Exercise 3: Particle Filter for Indoor Positioning using IMU Measurements

Due date: 11:59pm, Nov 13th, 2015

1 Introduction

In this exercise, you will implement the particle filter (PF) framework that incorporates IMU measurements in the motion model and a given map to determine the locations of a moving person in indoor environments. Unlike the general PF introduced in the class, there is no observation from other sensing modalities. As such, state updates are the results of motions and all particles that fall within the confinement of the corridors are assumed to have (non-zero) equal probabilities.

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 Dataset

In the dataset provided, there are two Mat files.

- `IMU.mat` contains 9-axis IMU readings, namely, accelerometer, gyro and magnetometer readings, and `marker_event` that corresponds to the time instances when the person passes the markers on the ground. All timestamps are in milliseconds.

- `map.mat` stores map info and the locations of the markers. A simplified floor map for the ITB 2nd floor is provided. The floor map is essentially defined by an outer polygon that follows the "exterior" of the hallways, and a collection of inner polygons that correspond to blocks

of areas that cannot be accessed. A polygon is defined by a collection of vertices. For convenience, both the vertices of the polygons and the marker locations have been converted using the Universal Transverse Mercator (UTM) projection to the 2-D Cartesian coordinate system, and the south-west corner is set to be the origin.

During the data collection, the device is always face-up with the y-axis in the device frame aligned with the heading direction of the person. This configuration simplifies heading estimations. It should be noted that in practice, phones can have arbitrary attitudes. Accurate heading estimate is in face a non-trivial problem.

In addition to the Mat files, a few supplementary functions have been provided to ease your implementation. Try "help xxx.m" for more information. All codes are self-explanatory.

- `map_visualization.m` displays the floor map and marker locations with x-axis pointing east and y-axis pointing north.
- `OnMapTest.m` returns a binary array indicating whether a collection of points fall with the accessible parts of the map.
- `HeadingFusion.m` returns the heading direction expressed as counter clock-wise angles of the y-axis in the device frame every 20ms. This function implements a complementary filter that combines the heading estimation from gyro and that from accelerometer and magnetometer. This implementation assumes that the device y-axis aligns with the heading direction.
- `fuseTwoHeadings.m` fuses two heading directions with a given weight. This function handles the cases where the headings are more than π apart.
- `fuseTwoAngles.m` fuses two heading angles (ccw with respective to y) with a given weight. This function handles the cases where the headings are more than π apart.

3 Particle Filter for Location Estimation

A particle filter uses Monte Carlo methods for prediction and updates in the Bayesian filter framework. It is particularly suitable when the observation or the motion models are non-linear.

For this assignment, you can assume the initial position is known and is the same as the first marker location. Each particle represents a possible state of the person, namely, her location. In the prediction step, particles' positions are updated based on the IMU readings. For instance, if you choose to make predictions for every footstep, a particle's position is updated based the stride length and the heading direction. Due to the uncertainty in the measurements and estimations, it is advisable to treat the stride length and the heading direction as random variables. For example, if the estimated heading angle is θ, you can draw the actual heading angles uniformly in $[\theta - \Delta\theta, \theta + \Delta\theta]$. Similarly, you can sample the stride length between $[0.7 - \Delta s, 0.7 + \Delta s]$ meter. Both $\Delta\theta$ and Δs can be tuned in your implementation.

The update step of a particle filter updates the weight of each particle based on the observation model. In this project, in absence of an actual observation model, you only need to consider the map constraints, namely, if a particle is outside the accessible area in the map, its weight should be zero. In this case, we can say the particle is dead. A re-sampling step ensues if some particles die in the update step.

To assess the location accuracy of your algorithm, you can compute the centroid of surviving particles and compare them against the ground truth values contained in the `marker_event` and the markers' locations.

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.

Hint:

- One design choice is when to update the particles. You may choose to update them every footstep or every fixed interval. The later case, you would need to estimate the displacement and heading in the interval.
- In some cases, you may find all particles die in an update step. This may happen for a couple of reasons, i) the size of particles is too small, and ii) the displacement estimation is highly inaccurate. To mitigate this problem, you may increase the size of the particles and increase $\Delta\theta$ and Δs. Furthermore, how to reinitialize particles when all of them have died is another design choice to make.
- Roughly, the IMU dataset contains 203 steps. This dataset is collected from a person with relatively light footsteps. You may

need to tune the parameters you used for step counting from the previous assignment.

- Matlab/Octave function `interp1` can be used to interpolate 1-D data to specified points.

4 Submission

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

In the report, you should articulate your design choices when to update the particles and how to re-initialize particles when all of them die. Investigate the impacts of parameters including size of particles, $\Delta\theta$ and Δs on location accuracy.

Acknowledgement

Many thanks to Qiang Xu for data collection and preparing supplementary codes for the project.