop (that’s why we need to run this function 2000 times, so the PI loop could converge):

```
self.IntError = self.IntError + error
Ref = Gyr - np.transpose(self.Kp*error+self.Ki*self.IntError)
```

The quaternion could then be calculated by the corrected gyroscope data:

```
pDot = np.multiply(0.5 , self.quaternProd_single(self.q, [0, Ref[0,0], Ref[0,1], Ref[0,2]]))
self.q = self.q + pDot * self.SamplePeriod;
self.q = self.q / np.linalg.norm(self.q);
self.Quaternion = self.quaternConj(self.q);
```

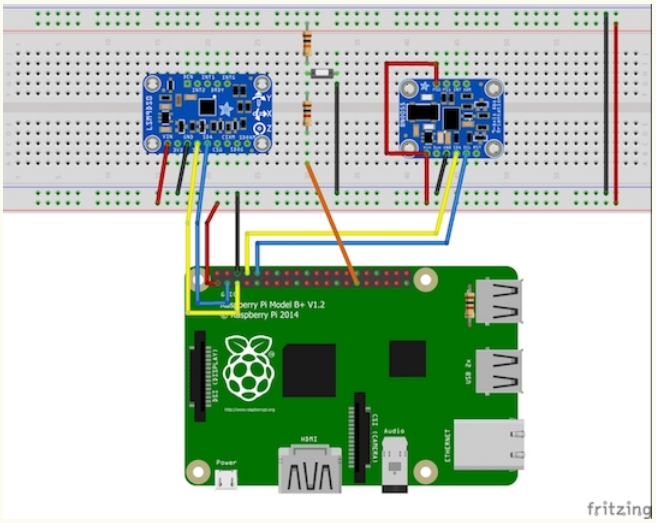


Smart Pen: Final Project for ECE5725

Author: Ziqi Yang (zy259), Haowen Tao (ht398)

Overall Setup

Inertial measurement unit (IMU) on the module is used to gather the acceleration and rotation information of the moving object. Two different devices were used: LSM9DS0 and BNO055 the hardware connection is shown below (I2C is used for the connection for LSM9DS0 and UART is used for the connection for BNO055).

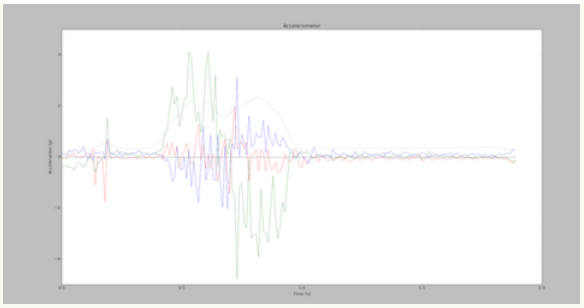


Device Setup

LSM9DS0 is a 9 DOF IMU with could provide the raw accelerometer, gyroscope and magnetometer data with the scale set to +_ 2g for acceleration, +- 245 dps for angular rate and gyroscope disabled. And BNO055 is an absolute orientation sensor with an onboard microcontroller doing the sensor fusion for the IMU raw data output, and could generate the acceleration data without the gravity component as well as the quaternion of the IMU with respect to the world frame.

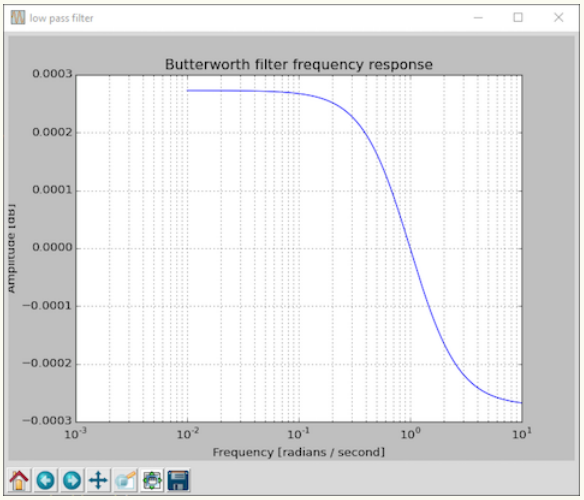
IMU Movement Restore

To restore the movement data, we need to do the integration for the acceleration data. However, as can be seen in the figure, there are a significant amount of noises on the raw data and needed to be removed.

