Filters and estimation [HW 4]

Special Topics in Robotics

Jakub Tomasek

November 22, 2013

Contents

1 First problem

1.1 Problem specification

I have a mobile robot for which it is possible to control forward and rotational velocity. The initial or current position of the robot is uncertain (e.g. due to noise and error of sensors). Robot moves forward with 100% throttle ($100\,\mathrm{m\,s^{-1}}$).

The task is to predict position of robot in the next step (step size T) assuming we perfectly know the applied force (i.e. no slipping wheels).

1.2 Solution

The system can be defined using difference equation:

$$\begin{aligned} x_{k+1} &= x_k + T u_k \cos\theta_k \\ y_{k+1} &= y_k + T u_k \sin\theta_k \\ \theta_{k+1} &= \theta_k + T w_k \end{aligned}$$

where x and y is the position of the robot in 2D coordinates, θ is rotation of the robot, and u and w are the control input, i.e. the forward velocity and angular velocity. I set

$T = 1\,\mathrm{s}$. There is no control noise (no error on motors). I set the state vector:

$$\begin{aligned} \mathbf{x}_k &= \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}, \\ \mathbf{x}_{k+1} &= f(\mathbf{x}_k). \end{aligned}$$

I assume that the initial distribution is normal with mean μ_0 and covariance matrix Σ_0:

$$\begin{aligned} \boldsymbol{\mu}_0 &= E\{\mathbf{x}_0\} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \\ \boldsymbol{\Sigma}_0 &= E\{\mathbf{x}_0\} = \begin{bmatrix} p_x & p_{xy} & p_{x\theta} \\ p_{xy} & p_y & p_{y\theta} \\ p_{x\theta} & p_{y\theta} & p_\theta \end{bmatrix}. \end{aligned}$$

I assume that x_k, y_k and θ_k are uncorrelated, thus $p_{xy} = p_{x\theta} = p_{y\theta} = 0$. But, in the next section I also discuss the correlated case. Firstly, I generate 1000 samples from this distribution; this yields a thousand initial points $\mathbf{x}_0$. Next I use $f(\mathbf{x}_0)$ to obtain predicted value $\mathbf{x}_1$. I used Matlab:

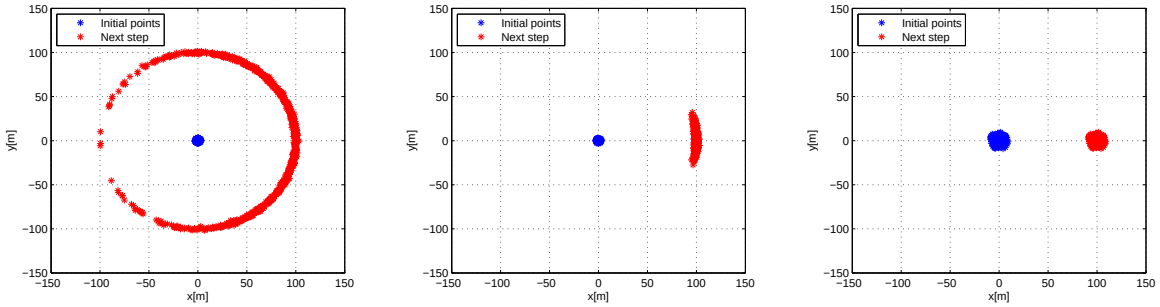
```
%number of points
n=1000
%mean
mu = [0 0 0];
%covariance matrix
Sigma = [1 0 0;0 1 0; 0 0 0.01];
%generate points
points = mvnrnd(mu,Sigma,n);

plot(points(:,1),points(:,2),'*');
nextStep=[points(:,1)+100*cos(points(:,3)),points(:,2)+100*sin(points
  (:,3)),points(:,3)]
hold on
plot(nextStep(:,1),nextStep(:,2),'r*');
hold off
```

1.3 Discussion

Figures 1.1, 1.2, and 1.3 show how the prediction changes with increasing variances of the initial states separately for uncorrelated case. Obviously, the problem is in the uncertainty of the angle. Points in the next step form a section of a circle with radius approximately 100. If there is a large error of the initial angle θ_0 they form a whole circle. Further Figures 1.4 and 1.5 show the case when the distribution is correlated.

