Intelligent Vehicles
Localization based Extended Kalman Filter and Particle Filter

David Marquez-Gamez
david.marquez@irt-jules-verne.fr

1 Introduction

In this practical work we will use the Extended Kalman Filter (EKF) and the Particle Filter for the localization of a robot. For this, we will use the MATLAB code written by Paul Newman[1]. This code, allows us to : (1) simulate a robot moving on a given path in an environment consisting of known landmarks and (2) implement simple EKF and Particle filter methods using perception of the range and bearing of these landmarks.

2 Robot Model

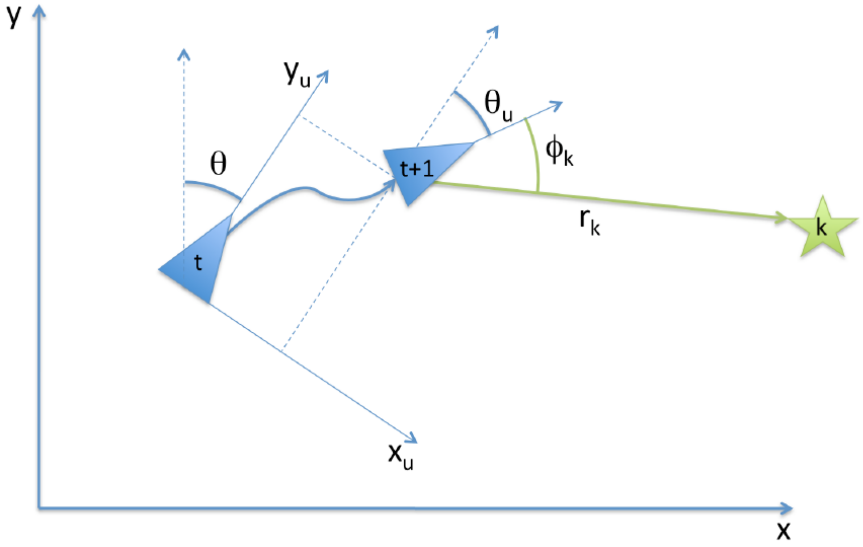

Figure 1: The robot and landmarks

[1] http://www.robots.ox.ac.uk/ pnewman/Teaching/C4CourseResources/C4BResources.html

The robot moves on a plane and perceives range and bearing of point landmarks located on the same plane (Figure 1). The robot state is represented by its position and orientation in a global reference frame:

$$X_t = [x_t, y_t, \theta_t]^T \tag{1}$$

The displacement of the robot between t and $t+1$ is measured through odometry given in the robot frame at time t:

$$U_t = [x_u, y_u, \theta_u]^T \tag{2}$$

The state evolution model is given by:

$$X_{t+1} = f(X_t, U_t) = \begin{bmatrix} x_t + x_u cos(\theta_t) + y_u sin(\theta_t) \\ y_t - x_u sin(\theta_t) + y_u cos(\theta_t) \\ \theta_t + \theta_u \end{bmatrix} \tag{3}$$

$$Z_t = [r_t^k, \phi_t^k]^T \tag{4}$$

$$Z_t = h^k(X_t) = \begin{bmatrix} \sqrt{(x_k - x_t)^2 + (y_k - y_t)^2} \\ atan2(\frac{y_k - y_t}{x_k - x_t}) - \theta_t \end{bmatrix} \tag{5}$$

3 Extended Kalman Filter

3.1 EKF equations and models

Since the models are non-linear, we will use the extended Kalman filter to integrate odometry measurements and landmark observations to produce an estimation $\hat{X}_t$ of robot position X_t.

Let us recall EKF equations:

$$\begin{aligned} X_t^* &= f(\hat{X}_t, U_t) & (6) \\ P_t^* &= A.\hat{P}_t.A^T + B.Q.B^T & (7) \\ Y_t^* &= h(X_t^*) & (8) \\ K &= P_t^*.H^T.(H.P_t^*.H^T + P_Y)^-1 & (9) \\ \hat{X}_{t+1} &= X_t^* + K(Y - Y_t^*) & (10) \\ \hat{P}_{t+1} &= P_t^* - K.H.P_t^* & (11) \end{aligned}$$

where A, B, and H are the jacobian matrices of f and h, respectively:

$$A_{ij} = \frac{\partial f_i(x,u)}{\partial x_j} \tag{12}$$

$$B_{ij} = \frac{\partial f_i(x,u)}{\partial u_j} \tag{13}$$

$$H_{ij} = \frac{\partial h_i(x)}{\partial x_j} \tag{14}$$

The given code, implements EKF localization in a map of known landmarks. We suppose, that the observed landmark is known all the time. Thus, there is no problem of data association.

3.2 Question: Jacobians

1. Open EKFLocalisation.m file. Start by studying the code
2. Calculate the Jacobians J1 (line 52), J2 (line 52) and JH (line 62) and write the corresponding code (function GetObsJac(xPred, iFeature,Map) (line 156 of file EKFLocalisation.m), J1(xEst,u) (J1.m file) and J2(xEst,u) (J2.m file))
3. Run EKFLocalisation function and observe the results. Analyse the obtained graphs.

3.3 Question: Parameters role

Change (play with) the parameters *UEst*, *REst* and *initial conditions* What do you observe and how do you explain these observations?

4 Particle Filter

The Particle Filter uses the same odometry and sensor model as the EKF, but does not need the jacobians to correct prediction errors. Instead the particles likelihood and re-sampling allows concentration of particles around the “true” position. In addition, particle filter allows the management of the global localization.

4.1 Question: Parameters role

1. Open MCL.m file and run the simulation. Start by studying the code
2. Change the different parameters to see their role in the convergence and stability of the filter, precision in the estimation, etc.

   Try to play particularly with:

- The number of particles (nParticles, line 15)
- The error used in particle initialization (diag([8,8,0.4]), line 29)
- The model noise in the filter (UEst, line 23 and REst, line 24)

3. What can you say about the precision of the filter in comparison with EKF ?

4.2 Question: Global localization

Modify the particles initialization (line 29 MCL.m) without taking into account the initial position of the robot (uniform probability throughout the environment).

1. Playing with the parameters, make the filter converge to the “true” position.

2. What do you observe about the precision of localization?