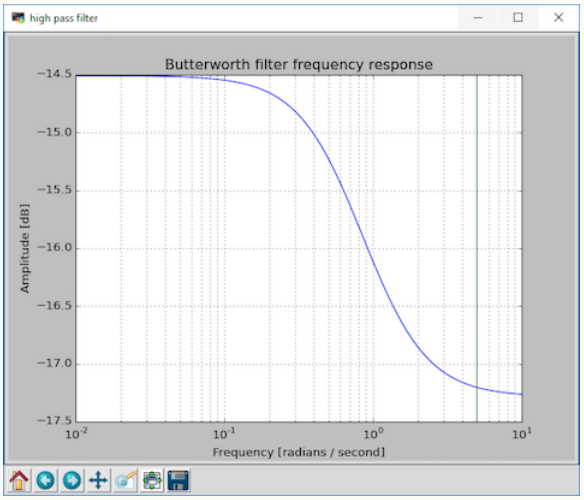


Raw Acceleration Data (Click for more details)

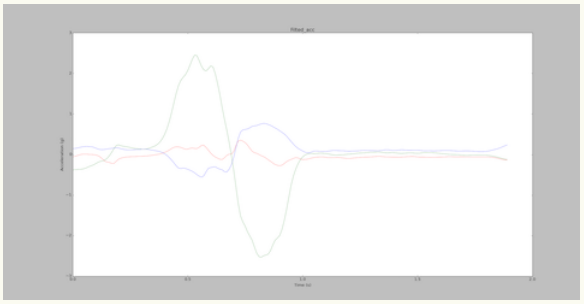
A low-pass butterworth filter and a high pass butterworth filter is applied to the acceleration data, the low-pass filter is to remove the spikes, and the high pass filter is used to remove the noises. The filters and acceleration data after the filter are shown below:



Low-pass Butterworth Filter (Click for more details)

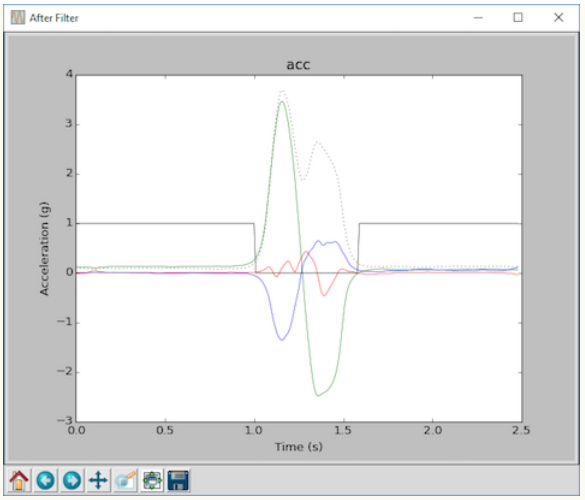


High-pass Butterworth Filter (Click for more details)



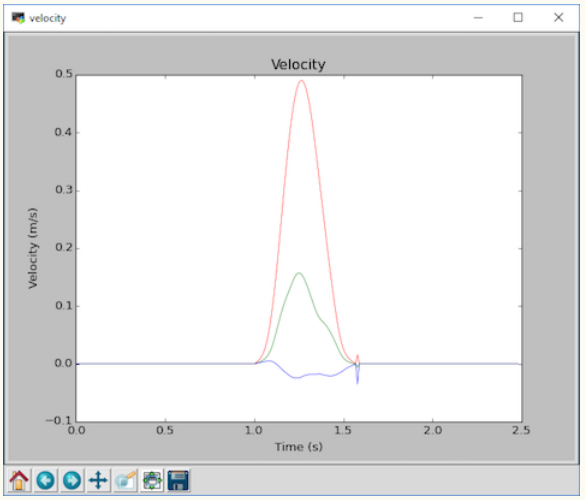
Acceleration Fata After Filter (Click for more details)

It could be seen that the noises and spikes are greatly removed.The magnitude of the acceleration is also filtered by the filters and the parts where the magnitude is smaller than a threshold 0.05 is called stationary parts, and to minimize the drift in the stationary parts, we only do the integration of the acceleration in the nonstationary parts.

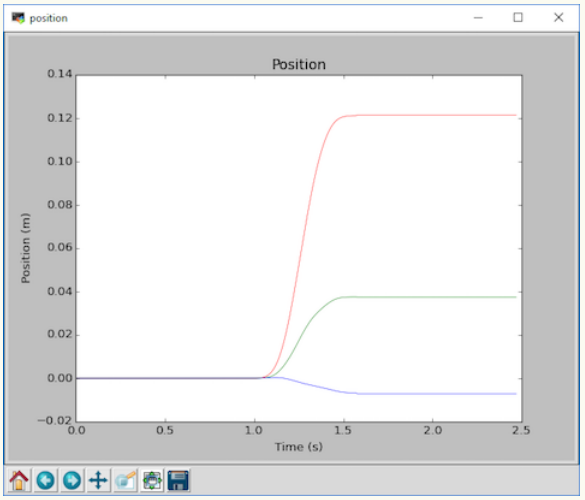


Stationary Section (Click for more details)

The acceleration data is represented in the IMU frame with a quaternion of q with respect to the world frame. Using the quaternion rotation of the frame mentioned in the “background knowledge” part, we could transfer the acceleration vector in IMU frame to the world frame. The velocity and position information calculated by the integration of the acceleration data could be seen below:

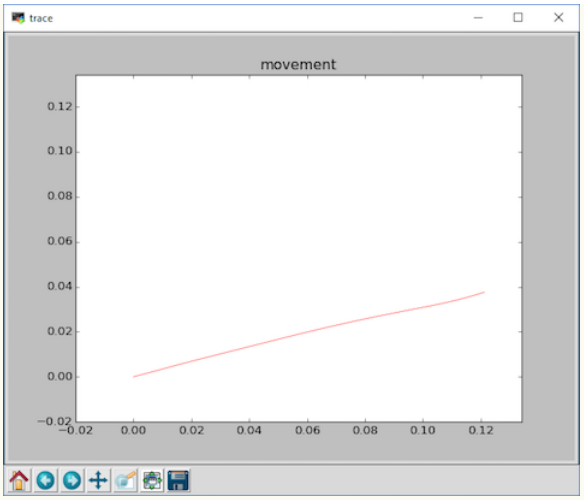


Velocity Data Calculated By Acceleration Integration (Click for more details)



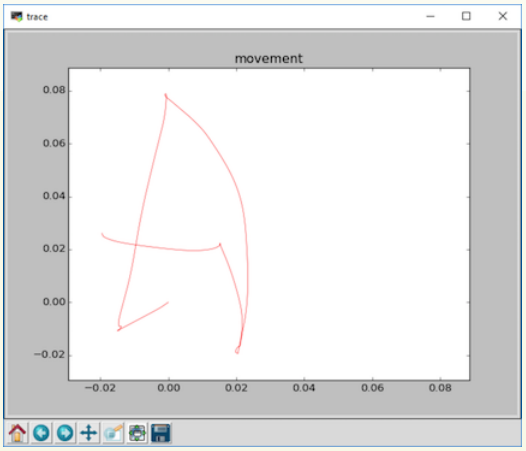
Position Data Calculated By Velocity Integration (Click for more details)

The movement in the world X-Y plane could be seen as:



IMU Movement In World X-Y Plane (Click for more details)

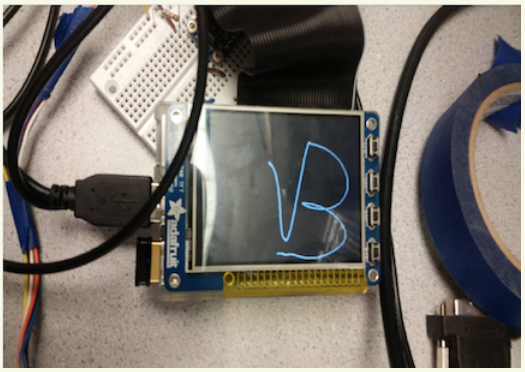
More experiment result could be seen below:



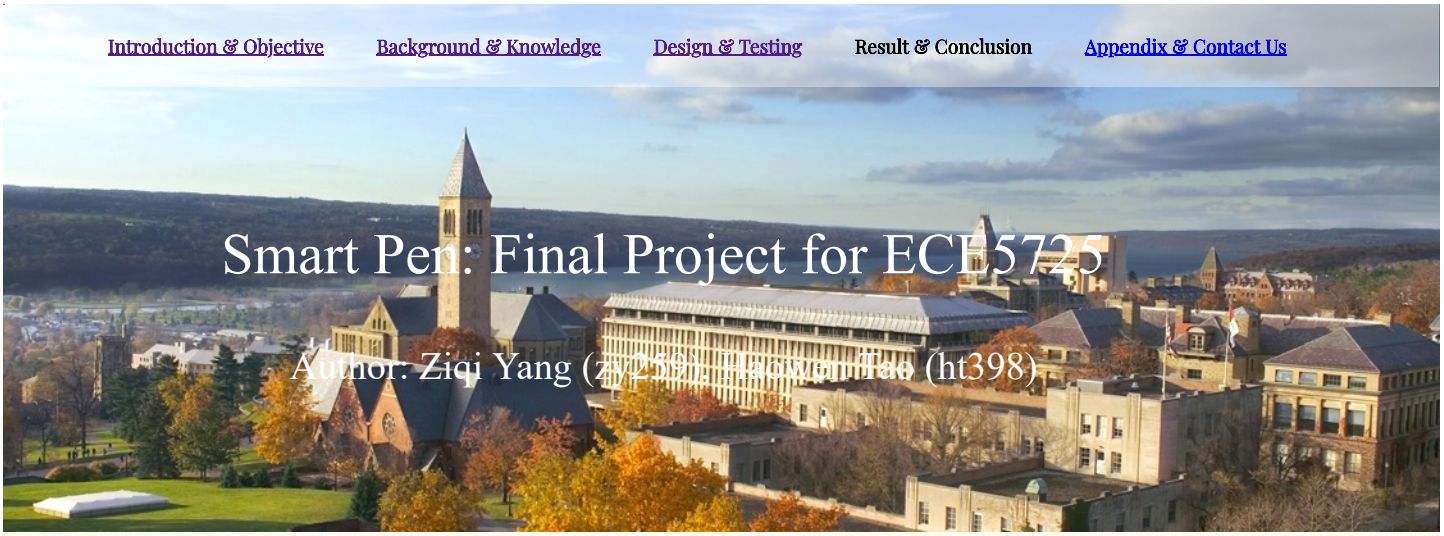
IMU Movement In World X-Y Plane (Click for more details)