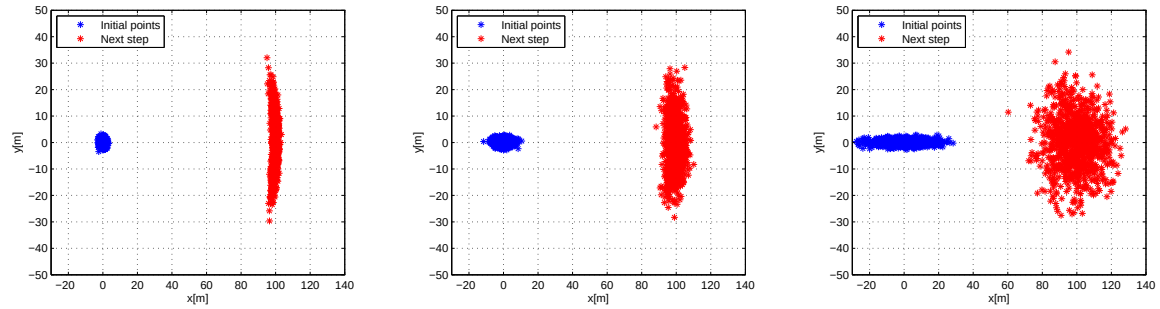
1.1 shows distribution for p_θ equal to 1 and 0.01. $p_\theta = 1$ corresponds to standard deviation 57.3°, $p_\theta = 0.01$ to 5.73°. Indeed, $p_\theta = 1$ is large and in the real world we probably won't have to tackle with such error. However, even for standard deviation 5.73° the distribution shape in the next step is highly nonlinear and thus hard to approximate by normal distribution used by widespread Kalman filter. Thus no, normal distribution in cartesian coordinate system doesn't well represent the distribution.

Yet, the solution for this appears to be simple and obvious from 1.1. We could remove the nonlinearity and still use the normal distribution by mapping our model to some other



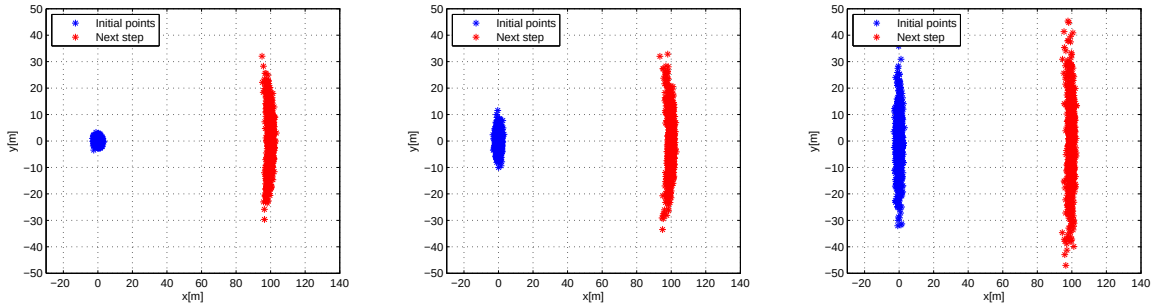
(a) $p_1 = 1$, $p_2 = 1$, $p_3 = 1$ (b) $p_1 = 1$, $p_2 = 1$, $p_3 = 0.01$ (c) $p_1 = 10$, $p_2 = 10$, $p_3 = 0$

Figure 1.1: Changing uncertainty of initial angle θ_0. Last figure shows no uncertainty of θ_0.



