ESE650 Project 4: Gesture Recognition

Code Due Date: **3/21/2017 at 1:20pm** on Canvas, <pennkeyID>_project4.zip
Report Due Date: **3/22/2017 at 11:59pm** on Canvas, <pennkeyID>_project4.pdf

In this project, you will use IMU sensor readings from gyroscopes and accelerometers to train a set of Hidden Markov Models and recognize different arm motion gestures. You should then be able to classify unknown arm motions in real-time using your algorithm.

Training Data Download: Now available at https://upenn.box.com/v/ese650pj4-train-tmp
Additional Training Data Download: https://upenn.box.com/v/ese650pj4-train-add-tmp
The additional training data may be used to improve your model.
Test Data Release: 3/21/2017 3:00pm at https://upenn.box.com/v/ese650pj4-test
The data sets contain the raw IMU readings and corresponding labels that describe the arm motions associated with the movements. The datasets were collected from a consumer mobile device so there is no need to consider bias/sensitivity issues as you may have done in previous projects. You can find the coordinate system used here:
http://developer.android.com/reference/android/hardware/SensorEvent.html
*Data format for each column (7 columns in total) in the IMU data is
ts Ax Ay Az Wx Wy Wz
(Time (millisecond), 3x Accelerometer (m/s2), 3x Gyroscope (rad/sec))

Upload: on Canvas
(1) **Code** (<pennkeyID>_project4.zip) : Do not include data.
(2) **Write-up** (<pennkeyID>_project4.pdf) : Write a project report including the following sections: Introduction, Problem Formulation, Technical Approach, Results and Discussion. Make sure your results include proper visualization of your gesture classification.

Grading: Rubrics can be found on the Canvas assignment page.
*Clearly presenting your approach in the form of report and presentation and having good algorithm performance are equally important.

1. GETTING INTUITION. Experiment with filtering and quantizing the raw sensor information using Kalman filters. This will allow you to visualize the underlying orientation information as you did in the previous project. Given your filtered observations, you can then train HMMs for each motion gesture class using the Expectation-Maximization procedure.

2. IMPLEMENT HMM. You will use HMM models to describe the corresponding motion gestures. You should initially use a simple left-to-right HMM's in the dynamical models. Write down the corresponding model parameters that will need to be learned from

the training data.

3. TESTING. You should then make sure that your program can take input sensor readings from unknown gestures. You can then compute the log likelihood under the different HMM models, and then show how certain you can classify the unknown motion as a gesture.