JKalman

Java Kalman Filter

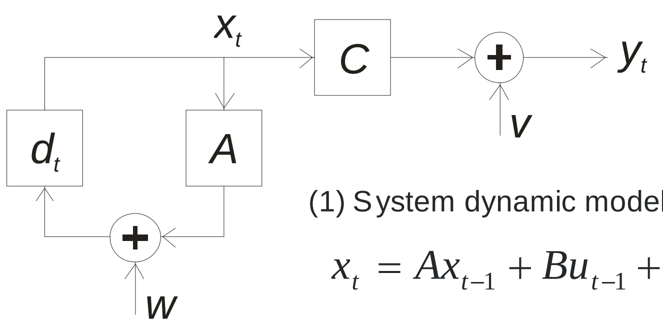
Petr Chmelar

FIT, Brno Univesity of Technology

Teoretical Background

The Kalman Filter was originally designed to represents information about a moving objects. Moving objects are modeled using a discrete time dynamic system. An object has spatio-temporal state x_t. The spatio-temporal state is represented by location [x, y, t] and velocity [d_x, d_y]. Here [x, y] is referred to as 2D position [x, y] of the object at the time t.

Data is supposed to be uncertain – noisy, some states might be missing and other are biased. At the moment, the object state x_t is characterized by a state vector *x*, containing position and velocity vector [x, y, d_x, d_y]. In that way, the trajectory is constructed (in 2D). However it cannot be observed directly, because it is encumbered by hidden (Gaussian) noise *w*. The system produces visible output vector *y* that is a simple linear observation (*x* evolves first-order Markov process), encumbered by noise *v*. At time *t*, the model can be written as in the Figure 1.



(1) System dynamic model

$$x_t = Ax_{t-1} + Bu_{t-1} + w \qquad w \approx N(0, Q)$$

(2) Measurement model

$$y_t = Cx_t + v \qquad v \approx N(0, R)$$

Figure 1. The dynamic process illustration.

In the Figure 1, d_t is a delay, *A* is a state transition matrix; *B* the optional control matrix with input *u*, *C* is the observation or measurement matrix. *N* is a normal distribution of mean 0, the (covariance) matrices of *w* and *v* are *Q* and *R*, respectively. Based on Bayesian probability $P(x_t \mid y_t) \approx N(\hat{x}_t, P_t)$, as described below.

An efficient computational (recursive) tool to estimate the state of a process, in a way that minimizes the mean of error, is the Kalman filter.

Kalman Filter

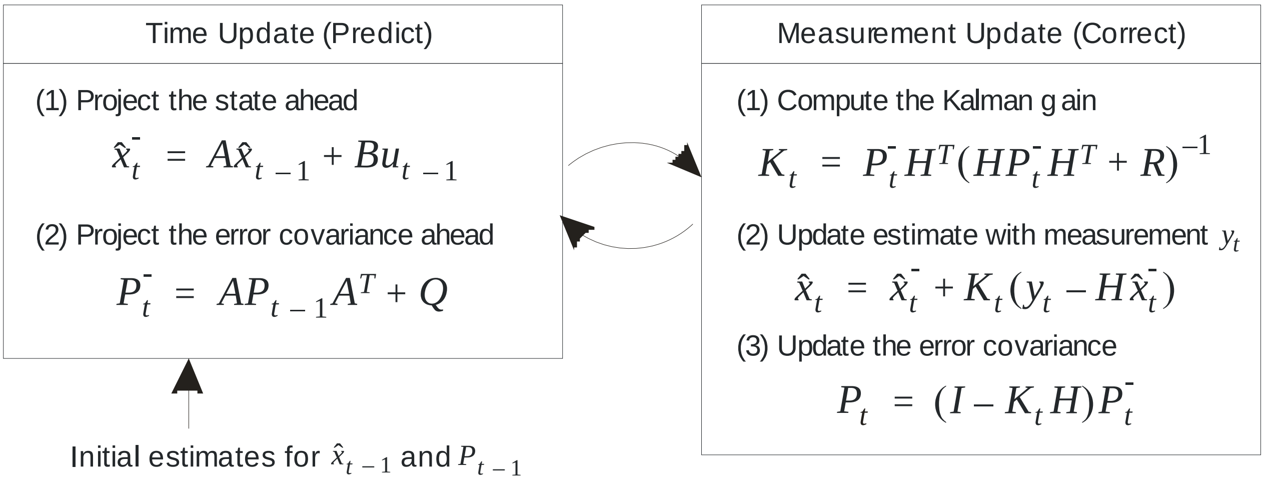


Figure 2. Operation of Kalman filter, combining the high-level diagram with the equations.

The Kalman filter iteratively applies two stages of computations using feedback control (see Figure 2):

1. Time update computations (prediction). Here xˆ-t is the prior state estimate (predicted) at time t, xˆt is the posterior (corrected) state estimation at the time (xˆt = E[xk]), P-t-1 is the prior estimate error covariance matrix and Pt-1 is the posterior error covariance matrix (Pt = E[(xt - xˆt)(xt - xˆt)T]).

2. Measurement update computations (correction). There H is a measurement matrix and Kt is a Kalman gain matrix. The weighting by K minimizes the posterior error covariance, while R approaches zero, the actual measurement yt is trusted more to the detriment of measurement prediction Hxˆ-t.

The detailed explanation is provided in:

G. Welch, G. Bishop, “An Introduction to the Kalman Filter”, 2006, http://www.cs.unc.edu/~welch/kalman/.
S. Roweis, Z. Ghahramani, “An Unifying Review of Linear Gaussian Models”, Neural Computation, 1999.

JKalman Java Implementation

The library can be found in "dist" directory.

The souce documentation for the JKalman library is included in "javadoc" directory.

Use "test.KalmanTest.java" as intoduction to JKalman development.