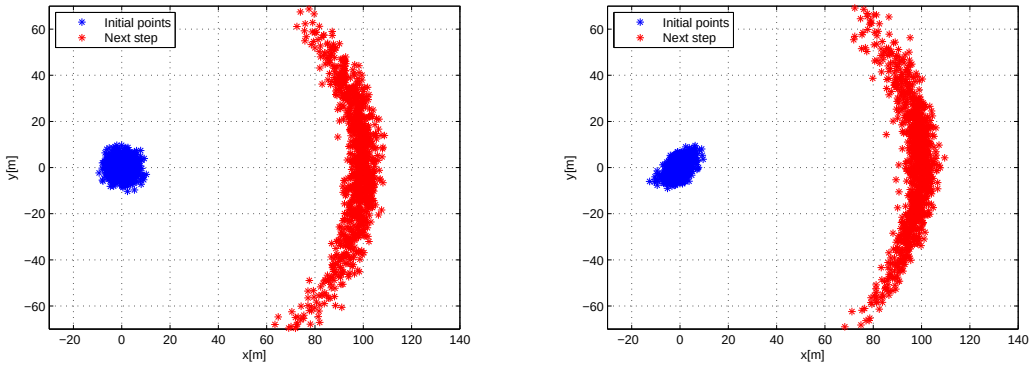
(a) $p_1 = 1$, $p_2 = 1$, $p_3 = 0.01$ (b) $p_1 = 10$, $p_2 = 1$, $p_3 = 0.01$ (c) $p_1 = 100$, $p_2 = 1$, $p_3 = 0.01$

Figure 1.2: Increasing uncertainty of x_0



(a) $p_1 = 1$, $p_2 = 1$, $p_3 = 0.01$ (b) $p_1 = 1$, $p_2 = 10$, $p_3 = 0.01$ (c) $p_1 = 1$, $p_2 = 100$, $p_3 = 0.01$

Figure 1.3: Increasing uncertainty of y_0



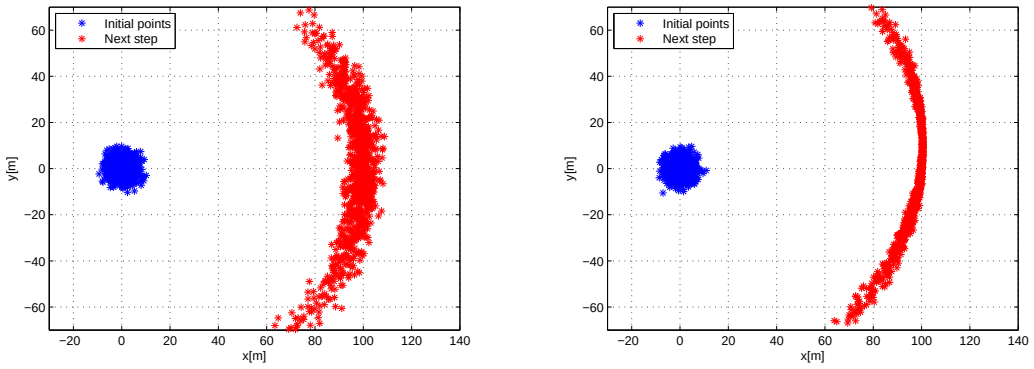
(a) $p_1 = 10$, $p_2 = 10$, $p_3 = 0.1$, $p_{xy} = 0$ (b) $p_1 = 10$, $p_2 = 10$, $p_3 = 0.1$, $p_{xy} = 5$

Figure 1.4: Crosscorrelation between x_0 and y_0

linear space. In our case we could transform the cartesian coordinates x and y to polar coordinates r and ϕ. This is called circular normal distribution.

How would I improve the modeling? Mainly, I would add "propelling" uncertainty, i.e. I would introduce noise into the state equation. Noise is random process and in the real world, unlike in the case of Gaussian noise, it is usually colored, i.e. it is more likely to be pronounced on some frequencies, typically high frequencies. This can be approximated using shaping filter with white-noise input. To model all this, I could use some more sophisticated methods like Kalman filter.

Secondly, we might consider other distribution than gaussian which would encompass other uncertainties. The position uncertainty is not necessarily best described by gaussian as it is discussed in [Thrun et al., 2005]. Gaussian noise well describes error which arises from the limited resolution of our sensors. Nevertheless, considering only the position measurement, it fails for other error types [Thrun et al., 2005]: unexpected objects, failures of measurement (e.g. black object for laser rangefinder), "phantom" readings. Further, in robotics we often tackle distribution with multimodal probabilities (localization problem). We might use Monte carlo methods to generate samples from almost arbitrary distribution in more dimensions.



