TTK4250 Sensor Fusion

Graded Assignment 3 (15% of total grade)

Code and results hand in (5%): *Friday 22. November 16.00* on Blackboard as a single .zip file.
Report hand in (10%): *Wednesday 27. November 16.00 (preliminary deadline)* on Blackboard as a single PDF file.

Pairs: This assignement is to be solved in pairs. These pairs will be the same for all the graded assignments.

Code hand in: Functioning code that produces reasonable, but not necessarily perfect, results for task 3, along with one paragraph describing why you ended up tuning as you did in task 3, should be handed in by 22. November 16.00 as a zip file on Blackboard. This will account for one third of the possible points for this assignment (5% of total grade).

Report hand in: At the end of the semester you will hand in a report for all the graded assignements combined. The whole report should be maximum 10 pages long, including abstract, introduction and conclusion. This leaves approximately 3 pages per assignment, although this is *up to you*. You should, therefore consider carefully what to have in it. In addition to the 10 pages we give you one page per assignment for the plot that is asked for, giving a maximum of 13 pages. Note that these extra pages cannot be used for other material than these plots. We recomend starting to write this now, when you actually are given some time to do it.

A full answer and result analysis is expected for task 3 and 4. For task 3 you should include a plot of the pose over time as well as the final estimated landmarks for your choice of parameters, in addition to NEES and NIS over time for the same set of parameters. In task 4 the same plots are expected, where the GPS can be used for ground truth. For both tasks it should be made clear why the parameters were chosen in terms of error metrics, consistency and overall result. Answers and analysis should connect theory and results to the real world, and show your understanding for the problem and solution. Try to connect the results on the simulated data to the results of the real data where and if it is possible.

Task 1: *Implement EKFSLAM*

(a) Implement the motion prediction model, `EKFSLAM.f(-, x, u)`, corresponding to equation (11.7) that takes a pose and odometry and predicts it to the next time step,

(b) Implement the motion prediction Jacobian, `EKFSLAM.Fx(-, x, u)`, that is the Jacobian of the above function with respect to the pose, given by equation (11.13).

(c) Implement the motion prediction Jacobian, `EKFSLAM.Fu(-, x, u)`, hat is the Jacobian of the above function with respect to the odometry, given by equation (11.14).

(d) Implement the predict function, `EKFSLAM.predict(obj, eta, P, zOdo)`, which takes the state (pose and map) , its covariance and odometry to predict the state and covariance, given by (11.19).

Note that the map is assumed static. This means that you only need to do something with the state and covariance that has to do with the pose. There is a huge save computational cost if you take this into consideration for the covariance matrix when the problem gets large, and also do it in place (ie. reuse the input covaraince matrix). This should also be evident from F in (11.18).

(e) implement the measurement function, `EKFSLAM.h(obj, eta)` that predicts the measurements of all the landmarks in the state, corresponding to equations (11.10) - (11.11).

Note that there is a offset between the robot center and the sensor location in the Victoria Park data set making us having to use $m^i - \rho_k - R(\psi_k)L$ instead of simply $m^i - \rho_k$, where L is the offset vector in body frame.

(f) implement the jacobian of the above measurement function, `EKFSLAM.H(obj, eta)`, given by equations (11.16) - (11.18). This matrix is dense in the three leftmost columns corresponding to the pose, and block diagonal for the landmarks.

We need to compensate for the sensor offset in the jacobian as well. Using $\frac{\partial}{\partial x}||x|| = \frac{x^T}{||x||}$, and $\frac{\partial}{\partial x}\angle(x) = \frac{x^T R(\pi/2)^T}{||x||^2}$ along with $z_c = m^i - \rho_k - R(\psi_k)L$ and $z_b(x) = R(-\psi)z_c = R(-\psi)(m^i - \rho_k) - L$, it can be shown that the measurement Jacobians can be written as

$$\frac{\partial}{\partial x}||z_b(x)|| = (\frac{\partial}{\partial z_b}||z_b(x)||)(\frac{\partial}{\partial x}z_b(x)) = \frac{z_c^T}{||z_c||}\begin{bmatrix}-I_2 & -R(\frac{\pi}{2})(m^i - \rho_k)\end{bmatrix},$$
$$\frac{\partial}{\partial x}\angle z_b(x) = (\frac{\partial}{\partial z_b}\angle(z_b(x)))(\frac{\partial}{\partial x}z_b(x)) = \frac{z_c^T R(\frac{\pi}{2})^T}{||z_c||^2}\begin{bmatrix}-I_2 & -R(\frac{\pi}{2})(m^i - \rho_k)\end{bmatrix}.$$

(g) Implement the function that inverts the masurement function and creates new landmarks and their covariances in the function `EKFSLAM.addLandmarks(obj, eta, P, z)`.

The covariances are generated using

$$G_x^j = \frac{\partial}{\partial x}h^{-1}(z,x) = \begin{bmatrix}I_2 & z_r^j\begin{bmatrix}-\sin(z_\phi^j + \psi)\\ \cos(z_\phi^j + \psi)\end{bmatrix} + R(\psi + \frac{\pi}{2})L\end{bmatrix}$$
$$G_z^j = \frac{\partial}{\partial z}h^{-1}(z,x) = R(z_\phi^j + \psi)\mathrm{diag}(1, z_r^j)$$
$$P_{\text{new}} = \begin{bmatrix}P & P_{:,x}\{G_x^j\}^T\\ \{G_x^j\}P_{x,:} & \{G_x^j\}P_{xx}\{G_x^j\} + \mathrm{blkdiag}(G_z^j R(G_z^j)^T)\end{bmatrix}$$

(h) Implement the update function, `EKFSLAM.update(obj, eta, P, z)`, that takes the prior mean and covariance and a set of measurements to update the map and pose estimates as well as calculating NIS and creating new landmarks. The data association is done for you so you only need to make make the EKF update.

Task 2: *Run EKFSLAM on simulated data*

You are given a data set consisting of

- odometry (3 x K): the measured movement of the robot in body frame
- z (K x 1): cell of detections at each time step
- poseGT (3 x K): the pose ground truth at each time step
- landmarks (2 x M): the ground truth landmarks

Use EKFSLAM on this dataset.

Task 3: *Run EKFSLAM on the Victoria Park dataset*

The victoria park data set is a SLAM data set available along with some additional information here. All times are converted to seconds for you in the script, and a function to create odometry and detections from the data are also provided. The dataset consists of

- LASER (Klsr x 361): Laser scanner returns. There are 361 returns per scan evenly divided in a half circle. The MATLAB file `viewLsr.m` provides a scripts to compare the detections and raw laser scans.

- speed (K x 1): measured speed from a wheel encoder at the rear left wheel.
- steering (K x 1): measured wheel steering angle
- Lo_m (Kgps x 1): GPS longitude in meters
- La_m (Kgps x 1): GPS latitude in meters
- TLsr (Klsr x 1): Time of laser scans in milli seconds
- time (K x 1): Time of odometry measurements in milliseconds
- timeGPS (Kgps x 1): Time of GPS measurements in milliseconds

See the linked webpage and the files in the Victoria Park folder for further information.

Use EKFSLAM on the victoria park dataset.