Progress Report
Human Arm Simulation Model

系級：電機四 學號：B03901065 姓名：林宣竹

January 22, 2018

1 Introduction

By computing the IMU data from two Myos on upper arm and forearm, we can simulate the motion of human arms. I have analyzed the angle of each joint last semester, but there still exists possiblity for improvement. Thus, I aim to revise the measurement that I used before, hoping to get a more precise result. Furthermore, I try to develop a 3D arm simulation model so that we can better visulaize the value and the change of each joint angle. In the following sections, I will further demonstrate the details of the work I have done this semester.

2 Optimization of previous project

2.1 Previous work

Last semester, I used the IMG data given by Myos to derive the roll, pitch, and yaw angles of upper arm and forearm and applied complementary filter to solve the gyroscopic drift. With these angles, the movement of upper arm and forearm can be analyzed, and the joint angle of elbow can be further derived. However, in the method I previously used, some angles may drift when several joints move simultaneously. Thus, I came up with another method to calculate the elbow angle.

2.2 Optimization 1 - Mapping

Since the rotation movement on upper arm and forearm are based on different frame, I tried to change the descriptions from down frame to upper one, that is, mapping. First, we assume that the initial position of arm vector being (1,0,0) for both upper arm and forearm, which indicates hand pointing to the front. Applying the rotation matrices to the initial vector results in the current arm vector of both upper arm and forearm in their relative frame. Then, we need to transform the vector of forearm from down frame to the upper one. This transformation depends on the rotation matrix dervied from the rotation angles of forearm. We can describe the transformation in following equation, where $\psi, \theta, and \phi$ represent yaw, pitch, and roll angle of forearm respectively.

$\mathbf{ForearmVector_{UpperFrame}} = R_z(\psi)R_y(\theta)R_x(\phi)\mathbf{ForearmVector_{DownFrame}}$

$$= \begin{pmatrix} cos(\psi)cos(\theta) & sin(\psi)cos(\phi) + cos(\psi)sin(\theta)sin(\phi) & sin(\psi)sin(\phi) + cos(\psi)sin(\theta)sin(\phi) \\ sin(\psi)cos(\theta) & cos(\psi)cos(\phi) + cos(\psi)sin(\phi) & -cos(\psi)sin(\phi) + sin(\psi)sin(\theta)cos(\phi) \\ -sin(\theta) & cos(\theta)sin(\phi) & cos(\psi)cos(\theta) \end{pmatrix}$$

$\mathbf{DownArmVector_{DownFrame}}$

By computing the inner product of upper arm vector and forearm vector in the upper arm frame, we can get the joint angle of the elbow.

$\mathbf{JointAng} = \mathbf{UpperArmVector_{UpperFrame}} \cdot \mathbf{ForearmVector_{UpperFrame}}$

2.3 Optimization 2 - Kalman Filter

Kalman filter is an algorithm that can estimate the state of the system, which based on a series of measurements observed over time, in this context an accelerometer and a gyroscope. I applied Kalman filter to solve the problem that the gyroscope drifted over time and got a pretty nice result. In general, the state of the system at time k in Kalman filter is given by:

$\mathbf{x_k} = \mathbf{F}x_{k-1} + \mathbf{B}u_k + w_k$

In this case:

$\mathbf{x_k} = \mathbf{F}x_{k-1} + \mathbf{B}\dot{\theta_k} + w_k$

where $\mathbf{x_i} = \begin{pmatrix} \theta_i \\ \dot{\theta_i} \end{pmatrix}$: state at time i, including the angle θ and the bias $\dot{\theta}$

$\mathbf{F} = \begin{pmatrix} 1 & -\delta t \\ 0 & 1 \end{pmatrix}$: prediction model, δt: sample rate

$\mathbf{B} = \begin{pmatrix} \delta t \\ 0 \end{pmatrix}$: control model

w_k: process noise which is Gaussian distributed with a zero mean and with covariance Q, and $\mathbf{Q} = \begin{pmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{pmatrix}$

We set $Q_\theta = 0.001 and Q_{\dot{\theta}_b} = 0.003$ through trial and error. Then, we can estimate the current state by previous measurement. After integrating this method into programming, we can get more accurate and robust IMU data of Myo.

