EECS 495: Machine Learning and Artificial Intelligence for Robotics

Assignment #0: Filtering Algorithms

Out: September 29th 2014 **Due:** October 13th 2014

This assignment will focus on the implementation and assessment of a filtering algorithm on a robot dataset. You have been assigned to a specific Gaussian or nonparametric filtering algorithm and one of two robot datasets (see blackboard for assignment). The goal is to implement the algorithm, and asses it (guidelines below).

Datasets

Both datasets derive from the same robot platform performing a mobile navigation behavior while visually observing landmarks; the same file formats are used for both. Ground truth positions for the landmarks are known, and should be considered during the measurement update. In the first dataset (ds0) the robot is freely roaming, while in the second dataset (ds1) there are walls that partially occlude landmarks.

The datasets are courtesy of the Autonomous Space Robotics Lab at the University of Toronto. Thorough documentation of the robot platform, task and file formats, as well as video, are provided here: http://asrl.utias.utoronto.ca/datasets/mrclam/index.html. Note that there are data files from multiple robots and multiple experimental runs; feel free (but not obligated) to test your algorithm on other datasets as well. (For your reference, ds0 is [MRSLAM_Dataset4, Robot3] and ds1 is [MRCLAM_Dataset9, Robot3].) Note that the files provided on blackboard have been truncated.

Files:
_Odometry.dat: Controls information (velocities)
_Measurement.dat: Measurement information (visually observed landmarks, and other robots)
_Groundtruth.dat: Ground truth robot position (measured via Vicon motion capture – use for *assessment* only)
_Landmark_Groundtruth.dat: Ground truth landmark positions (measured via Vicon motion capture - may be taken as assumed knowledge for use during filtering)
_Barcodes.dat: Associates the barcode IDs (" Barcode # " in Barcodes.dat, " Subject # " in _Measurement.dat) with landmark IDs (" Subject # " in _Landmark_Groundtruth.dat). *Note that here " Subject # " has mapped to* ***different*** *meanings in different files.* (This was not changed, to keep consistent with the website above.)

Implementation

Each algorithm differs in which parts are the more challenging to implement. Therefore, different weight (with respect to grading) is given to each of the steps below, depending on the filtering algorithm. Higher weight is indicated by the algorithm being cited in brackets (e.g. [UKF] means that the step has higher weight for the UKF algorithm). If higher weight is assigned to your algorithm for a particular step, pay attention – this means the step is either challenging, or key to improved performance and with higher performance expectations as a result. Steps annotated with [*] are higher weight for all algorithms. If you have any questions, ask the instructor.

1. [UKF, PF] Design a simulated controller to estimate the position (2-D location and heading) changes of a wheeled mobile robot in response to commanded speed (translational and rotational) controls. Clearly define the inputs, outputs and parameters of this model, and report the math. Explain your reasoning, as needed.

2. Verify your simulated controller on the following sequence of commands. Report the values of any parameters. (Below, ν is translational speed, ω is rotational speed, and *t* is the duration of the commands.) Report the resulting plot.

$$[\nu = 0.5\frac{m}{s},\ \omega = 0\frac{rad}{s},\ t = 1s]$$
$$[\nu = 0\frac{m}{s},\ \omega = \frac{-1}{2\pi}\frac{rad}{s},\ t = 1s]$$
$$[\nu = 0.5\frac{m}{s},\ \omega = 0\frac{rad}{s},\ t = 1s]$$
$$[\nu = 0\frac{m}{s},\ \omega = \frac{1}{2\pi}\frac{rad}{s},\ t = 1s]$$
$$[\nu = 0.5\frac{m}{s},\ \omega = 0\frac{rad}{s},\ t = 1s]$$

3. Test your simulated controller on the robot dataset. Issue the controls commands (in _Odometry.dat) to your controller, and compare this dead-reckoned path (i.e. controls propagation only) to the ground truth path. Report and discuss the resulting plot.

4. Describe, concisely, the operation of your assigned filtering algorithm. Include any key insights, assumptions, requirements. Use equations in your explanation.

5. [KF, IF] Design a motion model (starting from your simulated controller) that is appropriate for your assigned filter. Report all maths and reasoning; include images for illustration when appropriate.

6. [KF, IF] Design a measurement model that is appropriate for your assigned filter. Report all maths and reasoning; include images for illustration when appropriate.

7. [UKF, PF] Implement the full filter.

8. [*] Compare the performance of your motion model (i.e. your filter without the measurement update step) and your simulated controller on the sequence of commands from step 2, and also on the robot dataset (as in step 3). Report the results, explain any differences.

9. [*] Compare the full filter performance to the robot dataset results reported in step 8. Provide explanations for any differences (what performs well? what performs poorly? under what conditions? and why?) -- ground your explanations in the algorithm maths.

10. [*] Examine the effect (trends? limits?) of uncertainty in the algorithm (i.e. changing the noise parameters). Examine its performance at different parts of the execution (e.g. when missing an expected measurement reading). Report plots and/or tables, and provide explanations.

Code

To be submitted with the report. Development in Matlab, Octave or Python is a requirement. A single executable script file named "run.m" or "run.py" should output, when executed, all of the data (plots, table values) reported in the write-up (as instructed above), with clear presentation and labels. *Code that does not run will receive zero points.* The run.m/run.py file should be easy to read – a sequence of calls to standalone functions, that hide away the meat of code (rather than placing that code in the run function). Clear commenting for all functions and function calls is encouraged. (No need for excessive verbosity, but the quicker and easier we are able to read the code, the better for your grade...) You can assume that the data files provided on Blackboard will be available in the same directory that your run.m/run.py file is run.

Write-up

For each step above, explain thoroughly, and with clean consistent maths, all design decisions (motion model, measurement model, how parameters were set...) and reported assessment results. Remember to label all plot axes, title all plots, provide captions for any figures, include (as appropriate) units for all reported numbers, etc. Language counts – specifically with respect to clarity. If you use other sources (e.g. follow guidelines to linearize a motion model as described in a scholarly article), be sure to cite them.

Hints

- Start simple, plot often and build on working components. For example, first verify that the motion model on its own and the filter prediction step produce similar/identical paths for noise parameters near zero, and only after this include the measurement update step.

- Start early. Successful formulations will require time to develop. If you start only a few days before the deadline, your chances of finishing a high performing implementation in time are *extremely* slim.

- Note the sampling rate of controls versus observations... it might be that multiple controls are executed before an observation is taken (and vice versa). Pay attention to timestamps; take care to synchronize data.

- For KF and IF implementations, the linear approximation *cannot* be a first-order Taylor Series approximation. Be careful about not embedding non-linear computations into the parameters of your A and C matrices. (And if you aren't able to come up with a fully non-linear approximation, be sure to acknowledge this in steps 5 and 6.)