Vector Kalman Filters

Abhiraj Singh Bhal (Y5011) and Varun Jain (Y5494)

Department of Electrical Engineering, Indian Institute of Technology Kanpur

Abstract

The Kalman filter is an efficient recursive filter that estimates the state of a linear dynamic system from a series of noisy measurements. It is used in a wide range of engineering applications from radar to computer vision, and is an important topic in control theory and control systems engineering. In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem [1].

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown [2].

This term paper gives an introduction to Vector Kalman filters. It includes a description and some discussion of the basics of Vector Kalman filters, a derivation, description and some discussion of the extended Kalman filter, and a small example of its applications.

Discrete Kalman Filter

The Process to be Estimated

The Kalman filter addresses the general problem of trying to estimate the state $\mathbf{x} \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} + \boldsymbol{B}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1}, \tag{1.1}$$

with a measurement $\mathbf{z} \in \Re^m$ that is

$$z_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k \tag{1.2}$$

The random variables $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ represent the process and measurement noise respectively. They are assumed to be independent (of each other), white, and with normal probability distributions

$$\boldsymbol{p}(\boldsymbol{w}) \sim \boldsymbol{N}(\mathbf{0}, \boldsymbol{Q}) \tag{1.3}$$

$$\boldsymbol{p}(\boldsymbol{v}) \sim \boldsymbol{N}(\mathbf{0}, \boldsymbol{R}) \tag{1.4}$$

In practice, the *process noise covariance* $\boldsymbol{Q}$ and *measurement noise covariance* $\boldsymbol{R}$ matrices might change with each time step or measurement, however here we assume they are constant.

The n × n matrix $\boldsymbol{A}$ in the difference equation (1.1) relates the state at the previous time step *k-1* to the state at the current step *k*, in the absence of either a driving function or process noise. In practice $\boldsymbol{A}$ might change with each time step, but here we assume it is constant. The matrix $\boldsymbol{B}$

relates the optional control input $\boldsymbol{u} \in \Re^l$ to the state $\boldsymbol{x}$. The m × n matrix $\boldsymbol{H}$ in the measurement equation (1.2) relates the state to the measurement z_k. In practice $\boldsymbol{H}$ might change with each time step or measurement, but here we assume it is constant [2].

Computational Origin of the Filter

We define $\hat{\boldsymbol{x}}^-_k \in \Re^n$ **("super minus")** to be our *a priori* state estimate at step k given knowledge of the process prior to step k, and $\hat{\mathbf{x}}_k \in \Re^n$ to be our *a posteriori* state estimate at step k given measurement z_k. We can then define *a priori* and *a posteriori* estimate errors as

$$\boldsymbol{e}^-_k \equiv \boldsymbol{x}_k - \hat{\boldsymbol{x}}^-_k, \text{ and}$$

$$\boldsymbol{e}_k \equiv \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k$$

The a priori estimate error covariance is then

$$\boldsymbol{P}^-_k = \boldsymbol{E}\left[\boldsymbol{e}^-_k \boldsymbol{e}^{-T}_k\right] \tag{1.5}$$

and the *a posteriori* estimate error covariance is

$$\boldsymbol{P}_k = \boldsymbol{E}\left[\boldsymbol{e}_k \boldsymbol{e}_k^T\right] \tag{1.6}$$

In deriving the equations for the Kalman filter, we begin with the goal of finding an equation that computes an *a posteriori* state estimate as a linear combination of an *a priori* estimate and a weighted difference between an actual measurement and a measurement prediction as shown below in (1.7).

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}^-_k + \boldsymbol{K}\left(z_k - \boldsymbol{H}\hat{\boldsymbol{x}}^-_k\right) \tag{1.7}$$

The $(z_k - \boldsymbol{H}\hat{\boldsymbol{x}}^-_k)$ difference in (1.7) is called the measurement *innovation*, or the *residual*. The residual reflects the discrepancy between the predicted measurement $\boldsymbol{H}\hat{\boldsymbol{x}}^-_k$ and the actual measurement z_k. A residual of zero means that the two are in complete agreement.

