Albert-Ludwigs-Universität Freiburg, Institut für Informatik
PD Dr. Cyrill Stachniss
Lecture: Robot Mapping
Winter term 2012

Sheet 2

Topic: Extended Kalman Filter SLAM

Submission deadline: Nov. 12th for Exercises 1 and 2, Nov. 19th for Exercise 3
Submit to: `robotmappingtutors@informatik.uni-freiburg.de`

Exercise 1: Bayes Filter and EKF

(a) Describe briefly the two main steps of the Bayes filter in your own words.

(b) Describe briefly the meaning of the following probability density functions: $p(x_t \mid u_t, x_{t-1})$, $p(z_t \mid x_t)$, and $\mathrm{bel}(x_t)$, which are processed by the Bayes filter.

(c) Specify the (normal) distributions that correspond to the above mentioned three terms in EKF SLAM.

(d) Explain in a few sentences all of the components of the EKF SLAM algorithm, i. e., μ_t, Σ_t, g, G_t^x, G_t, R_t^x, R_t, h, H_t, Q_t, K_t and why they are needed. Specify the dimensionality of these components.

Exercise 2: Jacobians

(a) Derive the Jacobian matrix G_t^x of the noise-free motion function g with respect to the pose of the robot. Use the odometry motion model as in exercise sheet 1:

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} \delta_{trans} \cos(\theta_{t-1} + \delta_{rot1}) \\ \delta_{trans} \sin(\theta_{t-1} + \delta_{rot1}) \\ \delta_{rot1} + \delta_{rot2} \end{pmatrix}.$$

Do not use Octave for this part of the exercise.

(b) Derive the Jacobian matrix ${}^{\mathrm{low}}H_t^i$ of the noise-free sensor function h corresponding to the i^{th} measurement:

$$h(\bar{\mu}_t, j) = z_t^i = \begin{pmatrix} r_t^i \\ \phi_t^i \end{pmatrix} = \begin{pmatrix} \sqrt{(\bar{\mu}_{j,x} - \bar{\mu}_{t,x})^2 + (\bar{\mu}_{j,y} - \bar{\mu}_{t,y})^2} \\ \mathrm{atan2}(\bar{\mu}_{j,y} - \bar{\mu}_{t,y}, \bar{\mu}_{j,x} - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \end{pmatrix},$$

where $(\bar{\mu}_{j,x}, \bar{\mu}_{j,y})^T$ is the pose of the j^{th} landmark, $(\bar{\mu}_{t,x}, \bar{\mu}_{t,y}, \bar{\mu}_{t,\theta})^T$ is the pose of the robot at time t, and r_t^i and ϕ_t^i are respectively the observed range and bearing of the landmark. Do not use Octave for this part of the exercise.

Hint: use $\frac{\partial}{\partial x}\mathrm{atan2}(y, x) = \frac{-y}{x^2+y^2}$, and $\frac{\partial}{\partial y}\mathrm{atan2}(y, x) = \frac{x}{x^2+y^2}$.

Exercise 3: Implement an EKF SLAM System

Implement an extended Kalman filter SLAM (EKF SLAM) system. To support this task, we provide a small Octave framework (see course website). The framework contains the following folders:

data contains files representing the world definition and sensor readings.

octave contains the EKF SLAM framework with stubs to complete.

plots this folder is used to store images.

The below mentioned tasks should be implemented inside the framework in the directory `octave` by completing the stubs.

After implementing the missing parts, you can run the EKF SLAM system. To do that, change into the directory `octave` and launch *Octave*. Type `ekf_slam` to start the main loop (this may take some time). The program plots the current belief of the robot (pose and landmarks) in the directory `plots`. You can use the images for debugging and to generate an animation. For example, you can use *ffmpeg* from inside the `plots` directory as follows:

```
ffmpeg -r 10 -b 500000 -i ekf_%03d.png ekf_slam.mp4
```

(a) Implement the prediction step of the EKF SLAM algorithm in the file `prediction_step.m`. Use the Jacobian G_t^x you derived above to construct the full Jacobian matrix G_t. For the noise in the motion model, assume

$$R_t^x = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.01 \end{pmatrix}.$$

(b) Implement the correction step in the file `correction_step.m`. The argument z of this function is a struct array containing m landmark observations made at time step t. Each observation $z(i)$ has an id $z(i).id$, a range $z(i).range$, and a bearing $z(i).bearing$.

Iterate over all measurements ($i = 1, \ldots, m$) and compute H_t^i using the Jacobian you derived above. You should compute a block Jacobian matrix H_t by stacking the H_t^i matrices corresponding to the individual measurements. Use it to compute the Kalman gain and update the system mean and covariance *after* the for-loop. For the noise in the sensor model, assume that Q_t is a diagonal square matrix as follows

$$Q_t = \begin{pmatrix} 0.01 & 0 & 0 & \cdots \\ 0 & 0.01 & 0 & \cdots \\ 0 & 0 & 0.01 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \in \mathbb{R}^{2m \times 2m}.$$

Some implementation tips:

- Turn off the visualization to speed up the computation by commenting out the line `plot_state(...` in the file `ekf_slam.m`.
- While debugging, run the filter only for a few steps by replacing the for-loop in `ekf_slam.m` by something along the lines of `for t = 1:50`.
- The command `repmat` allows you to replicate a given matrix in many different ways and is magnitudes faster than using for-loops.
- When converting implementations containing for-loops into a vectorized form it often helps to draw the dimensions of the data involved on a sheet of paper.
- Many of the functions in *Octave* can handle matrices and compute values along the rows or columns of a matrix. Some useful functions that support this are `sum`, `sqrt`, and many others.