Applied Signal Processing
Hand-In Problem 3: Target Tracking Using the Kalman Filter

November 2018

This hand-in problem concerns Kalman filtering. The project is individual. You should hand in solutions of the theoretical tasks as well as plots from the Matlab tasks. The code produced in Task 3 should pass the self-test function and by included in your report, as well as the value you set the `student_id` parameter to and the secret key given by the test function. Upload your report to Pingpong.

The Kalman filter can be used for many estimation tasks. A classical "Wiener-filter type" example is to recover a distorted and/or disturbed random signal. The difference to the Wiener filter is that the Kalman filter is time-varying. This means that it can take initial uncertainty into account in an optimal way, and also that the dynamics describing the signal do not have to be stationary. Like the Wiener filter, the Kalman filter gives an LMMSE (Linear Minimum Mean Square Error) estimate of a signal. In some cases there are better non-linear filters, but not if the signal and noise are jointly Gaussian distributed. The Kalman filter has therefore found a wide use in applications such as adaptive filtering, calibration and target tracking. In this problem you will apply the Kalman filter to track a moving target using 2D camera measurements.

The target is assumed to move freely in the xy-plane, and the position at time t is described by the variables $x(t),\ y(t)$. We will assume that the target is nominally moving at constant speed along a straight line, which means that $\ddot{x}(t) = \ddot{y}(t) = 0$ (i.e. zero acceleration). However, we will later allow for changes in velocity for when deriving the discrete time model.

The state of the target motion model is given by:

$$s(t) = \begin{bmatrix} s_1(t) \\ s_2(t) \\ s_3(t) \\ s_4(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ \dot{x}(t) \\ y(t) \\ \dot{y}(t) \end{bmatrix}$$

(the name $s(t)$ for the state vector is not conventional, but used because here $x(t)$ denotes the x-coordinate). The state equations are then

$$\begin{aligned} \dot{s}_1(t) &= s_2(t) \\ \dot{s}_2(t) &= 0 \\ \dot{s}_3(t) &= s_4(t) \\ \dot{s}_4(t) &= 0\,. \end{aligned}$$

In many applications, measurements come in discrete samples. It is then more practical to use a discrete-time model of the target motion.

Task 1 (Theory): Apply the finite-difference approximation

$$\dot{x}(t)\big|_{t=kT} \approx \frac{x(kT+T)-x(kT)}{T}\,,$$

where T is the sampling time, to the continuous-time state equations. Thus, derive a discrete-time state-space model of the form

$$s(k+1) = As(k) + w(k)\,,$$

where A is a 4×4 matrix (from now on we use $s(k)$ to denote discrete time, corresponding to $t = kT$). The symbol $w(k)$ in the equation above denote a zero mean discrete random noise vector, i.e. the discrete time process noise. The covariance matrix of the noise vector should be selected to model possible speed changes between the sampling instances but not influence the position states.

Assume that a camera-based sensor measures the xy-position of the target at discrete time k, with additive noise $v_x(k)$ and $v_y(k)$ respectively. Derive the measurement equation

$$z(k) = Cs(k) + v(k)\,,$$

where $z(k)$ is a 2×1 vector and C is 2×4 matrix.

A real-world target will of course not behave exactly according to our model. The usefulness of the Kalman filter stems from that it can give useful results even when the model is not perfect. In this project the target moves along piecewise straight lines.

Task 2 (Matlab): The x and y coordinates (in discrete time) are already implemented in `hip3.m`. Run the file to see a plot of the true and measured position. In this assignment there is a lot of noise present to illustrate the potential of the Kalman filter. In practice, the measurement noise would be much smaller.

Task 3 (Matlab): Complete the MATLAB-function described in the file `student_sols.m`. See the lecture notes for hints on how to do this. Remember to include the generated secret key and `student_id` parameter in your submission.

Task 4: (Matlab) Apply your MATLAB-implementation of the Kalman filter to the target tracking data. Set the sampling time $T = 0.01$s. The input to the Kalman filter is the noisy measurements `Z`. Try to figure out suitable values for Q and R from how the true data is generated. Use the zero vector as initial state vector, and $P_0 = 10^6 \times \mathbf{I}$ as initial state covariance matrix. Investigate how the algorithm behavior changes if you increase/decrease R by a factor of 10. Try to fine-tune the performance of the algorithm by testing different scaled values of R (you can verify that increasing R by a factor of 10 has the same effect as decreasing Q and P_0 by the same factor). Plot the estimated target trajectory: `scatter(Xfilt(1,:),Xfilt(3,:))`. Also plot the estimated speeds in the x and y directions (versus time). Imagine how a direct finite-difference approximation of the measurements would have looked like and be amazed by the power of the Kalman filter!