We simply test the motion of lifting the right arm above the head and lowering it only by accelerometer(blue line) or gyroscope(orange line), and also the result with the application of Kalman filter(gray line). The figures below show the differences. Apparently, accelerometer is very sensitive but has a better performance in long term, while gyroscope provide accurate data in short term. Combining two data through the filter results in more robust and accurate measurement.

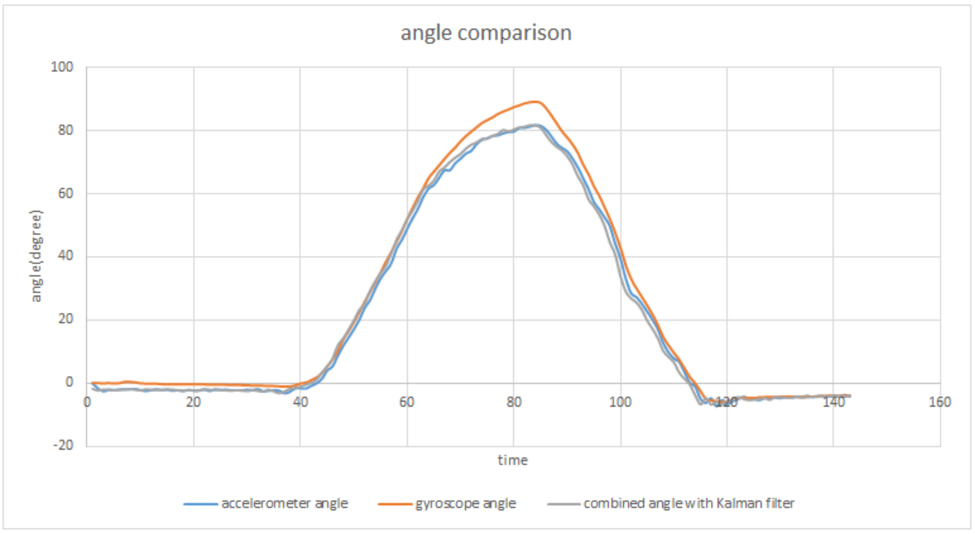


Figure 1: rotating the arm 90 about y axis, that is, pitch angle growing from 0 to 90, and then back to 0

2.4 Result

After optimization, I compare its performance with the original one and get the following result.

First, I try lifting my arm above the head and put it down. As figure 2 shows, both the original and new method perform well in pitch angle, which smoothly increase to 90 degree and drop back to 0 degree. However, the roll angle in original method, which should maintain around 0 degree, drifts up to 70 degrees while the pitch angle increasing. By contrast, in new method, the roll angle stay in the range of ± 15 degrees, which shows a better performance.

Then, I verify another situation, whose result displays in figure 3. This time, I rotate my arm about z axis, which indicates that the yaw angle should change responsively. The new method has a good result in yaw angle as the old one, and its preformance in the joint angle of elbow improves.

As the consequences shows, the new method has solved the angle shifting problem when multiple joint move at the same time and result in a better performance.

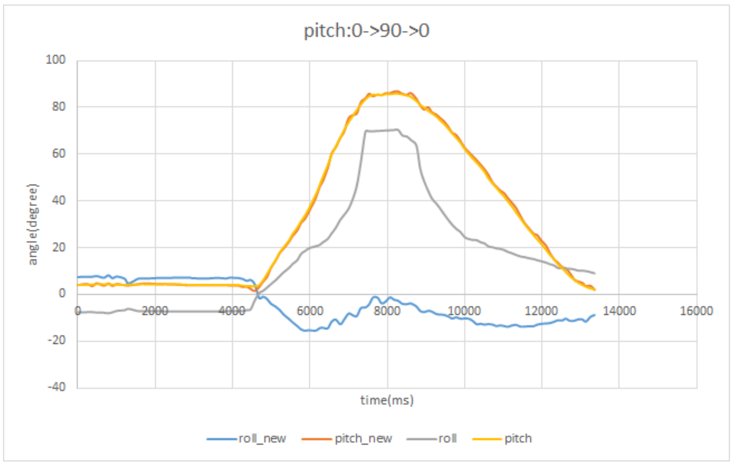


Figure 2:

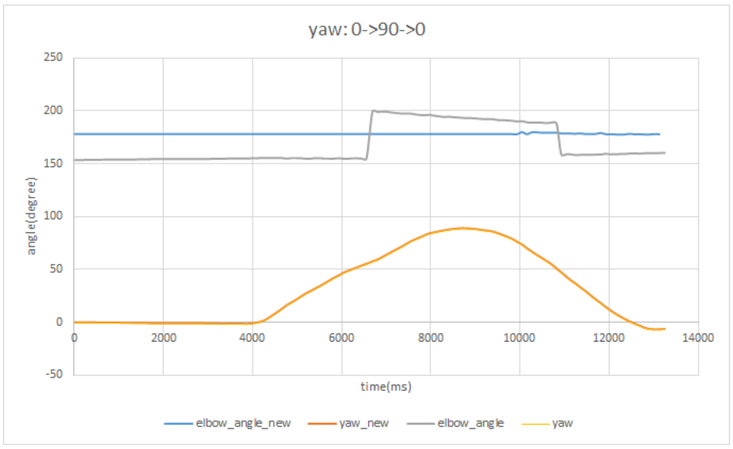


Figure 3:

3 3D arm simulation model

3.1 3D models of human arm

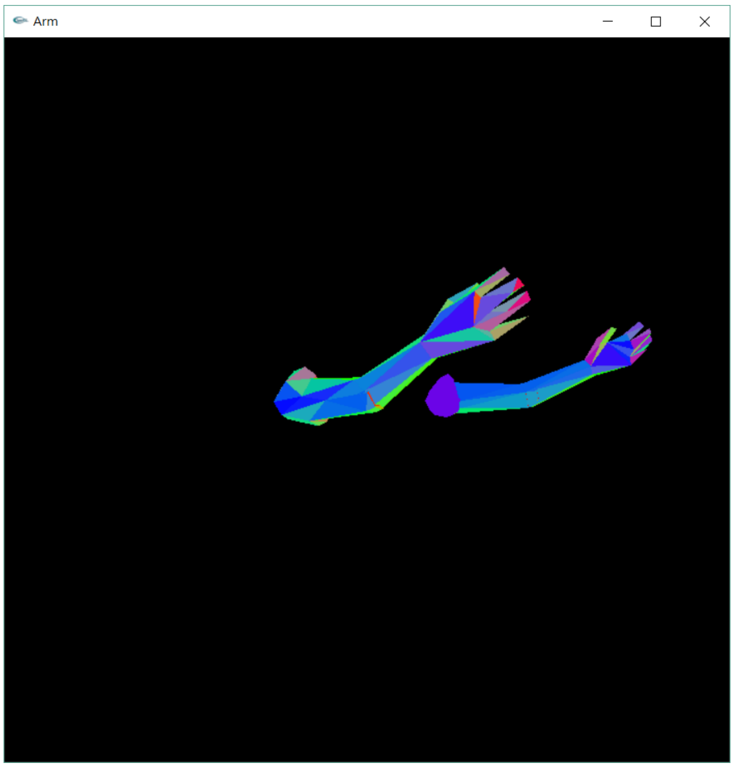
In order to build a arm simulation model, we should find the 3D human arm model first. I serached through the internet and got an obj file of human arm. The file should be transformed into stl file, which can be read by OpenGL, and cut into several parts, including upper arm, elbow, and forearm.

3.2 Animation by OpenGL

Read the stl files of each part. Translate and rotate them to appropriate position by adjusting the parameter of matrices so that the model can work as real human arms. The following figure shows the complete arm simulation model.

3.3 Receiving IMG data from Myo

After constructing the 3D arm model, we should receive the four joint angles detected



by Myos. I used socket to build the connection between two programs, and the arm simulation model can successfully emulate how human arms move in real time.

4 Conclusion

In conclusion, I have fixed some problem that occurred while reading and calculating the data extracted from Myos and made the sensor fusion algorithm more robust. Also, a 3D arm simulation model is built in OpenGL to better visualize each joint angle. By the way, the simulation model updating frequency is 10Hz, which enable the arm module moves quite smoothly. Through the method above, we accurately simulate human arms motion. Moreover, it can be applied to several fields that stongly needed wearable devices, such as VR games or medical appliance.

5 Reference

1. A practical approach to Kalman filter: `http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/`

2. Loading 3D files from OpenGL Introduction: `http://www.codersource.net/2011/01/29/loading-3d-files-from-opengl-opengl-tutorial-7/`

3. The OpenGL Utility Toolkit (GLUT) Programming Interface API Version 3: `https://www.opengl.org/resources/libraries/glut/spec3/spec3.html`