(a) $p_1 = 10\,,\, p_2 = 10\,,\, p_3 = 0.1\,,\, p_{xy} = 0$ (b) $p_1 = 10\,,\, p_2 = 10\,,\, p_3 = 0.1\,, p_{x\theta} = 5$

Figure 1.5: Crosscorrelation between x_0 and θ_0

2 Second problem

2.1 Problem definition

The problem is taken from Bell and Cathey [1993]. We have two ranging stations located at $A* = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$ and $B* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, see Fig. 2.1. These two ranging stations can detect distance to an object, the problem is simplified into a plane, i.e. it is only 2D problem.

We have current state estimate (at time k), and the current measurement are given by

$$\hat{\mathbf{x}}_k = \begin{bmatrix} 0 \\ \beta \end{bmatrix}\,,\, \mathbf{z}_k = \begin{bmatrix} 1 \\ 1 \end{bmatrix}\,.$$

The covariance of $\hat{x}_k$ and z_k is P and R respectively:

$$\mathbf{P}_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\,,\, \mathbf{R}_k = \rho \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\,.$$

There ρ and β are variables I can choose and vary to demonstrate the function. The measurement is defined below by measurement model function in Eq. 2.2. Current real state is

$$\mathbf{x}_k = \begin{bmatrix} 0 \\ 1 \end{bmatrix}\,.$$

The ultimate goal is to find precise update $\hat{\mathbf{x}}_k^+$ current estimate of location $\hat{\mathbf{x}}_k$ based on measurement $\mathbf{z}_k$ using the iterated Kalman filter (IKF).

e

2.2 Solution

Measurement can be described using measurement-model function $h\left(\boldsymbol{\xi}\right)$. We define it this way:

$$\boldsymbol{\xi} = \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix}\,, \tag{2.1}$$

$$h\left(\boldsymbol{\xi}\right) = \frac{1}{2}\begin{bmatrix} d_1^2 \\ d_2^2 \end{bmatrix} = \begin{bmatrix} \left(\xi_1 + 1\right)^2 + \xi_2^2 \\ \left(\xi_1 - 1\right)^2 + \xi_2^2 \end{bmatrix}\,. \tag{2.2}$$

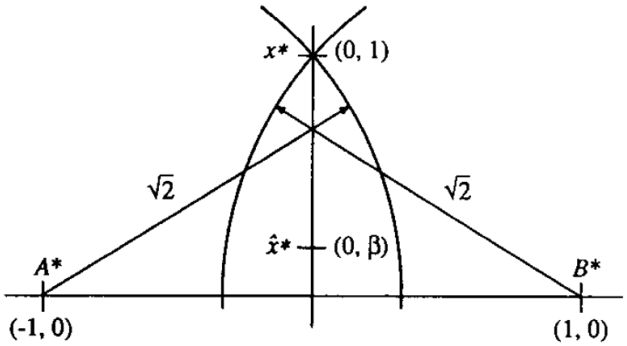


Figure 2.1: Problem definition. Two ranging stations at $A*$ and $B*$.

There $\boldsymbol{\xi}$ is state vector, i.e. ξ_1 and ξ_2 correspond with x and y position of the object. d_1 and d_2 are the distances from stations $A*$ and $B*$. Jacobian of h is

$$\mathbf{H} = h'(\boldsymbol{\xi}) = \begin{bmatrix} \frac{\partial h_1}{\partial \xi_1} & \frac{\partial h_1}{\partial \xi_2} \\ \frac{\partial h_2}{\partial \xi_1} & \frac{\partial h_2}{\partial \xi_2} \end{bmatrix} = \begin{bmatrix} \xi_1 + 1 & \xi_2 \\ \xi_1 - 1 & \xi_2 \end{bmatrix} .$$