The n × m matrix $\boldsymbol{K}$ in (1.7) is chosen to be the *gain* or *blending factor* that minimizes the *a posteriori* error covariance (1.6). This minimization can be accomplished by first substituting (1.7) into the above definition for $\boldsymbol{e}_k$, substituting that into (1.6), performing the indicated expectations, taking the derivative of the trace of the result with respect to $\boldsymbol{K}$, setting that result equal to zero, and then solving for $\boldsymbol{K}$ [3], [4], [5]. One form of the resulting $\boldsymbol{K}$ that minimizes (1.6) is given by

$$\boldsymbol{K}_k = \boldsymbol{P}^-_k \boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}^-_k \boldsymbol{H}^T + \boldsymbol{R}\right)^{-1} \tag{1.8}$$

Looking at (1.8) we see that as the measurement error covariance $\boldsymbol{R}$ approaches zero, the gain $\boldsymbol{K}$ weights the residual more heavily. Specifically,

$$\lim_{\mathbf{R}\to\mathbf{0}} \boldsymbol{K}_k = \boldsymbol{H}^{-1}$$

On the other hand, as the *a priori* estimate error covariance $\boldsymbol{P}^-_k$ approaches zero, the gain $\boldsymbol{K}$ weights the residual less heavily. Specifically,

$$\lim_{\boldsymbol{P}^-_k \to 0} \boldsymbol{K}_k = \boldsymbol{0}$$

Another way of thinking about the weighting by $\boldsymbol{K}$ is that as the measurement error covariance $\boldsymbol{R}$ approaches zero, the actual measurement z_k is "trusted" more and more, while the predicted

measurement $\boldsymbol{H}\hat{\boldsymbol{x}}^{-}_{k}$ is trusted less and less. On the other hand, as the *a priori* estimate error covariance $\boldsymbol{P}^{-}_{k}$ approaches zero the actual measurement z_k is trusted less and less, while the predicted measurement $\boldsymbol{H}\hat{\boldsymbol{x}}^{-}_{k}$ is trusted more and more.

The Discrete Kalman Filter Algorithm

The Kalman filter estimates a process by using a form of feedback control - the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback, that is, for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations. Indeed the final estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems. The specific equations for the time and measurement updates are presented below in Table 1-1 and Table 1-2

Table 1-1: Discrete Kalman filter time update equations.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{1.9}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{1.10}$$

The time update equations in Table 1-1 project the state and covariance estimates forward from time step *k -1* to step *k*. $\boldsymbol{A}$ and $\boldsymbol{B}$ are from (1.1), while $\boldsymbol{Q}$ is from (1.3).

Table 1-2: Discrete Kalman filter measurement update equations.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{1.11}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{1.12}$$

$$P_k = (I - K_k H)P_k^- \tag{1.13}$$

The first task during the measurement update is to compute the Kalman gain, $\boldsymbol{K}_k$. Notice that the equation given here as (1.11) is the same as (1.8). The next step is to actually measure the process to obtain z_k, and then to generate an *a posteriori* state estimate by incorporating the measurement as in (1.12). Again (1.12) is simply (1.7) repeated here for completeness. The final step is to obtain an *a posteriori* error covariance estimate via (1.13).

After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. This recursive nature is one of the very appealing features of the Kalman filter—it makes practical implementations much more feasible than (for example) an implementation of a Wiener filter [4] which is designed to operate on *all* of the data *directly* for each estimate. The Kalman filter instead recursively conditions the current estimate on all of the past measurements. Figure 1 below offers a complete picture of the operation of the filter,

