Exercise 2: Pose Estimation and Step Counting from IMU Data

Due date: 11:59pm, Oct 21st, 2015

1 Introduction

In this exercise, you will implement pose estimation and step counting. We strongly encourage you to go through the lecture slides before starting this exercise. To ease your work, you are provided with accelerometer, gyro and compass measurements collected in the Information Technology Building (ITB) in the McMaster University.

Your implementation should be based on Matlab/Octave. For instructions on downloading and installing Octave, check out Download GNU Octave. Further documentation for Octave functions can be found at the Octave documentation pages. MATLAB documentation can be found at the MATLAB documentation pages.

2 Pose Estimation

In the first part of the exercise, you will implement the procedure to estimate poses from IMU data collected from a *stationary* device. All IMU measurements are in the device coordinate system (Figure 1). Thus, the main task is to determine the rotation matrix from the device coordinate system to the global coordinate system. Upon obtaining the rotation matrix R, you can then compute the Euler angles, yaw, roll and pitch. For consistency, we assume the rotations follows the order of rotating in counter-clock-wise around the x axis, y axis and finally the z axis (called *x-y-z extrinsic*). Furthermore, for simplicity, the true north is assumed to be identical to the magnetic north.

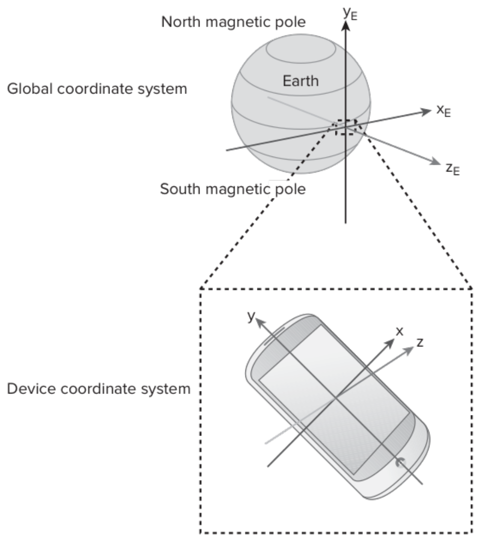

Figure 1: Global coordinate system and device coordinate system (Android)

2.1 Data set

In the data set provided, under the folder pose, there are two files, pose.mat and `imu`.mat

- pose.mat contains the ground truth poses. There are a total of 7 records (cells). Each record has three fields, i) a string "X-Y-Z order", ii) angles of roll-pitch-yaw in radian and iii) the corresponding rotation matrix. These records are for you to verify your implementation. It should be noted due to the lack of precise true north, the ground truth rotation may differ a bit and the angles (in absence of Gimbal lock where $\theta = \pm\pi/2$) may differ by $10 - 20$ degrees.

- `imu`.mat contains the IMU measurements. There are a total of 7 cells. The nth cell contains the 3-axis accelerometer readings, 3-axis gyro readings and 3-axis magnetometer readings, while the ground truth pose is in the nth cell in pose.mat. For each sensor, multiple readings in a time period are stored. The first entry of a reading is the label of the respective data (e.g, acc or gyro), the second entry contains the tuple $\{$*timestamp (ms), x_data, y_data, z_data*$\}$.

2.2 Implementation

In your implementation, you should take the IMU data file name (e.g., `imu.mat`) as an input argument and output the resulting rotation matrices and Euler angles, storing them in the same format as those in pose.mat in a mat file (under a different name). Do **NOT** assume the number of records is fixed (e.g., not necessarily 7). Instead, use `size()` to extract the number of records. For grading, I will use a different IMU dataset with different number of records.

Hint: In evaluating your results, compare the rotation matrix from the ground truth data first. When $\theta = \pm\pi/2$, due to the loss of DoF, the Euler angles are not unique. Thus, when $\theta = \pi/2$ ($\theta = -\pi/2$), you only need to verify if $\psi - \phi$ ($\psi + \phi$) are close to the corresponding values from the ground truth dataset.

3 Step Counting

Congratulations! You are half-way through. Now, we proceed to identify the steps and count steps. In this part of the exercise, you only need to utilize 3-axis accelerometer measurements.

3.1 Data set

In the folder `step/`, you will find the file `step.mat`. Upon loading this file, you will find several 1-d array of the same length.

- `step_event`: a binary array, where the ith element indicates whether a step event is detected ('1') or not ('0') at the ith index (the actual time can be retrieved from the time_stamp array using the index). This serves as the ground truth data. Here, a step event occurs when one of the feet hits the ground.
- `timestamp`: time measured in ms.
- x, y, z: x-, y-, z-axis accelerometer readings.

A total of 100 steps have occurred in this dataset.

3.2 Implementation

Your implementation should take the mat file as input and produce the following outputs:

- **detected_step_event:** similar to step_event, this list should contain binary values for the detected step event by your algorithm.
- **step count and step count error:** the total number of steps and step count error calculated as,

$$error = \frac{|num\_step\_detected - actual\_num\_step|}{actual\_num\_step}, \qquad (1)$$

  where the actual_num_step is 100.

- **False positive (FP), false negative (FN) errors:** To get more fine-grain assessment of accuracy of your implementation, we also provide a function eval_stat.m that computes the FP, FN, true positive (TP), precision and recall. FP and FN are commonly used in evaluating the performance of binary classifier, where FP is an error where the test reports the presence of a condition while in reality there is not; FN on the other hand, occurs when a test result improperly indicates no presence of a condition. The function takes the list of ground truth step events and your detected events as inputs and outputs FP, FN, TP, precision and recall. Try "help eval_stat" for more information.

To help debugging your codes and visualizing each step of the algorithm, it is always a good practice to plot the input data, intermediate results and the final results (e.g., detected step events).

Hint:

- The accelerometer data was collected from an Android device and was sampled as fast as the system can support. As a result, you may note that the sampling intervals vary. You may want to compute an average sample rate from the data first and use it to determine filter parameters.
- For the HPF to determine the linear acceleration, you can use an exponential moving average filter.

- For the LPF applied to the accelerometer magnitude data, you may use a 5th order Butterworth filter with a properly chosen cutoff frequency (`butt(...)`) combined with `filtfilt(...)`. The latter provides zero-phase forward and reverse digital IIR filtering.
- You can directly use (`findpeaks(...)`) for peak detection.

4 Submission

In the report, give a brief description of the algorithms you have implemented, summarize the performance results and key observations. Submit your report (in pdf) and codes (in .m) and output files (in .mat) through SVN.

For pose estimation, summarize the pose estimation error in a table. For step count, include the computed step count, step count error, FP and FN. In the report, also include a plot that shows the accelerometer magnitude ($acc = \sqrt{acc_x^2 + acc_y^2 + acc_z^2}$) over time and highlights the times that step events are detected in the ground truth data and by your algorithm.

Acknowledgement

Many thanks to Qiang Xu for data collection and preparing the test code for step counting.