Pygame Setup

To make the IMU movement easier to see between each update, we used Pygame to draw the trace of the IMU. We use a press button connected to the Pi GPIO input port to control the trace update, after the first press of the button, the system will begin to record the movement, and will stop the record at the second press. Then the movement recorded by the system will be scaled to fit the size of the TFT screen, and draw on the screen. The result could be seen as below:



Movement Shown On TFT Screen (Letter 'B')



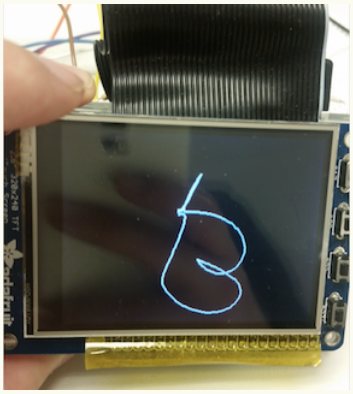
Smart Pen: Final Project for ECE5725

Author: Ziqi Yang (zy259), Haowen Tao (ht398)

Result

As can be seen from the final result, we were able to meet many of the goals outlined in the previous sections. Although the whole system is not capable of fully restoring the exact movement of the IMU module, it could return a recognizable result of the overall pattern of the movement. Our final version was able to restore the lines very well, but with the flaw on the restore of the turning, we cannot do the restoration on more complex patterns.

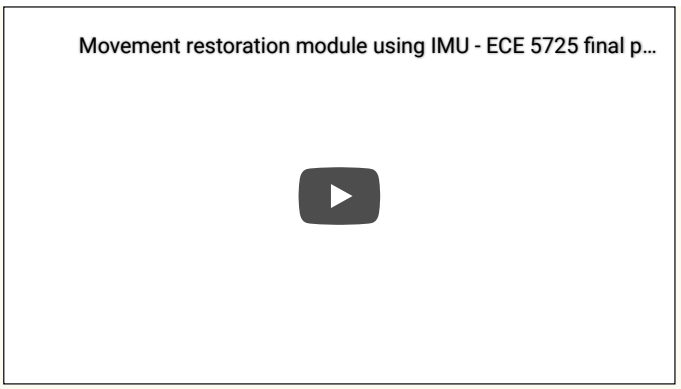
Several pictures of the result of handwriting pattern restoration can be seen below:



Letter 'B'

Letter 'A'

Number '9'



Conclusion

This project tries to restore the movement of an inertial measurement unit using the accelerometer and gyroscope data output, and output the result to to the TFT screen. The LSM9DS0 and BNO055 is used to report the movement parameters including acceleration and rotation speed to the Raspberry Pi and the Raspberry Pi is used to do the sensor fusion and plot the result.

The following functions were functions were successfully implemented:

1.Data calibration and data fusion using IMU raw data.

2.Raw acceleration filtering

3.Simple fast movements restoring: straight lines, curved lines and some simple characteristics

There are several flaws in the current setup:

1.The drift of error grows over time, and the accumulated error will distort the restored image, and it will get worse as the record time goes on.

2.The angles of the turns during the movement cannot be accurately restored.

3.Datas obtained from different recording cannot be combined together the movement between each recording cannot be restored.

In conclusion, our team implemented a handwriting tracking system using Raspberry Pi and IMU BNO055, which can track movements in a small range, such as english characters. The whole system works in a very functional way. By pressing the button on our module, users can easily start the recording process and repressing the button again will immediately stop the recording. And then their writing trajectory will be automatically displayed on the RPi TFT screen.

Our experiments and results have shown that the system will be more accurate and suitable if writings have more straight lines and arc curves.

Future Work

In the work of gait tracking using IMU, the quaternion and acceleration data is reset, and error is corrected when the foot touch the ground, thus the error will not accumulate from one step to another. A similar method could be used, every time the IMU is detected touched the ground, the quaternion and acceleration will be recalibrated.

Another way to reduce the error is to add a feedback loop, just like the way an optical mouse works, a small camera could be used to track its own movement, the result of the camera tracking could be used as a feedback of the IMU movements.