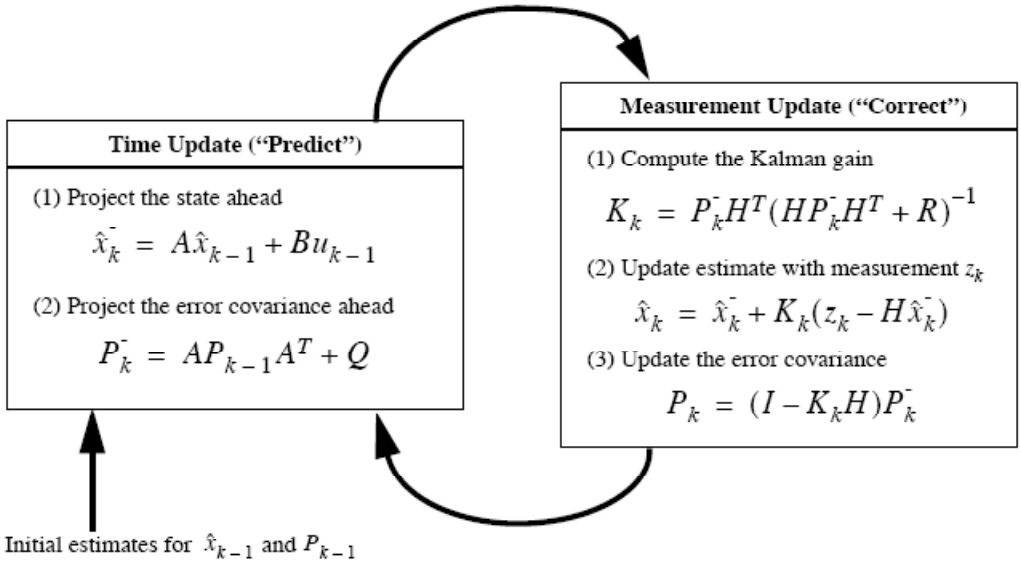

Figure 1 - A complete picture of the operation of the Kalman filter

Extended Kalman Filter

The Process to be Estimated

The general discrete Kalman filter addresses the problem of estimating the state $\mathbf{x} \in \Re^n$ of a discrete time controlled process governed by a liner stochastic difference equation. However, if this underlying equation is non-linear, a Kalman filter that linearizes about the current mean and covariance is referred to as an *extended Kalman filter* or EKF.

State $\mathbf{X} \in \Re n$ of a discrete-time controlled process given by the non linear stochastic equation,

$$\mathbf{X}_k = \mathbf{f}(\mathbf{X}_k - 1, \mathbf{U}_k - 1, \mathbf{W}_k - 1) \quad (2.1)$$

with measurement $\mathbf{Z} \in \Re m$ that is

$$\mathbf{z}_k = \mathbf{h}(\mathbf{X}_k, \mathbf{V}_k) \quad (2.2)$$

Approximate the state and measurement vector without the noise values as

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \quad (2.3)$$

$$\tilde{z}_k = h(\tilde{x}_k, 0) \quad (2.4)$$

where $\hat{x}_k$ is some *a posteriori* estimate of the state (from a previous time step *k*).

It is important to note that a fundamental flaw of the EKF is that the distributions (or densities in the continuous case) of the various random variables are no longer normal after undergoing their respective nonlinear transformations. The EKF is simply an *ad hoc* state estimator that only approximates the optimality of Bayes' rule by linearization. Some interesting work has been done by Julier et al. in developing a variation to the EKF, using methods that preserve the normal distributions throughout the non-linear transformations [6].

Computational Origins of the Filter

To estimate a process with non-linear difference and measurement relationships, we begin by writing new governing equations that linearize an estimate about (2.3) and (2.4),

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1} \quad (2.5)$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + Vv_k \quad (2.6)$$

where

- x_k and z_k are the actual state and measurement vectors,
- $\tilde{x}_k$ and $\tilde{z}_k$ are the approximate state and measurement vectors from (2.3) and (2.4),
- $\hat{x}_k$ is an *a posteriori* estimate of the state at step k,
- The random variables w_k and v_k are the process and measurement noise.
- *A* is the Jacobian matrix of partial derivatives of with respect to *x*,
- *W* is the Jacobian matrix of partial derivatives of with respect to *w*,
- *H* is the Jacobian matrix of partial derivatives of with respect to *x*,
- *V* is the Jacobian matrix of partial derivatives of with respect to *v*.

