Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lecture: Introduction to Mobile Robotics
Summer term 2007
Prof. Dr. W. Burgard
MSc AI J. Sturm
Dipl.-Inf. B. Lau

Sheet 7

Topic: Extended Kalman Filter III

Submission deadline: Fri 15.06.2007, 11:00 a.m. (before class)

Introduction

This exercise deals with the correction step of the Extended Kalman Filter (EKF). Remember that for last week[1], you have implemented the Kalman Prediction step, which updates the previous belief (represented by μ_{t-1} and Σ_{t-1}) given a motion command u_t (corresponding to an `odometry message`):

$$\begin{aligned} \mu_t &:= g(\mu_{t-1}, u_t) \\ \Sigma_t &:= G_t \Sigma_{t-1} G_t^T + V_t M V_t^T \end{aligned}$$

When the robot makes an observation z_t at a time step t, its internal belief represented by μ_{t-1} and Σ_{t-1} has to be updated (also referred to as the Kalman correction). Here an observation function $\hat{z} = h(x)$ is assumed, that maps the current state x to an expected observation $\hat{z}$. If this observation function h is non-linear in x, then $H = \frac{\partial h(x)}{\partial x}$ (the Jacobian) can be used as a first-order Taylor approximation (linearization).

$$\begin{aligned} K_t &:= \Sigma_{t-1} H_t^T \left(H_t \Sigma_{t-1} H_t^T + Q \right)^{-1} \\ \mu_t &:= \mu_{t-1} + K_t \left(z_t - h(\mu_{t-1}) \right) \\ \Sigma_t &:= \left(1 - K_t H_t \right) \Sigma_{t-1} \end{aligned}$$

Exercise 1:

To keep the sensor model simple, we use the `truepose`-messages and treat them like "extended" GPS-coordinates (x, y, θ); associated with a certain observation noise Q.

What is then the observation function $h(x)$?

What is the linearization $H = \frac{\partial h(x)}{\partial x}$ given this observation function?

Hint: The answer is simple.

[1] As this assignment relies on the correct solution of the previous sheet, please ask us if you have trouble correcting your program (email or in person). We then help you to correct your solution.

Exercise 2:

Add the corresponding code to your program from last week. In order to get a more interesting drawing, use every odometry message to perform the Kalman prediction, but use only every 20th truepose message for the Kalman correction.

The initial state μ_0 is assumed to be unknown, and therefore you should initialize it for example either to $(0, 0, 0)$ or the first odometry estimate (but certainly not to the first truepose). Thereby, the convergence of the Kalman estimate towards the true values becomes visible in the trajectories.

It is certainly useful for visualization/debugging reasons to plot the odometry pose, the true pose and the Kalman estimate on the panel all in the same coordinate system (but different colors).

IMPORTANT NOTE: The `CarmenMatrix.inverse()`-method was faulty, thus make sure you update CarmenMatrix-file (from last week's updated stub-files, or copy the lines below):

```
C.value[0][0] = value[1][1]*value[2][2]-value[1][2]*value[2][1];
C.value[0][1] = -value[0][1]*value[2][2]+value[0][2]*value[2][1];
C.value[0][2] = value[0][1]*value[1][2]-value[0][2]*value[1][1];

C.value[1][0] = -value[1][0]*value[2][2]+value[1][2]*value[2][0];
C.value[1][1] = value[0][0]*value[2][2]-value[0][2]*value[2][0];
C.value[1][2] = -value[0][0]*value[1][2]+value[0][2]*value[1][0];

C.value[2][0] = value[1][0]*value[2][1]-value[1][1]*value[2][0];
C.value[2][1] = -value[0][0]*value[2][1]+value[0][1]*value[2][0];
C.value[2][2] = value[0][0]*value[1][1]-value[0][1]*value[1][0];
```

Exercise 3:

Play around with different values for Σ_0, M and Q. Try to figure out which values could have produced the figures below (based on `fr_sim.log`). Give a short (textual) description, as well as a rough estimate of possible parameter values.

