Matlab Exercises

Lecture 5 – Bayesian tracking with Kalman Filters and Condensation

1) **Kalman Filter implementation**

Write a Matlab function that implements the two steps of Kalman Filter (prediction+correction):

A. Prediction: Given a Gaussian motion model with covariance matrix $\Lambda_w$, and linear matrix $A$, compute the predicted (prior) state $s^-$ and covariance matrix $S^-$

Inputs: Matrix $A$, covariance $\Lambda_w$, old posterior mean $s_{t-1}$ and covariance $S_{t-1}$

Outputs: new Prior mean and covariances, ($s^-, S^-$)

B. Correction: Given a Gaussian measurement model with covariance matrix $\Lambda_v$, and linear matrix $C$, compute the corrected (posterior) state $s_t$ and covariance matrix $S_t$

Inputs: Matrix $C$, covariance $\Lambda_v$, prior mean $s^-$ and covariance $S^-$

Outputs: new posterior mean and covariances, ($s_t, S_t$)

2) **Kalman Filter example**

With the previously implemented functions, now test the Bayesian tracker (Kalman) for the following case:

Suppose to have a random point moving on a 2D plane, with a random WNA motion:

$$s_t = As_{t-1} + w_t$$

with $\Lambda_w = \text{diag}(0,0,1,1)$ (the noise is only in acceleration, so it goes into the velocity equations, not in the position!)

Suppose the initial state is also not known, and has a prior probability distribution $P_0(s) = \text{Gauss}(0,10)$, that is: the initial state is all zero (2D pose+velocity) both with uncertainty $\sigma^2=10$.

The measurement $z$ is a position measurement: $z = Cs+v$, where $C = [I 0]$ is a 4x2 matrix that takes only the upper part of $s$ (i.e. the pose), plus a 2D measurement uncertainty $v=\text{Gauss}(0,1)$.

With the given model, do the following parallel things:

A. Simulate the random process:

- Give a random initial state $s_0$ according to $P_0$ (use the Matlab function $\text{randn()}$ to generate Gaussian random numbers)
- At time $t$, apply the motion model (1) by generating a random acceleration $w_t$, and
updating the real state $s_t$.
- At time $t$, simulate also the measurement $z_t = C s_t + v_t$ by generating random 2D Gaussian number $v$.

B. Apply the Kalman Filter:

- At time $0$, use only the correction function, with the prior knowledge: $s_{0^-} = [0,0]$, $S_{0^-} = \text{diag}(10,10,10,10)$
- At time $t$, use both prediction+correction functions developed in the previous exercise.

C. Compare the real state with the Kalman estimation (plot a graph of the state components in time: $x(t)$, $y(t)$ and $x,y$ velocity)
- At each time, compute the difference between the real state (simulated in A) and the estimated posterior state obtained in B (the pose only is sufficient)
- Plot the results on a 2D graph: the real trajectory $s_0,s_1,...$ and the estimated one (posterior), again only the pose.

3) Extended Kalman Filter: track a flying ball (DLR system)

Suppose to have two cameras (a stereo system), looking a ball thrown across the room.

The setup is the one described in Exercise 3-Lecture 4, as below indicated

The ball $p=(x,y,z)$ describes a parabolic trajectory during the flight, and its motion model can be described by a constant gravity acceleration towards the bottom ($-g$) + a small random component $w$ (e.g. air resistance in different points of the trajectory).

This motion (described in Lecture 4 - Slide 10) gives a probabilistic state model: $P(s_t|s_{t-1}) = \text{Gauss}(A s_{t-1} + C, B\Delta \omega B^T)$.

with
\[ A = \begin{bmatrix} I & I \Delta t \\ 0 & I \end{bmatrix} \quad B = \begin{bmatrix} I \Delta t^2 \\ I \Delta t \end{bmatrix} \quad C = \begin{bmatrix} -gL \Delta t^2 \\ -gL \Delta t \end{bmatrix} \]

\((A \ s_{t-1} + C)\) is the prediction of \(s_t\).
\(g = [0 \ 981 \ 0]\) is the gravity acceleration (y direction)
\(\Lambda_w = \text{diag}(0,0,0,1,1,1)\) is the covariance of motion noise (acceleration noise)
\(\Delta t = 0.1\) is the time sampling interval (10 frames/sec).