Further I define $\mathbf{H}_i$:

$$\mathbf{H}_i = h'(\hat{\boldsymbol{x}}_i) .$$

The IKF update isBell and Cathey [1993]:

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{x}_0 + \mathbf{P}_{i+1}\mathbf{H}_i^T\mathbf{R}^{-1}\left[\mathbf{z} - h(\mathbf{x}_i) - \mathbf{H}_i(\mathbf{x}_0 - \mathbf{x}_i)\right] , \\ \mathbf{P}_{i+1} &= \left(\mathbf{H}_i^T\mathbf{R}^{-1}\mathbf{H}_i + \mathbf{P}^{-1}\right)^{-1} , \\ \mathbf{x}_0 &= \hat{\mathbf{x}}_k . \end{aligned}$$

The updated position given the measurement is obtained when the Gaussian Newton method converges:

$$\hat{\mathbf{x}}_k^+ = \mathbf{x}_i .$$

I implemented this in Matlab:

```
%% Iterated Kalman Filter
beta=0.8;
rho=0.01; %error of measurement = our belief in measurement
z=[1;1]; %measurement

H=@(x) [x(1)+1 x(2);x(1)-1 x(2)];
h=@(x) 1/2*[(x(1)+1)^2+x(2)^2;(x(1)-1)^2+x(2)^2];
ex=[0;beta];
P0=[1 0; 0 1];
R=rho*[1 0; 0 1];
x=ex

%% one iteration of Gauss Newton method

P=(H(x)'*R^-1*H(x)+P0^-1)^-1
x=ex+P*H(x)'*R^-1*[z-h(x)-H(x)*(ex-x)]

% alternative formulation we derived in class with same results
% x=x+(H(x)'*R^-1*H(x)+P0)^-1*(H(x)'*R^-1*(z-h(x))+P0^-1*(ex-x))
```

β	ρ	$\hat{\mathbf{x}}^+_{EKF}$	$\hat{\mathbf{x}}^+_{IKF}$
0.5	10	$\begin{bmatrix}0.54\\0\end{bmatrix}$	$\begin{bmatrix}0.54\\0\end{bmatrix}$
0.5	1	$\begin{bmatrix}0.75\\0\end{bmatrix}$	$\begin{bmatrix}0.80\\0\end{bmatrix}$
0.5	0.1	$\begin{bmatrix}1.13\\0\end{bmatrix}$	$\begin{bmatrix}0.98\\0\end{bmatrix}$
0.5	0.01	$\begin{bmatrix}1.23\\0\end{bmatrix}$	$\begin{bmatrix}0.98\\0\end{bmatrix}$
0.5	0.001	$\begin{bmatrix}1.25\\0\end{bmatrix}$	$\begin{bmatrix}1.00\\0\end{bmatrix}$

Table 1: Increasing the belief in the measurement doesn't much improve the accuracy of the EKF.

2.3 Discussion

In general, if the initial belief in measurement is high, i.e. the parameter ρ is low, no matter what the initial predictions were the algorithm converges almost to the correct state . In this case the prediction has only small weight. But of course, the result improves with better prediction. Additionally, if the measurement is unprecise and thus ρ is high, the prediction has much more weight. That is natural. Some of the results are shown in the Table 1.

Finally, the result is different for Kalman filter. As it was shown in the class, extended Kalman filter (EKF) is basically one-step iterated Kalman filter. Here, the initial conditions are very important no matter how precise the measurement is. Table shows compared updated predictions for EKF $\hat{\mathbf{x}}^+_{EKF}$ and IKF $\hat{\mathbf{x}}^+_{IKF}$. IKF always returned better results than EKF. The difference was pronounce for measurements with very high belief.For instance for $\beta = 0.5$, $\rho = 0.01$, the different for EKF was 23% while for IKF it was only 2%.

References

B.M. Bell and F.W. Cathey. The iterated kalman filter update as a gauss-newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297, 1993. ISSN 0018-9286. doi: 10.1109/9.250476. 5, 6

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2005. URL http://www.lavoisier.fr/livre/notice.asp?ouvrage=1218883. 4