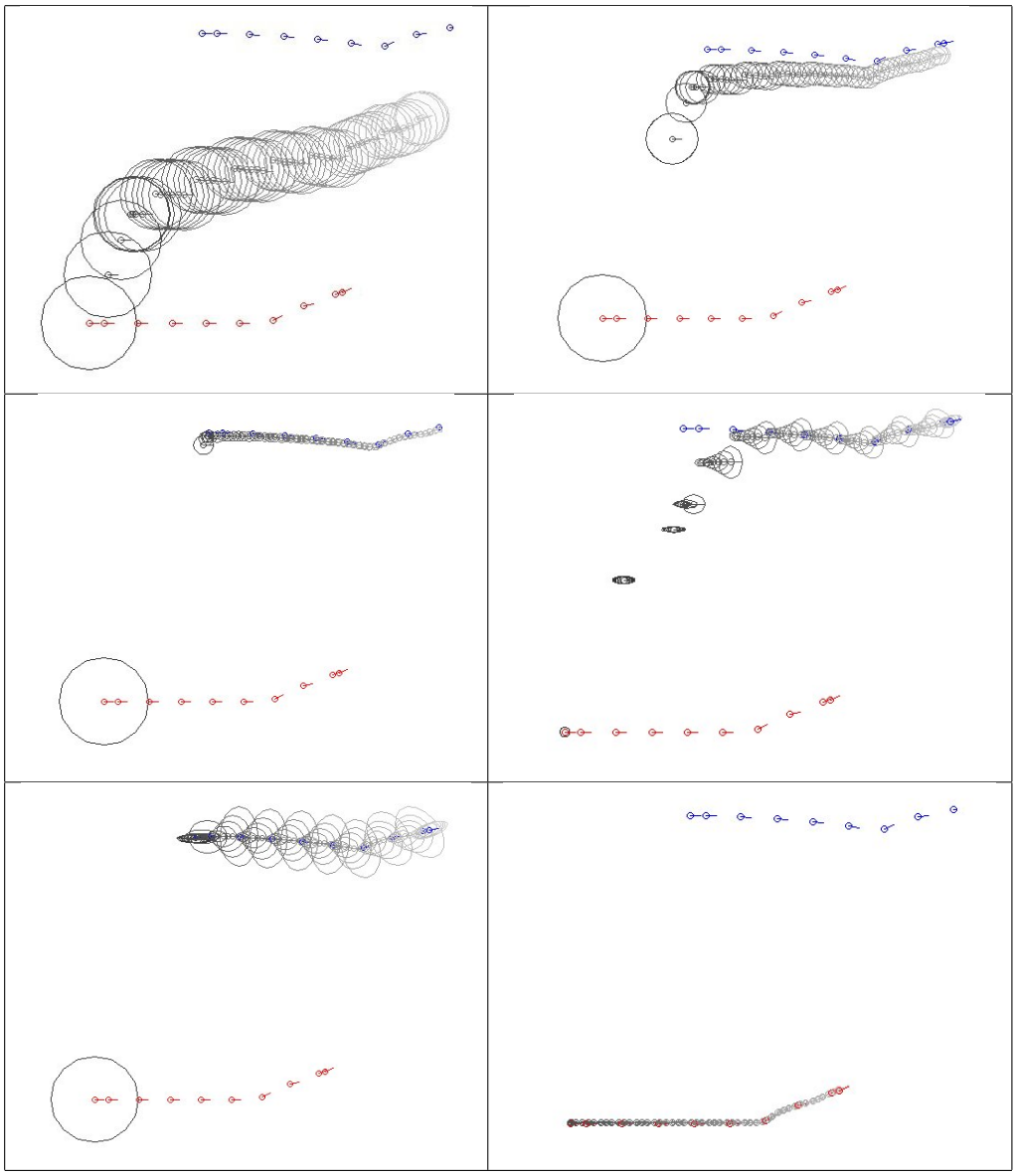

Figure 1: Six Kalman Trajectories (of `fr_sim.log`) with different parameter choices for Σ_0, M and Q. The red poses visualize odometry, the blue poses visualize the true pose (painted every 20th `truepose`-message. The black poses and ellipses represent the Kalman estimate every 10th logfile-message.