Now we define the prediction error and the measurement residual as –

- Prediction error $\tilde{e}_{xk} \equiv x_k - \tilde{x}_k$ (2.7)
- Measurement Residual $\tilde{e}_{zk} \equiv z_k - \tilde{z}_k$ (2.8)
- Error Process
  - $\tilde{e}_{xk} \approx A(x_{k-1} - \hat{x}_{k-1}) + \varepsilon_k$ (2.9)
  - $\tilde{e}_{zk} \approx H\tilde{e}_{xk} + \eta_k$ (2.10)

where ε_k and η_k represent new independent random variables having zero mean and covariance matrices WQW^T and VRV^T, with Q and R as in (1.3) and (1.4) respectively.

The *a posteriori* state estimates for the original non-linear process is given by

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k \tag{2.11}$$

The random variables of (2.9) and (2.10) have the following probability distributions

$$p(\tilde{e}) \sim N(0, E[\tilde{e}\tilde{e}^T])$$
$$p(\varepsilon_k) \sim N(0, WQ_kW^T)$$
$$p(\eta_k) \sim N(0, VR_kV^T)$$

With these approximations and letting $\hat{e}_k$ be zero

$$\hat{e}_k = K_k\tilde{e}_{zk} \tag{2.12}$$

By substituting (2.12) in (2.11) and using (2.8), we avoid the use of the hypothetical Kalman Filter and get

$$\hat{x}_k = \tilde{x}_k + K_k\tilde{e}_{zk} = \tilde{x}_k + K_k(z_k - \tilde{z}_k) \tag{2.13}$$

Extended Kalman Filter Algorithm

The algorithm of the Extended Kalman filter closely follows that of the discrete Kalman filter. Figure 2 below offers a complete picture of the operation of the EKF.

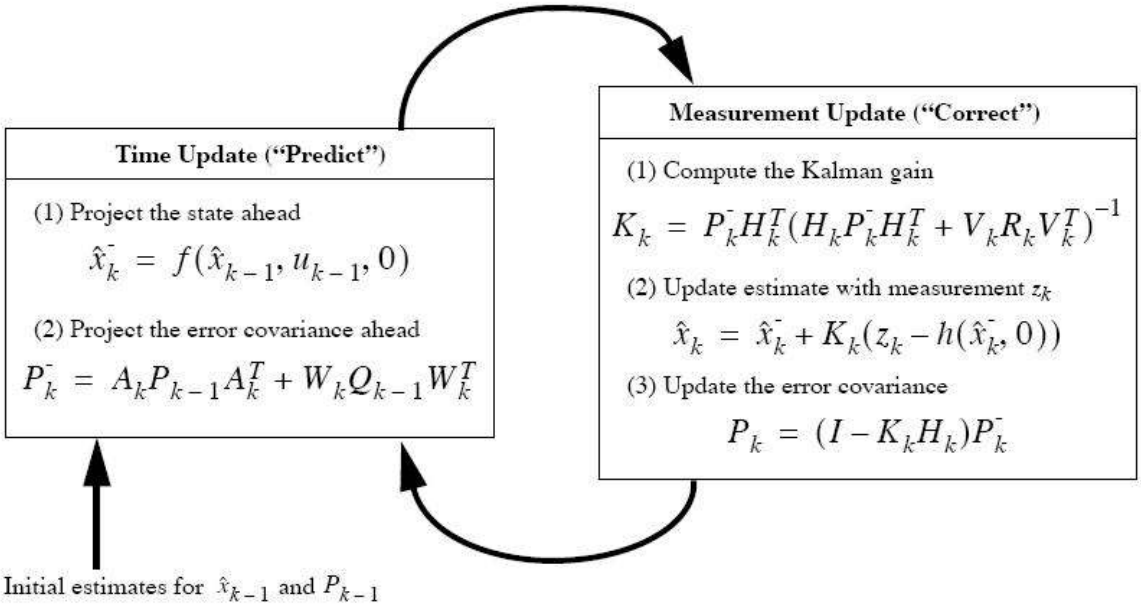

Figure 2: Complete Operation of the Extended Kalman Filter

An important feature of the EKF is that the Jacobian H_K in the equation for the Kalman K_K gain serves to correctly propagate or "magnify" only the relevant component of the measurement information. For example, if there is not a one-to-one mapping between the measurement z_K and the state via h, the Jacobian H_K affects the Kalman gain so that it only magnifies the portion of the residual $z_K - h(\hat{x}_k, 0)$ that does affect the state. Of course if over *all* measurements there is

