Probabilistic Mobile Robotics - EE698G

Assignment- 4

- Due Date: **....., 2017, 11:59pm**
- Late submission penalty is 20% for every late day.
- Please submit MATLAB codes to moodle. You can submit other solutions by
  (i) **(Recommended)** Typing it in latex and uploading the PDF to moodle along with codes.
  (ii) Dropping handwritten solutions in the mailbox.
  (iii) Creating a PDF by scanning handwritten documents and uploading it to moodle – should be clear.
- **MATLAB**
  (i) Submit a single .zip file for this assignment with every question in different directory.
  (ii) The name of the top-level directory should be your roll number.
  (iii) Include a README textfile in the .zip file. It should mention which scripts to run to generate the desired results for each question.

Q.1 [MATLAB problem] EKF.

In this programming assignment you have to perform localization of a mobile robot traversing in a static environment (with landmarks) using EKF (Extended Kalman filter). You are given snippets of MATLAB files that help you in getting the task done. All you have to do is to code the prediction (**predict.m**) and the correction **(update.m)** steps of the filter.

Note :

- You have to run '**main.m**' file after coding the above mentioned files.
- Open '**ekf_localization_sim.m**' and go through the comments while observing the code, to understand the flow. In this file you will encounter two functions **predict()** and **update()** that are to be coded.
- The rest of the files contain some helpful functions. Interested people can go through them. To gain good understanding of the flow, do read the comments in each file.
- The configuration of the EKF-simulator is managed by the script file '**configfile.m**'. You have the flexibility to alter the parameters of the vehicle, sensors, etc.
- You can really play with the simulator by changing the parameters in '**configfile.m**'. If you don't disturb this file the filter runs with the default parameters.

For you information the true robot path and the estimated robot path will be automatically saved in the workspace. Perform the following detail analysis of your output :

- Plot the x-coordinates of true pose and the estimated pose against time, along with the 3-σ bound.
- Similarly do it for y-coordinates and theta of true pose and the estimated pose in separate figures.
- Also plot the determinant of covariance of estimated pose against time. For your information determinant of covariance matrix is the generalized variance and moreover it captures the volume of your data cloud.
- Change the noise parameters of motion model and measurement model in '**configfile.m**' file and comment on the performance and computation complexity of the filter.

Hints :

- **Prediction :** We use the following **non-linear** motion model :

$$\mathbf{x_t} = \mathbf{f}(\mathbf{x_{t-1}}, \mathbf{u_t}) \Rightarrow \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + V * dt * cos(G + \theta_{t-1}) \\ y_{t-1} + V * dt * sin(G + \theta_{t-1}) \\ \mathbf{pi\_to\_pi}(\theta_{t-1} + G) \end{bmatrix} \tag{1}$$

  where V, G are the noisy controls which represents the linear velocity and angular rotation respectively.

  Note : '**pi_to_pi**' function is given to you to keep the orientations (angles) in the range $[-\pi \ \ \pi]$. Make sure you use this function when representing the robot heading.

- **Update :** Here we use the non-linear measurement model as follows :

$$\mathbf{z_t} = \mathbf{h}(\mathbf{x_t}) + \eta \tag{2}$$

  where $\mathbf{h}(\mathbf{x_t}) = \begin{bmatrix} r \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{(x_t - \ell_x)^2 + (y_t - \ell_y)^2} \\ atan2(\frac{y_t - \ell_y}{x_t - \ell_x}) - \theta_t \end{bmatrix}$ and $\ell = \begin{pmatrix} \ell_x \\ \ell_y \end{pmatrix}$ is the true pose of the landmark.

Note : See the '**configfile.m**' to know about the noise parameters associated with prediction and measurement model.