The state \(s\) is a \((3+3)\)-vector (position+velocity), and positions are measured in \(mm\).

The measurement \(z\) (solution of the other exercise) is the collection of two positions located on the two camera images:
\(z = (p_{s1}, p_{s2})\), which are 4 image coordinates \((x_1,y_1,x_2,y_2)\).

The measurement model, for a given hypothesis \(p\), gives an expected measurement
\[ p_{s1,\text{exp}}(p_{c1}) = \left( \frac{x_{c1}}{z_{c1}} f + \frac{r_x}{2}, \frac{y_{c1}}{z_{c1}} f + \frac{r_y}{2} \right) \]
\[ p_{s2,\text{exp}}(p_{c2}) = \left( \frac{x_{c2}}{z_{c2}} f + \frac{r_x}{2}, \frac{y_{c2}}{z_{c2}} f + \frac{r_y}{2} \right) \]
\[ p_{c1} = (x_{c1}, y_{c1}, z_{c1}) = T_1 p \quad p_{c2} = (x_{c2}, y_{c2}, z_{c2}) = T_2 p \]

where \(p_{c1}\) and \(p_{c2}\) are the coordinates of \(p\) in the two cameras (extrinsic transformations \(T_1, T_2\)), and \(p_{s1,\text{exp}}, p_{s2,\text{exp}}\) are the projections on the screens (intrinsic transformation: \(f, r_x, r_y\)).

The parameters for this example are the following ones:

\(T_1\): only translation to the left \(t_x = -100\)mm
\(T_2\): only translation to the right \(t_x = +100\)mm
\(f = 1000, r_x = 640\) pixels, \(r_y = 480\) pixels
\(\Lambda_v = \text{covariance of measurement noise} = I\) (1 pixel uncertainty)

The probabilistic measurement model is (nonlinear \(z_{\text{exp}} + \text{Gaussian}\)), therefore an Extended Kalman Filter can be used for Bayesian tracking.

A. Compute the Jacobian matrix \(J = \frac{\partial z_{\text{exp}}}{\partial s}\) at given hypothesis \(s\), (write a Matlab function returning \(J\), with input \(s\))

B. Implement the Extended Kalman Filter (equations in Lecture 5-Slide 19).

NOTE: the motion model is already linear, so the Jacobian is just \(A\).

C. Do a simulated experiment (real vs. estimated state), where the ball is thrown from the ground:

Real initial state \(p_0 = [0,0,0]\) with initial velocity \(v_0 = [0, 10, 10]\) (forward z, up y).
Initial state hypothesis: \(p_0 = [0,0,0], v_0 = [0,0,0]\) (no knowledge).
Perturbation of acceleration during the flight = Gaussian random \(w\), with covariance 1.
Run the EKF, and report the results as for the Kalman filter (trajectories).

4) **Particle Filters implementation**

Implement a basic Particle Filter, by defining the 3 functions:

A. Re-sample: given a N-particles set \((s^i,\pi^i)\), take a new particle set obtained by sampling N times between \((1,\ldots,N)\) (with evtl. repetitions) with probabilities \((\pi^1,\ldots,\pi^N)\).

B. Move: for every re-sampled particle, apply a motion model \(s_t = g(s_{t-1},w_t)\), for example a WNA motion with given acceleration covariance. For this purpose, generate a random acceleration for every particle (hypothesis).

C. Re-weight: give new weights \(\pi^i = P(z|s^i)\) to the moved particles, by using a Likelihood function \(P(z|s)\) given by the user.

5) **Particle Filters example**

Apply the Particle Filters implementation to the same example used for Kalman Filter; here the Likelihood is Gaussian: \(P(z|s) = \text{Gauss}(Cs,\Lambda_s)\).