not a one-to-one mapping between z_K the measurement and the state via h, then as you might expect the filter will quickly diverge. In this case the process is *unobservable.*

Kalman Filter Example – RoboCup Soccer

The challenge of the RoboCup soccer team developers is to develop a robotic soccer team by 2050 which can defeat the then world cup champions. Kalman filters have been used to estimate the position and orientation of the robot in these teams [7].

The robot position is modelled by a Kalman Filter and the ball position is also modelled by a Kalman Filter. The robot position is determined by the x and y coordinates on the field and the orientation. There are six uniquely identifiable beacons and four goal edges on the RoboCup soccer field for which the coordinates are known before runtime, allowing them to be used as a reference for position on the field. To localise, the robot uses information from vision about the location of beacons and goals with respect to the robot and information about the amount of movement. The robot localization routine is called every time a vision update is received which is approximately twenty five times per second. As the measurements of distance and angle are not a linear function of x, y and orientation, the Extended Kalman Filter is used. Initially the ball's position was computed using a simple trigonometric calculation on the information from vision. The errors in calculating the position of the ball using this method are rather high, so a Kalman Filter was implemented on the ball to eliminate some of these errors. The ball position is determined by x and y coordinates, but the Kalman Filter also includes velocity estimation in the x and y directions. The velocity states allow prediction of the ball's position to be made when the robot cannot see the ball. Negative acceleration of the ball due to friction on the field is taken into account. For the ball there is no knowledge of the ball's input driving force (robots kicking the ball, ball bouncing of wall edges) so the only information that is used is the vision information of distance and angle to the ball. If communication between the robots is possible via the wireless LAN link it is advantageous for them to share information, especially about where the ball is. This is particularly useful if one robot cannot see the ball. In order to share information a Ka lman Filter is implemented on ball position measurements from all robots. For this Kalman Filter, each robot calculates the (x, y) position of the ball using trigonometry and sends this information to all the other robots. These x and y coordinates are treated as the 'measurements' and since the measurements are a linear function of the variables we wish to know, a linear Kalman Filter is used. All the measurements from all the robots are then used in a Kalman Filter. As there were some problems with the ball position "jumping" as measurements from different robots were put into it, two ball position Kalman Filters were implemented - one with measurements from all the robots, and one with measurements from one robot. The one robot ball position was used in order to position for a kick, and the multiple robot ball position was used when the robot could not see the ball.

References

[1] R. E. Kalman, "A New Approach to Linear Filtering and Reference Problems", Transactions of the ASME–Journal of Basic Engineering, 82 (Series D): 35-45.

[2] G. Welch and G. Bishop, "An Introduction to Kalman Filters", Department of Computer Science, University of North Carolina at Chapel Hill.

[3] Maybeck, Peter S. 1979. *Stochastic Models, Estimation, and Control, Volume 1*, Academic Press, Inc.

[4] Brown, R. G. and P. Y. C. Hwang. 1992. *Introduction to Random Signals and Applied Kalman Filtering, Second Edition*, John Wiley & Sons, Inc.

[5] Jacobs, O. L. R. 1993. *Introduction to Control Theory, 2nd Edition*. Oxford University Press.

[6] Julier, Simon and J. Uhlman. "A General Method of Approximating Nonlinear Transformations of Probability Distributions," Robotics Research Group, Department of Engineering Science, University of Oxford [cited 14 November 1995].

[7] L. Freeston, "Applications of the Kalman Filtering Algorithm to Robot Localisation and World Modeling", University of New Castle, NSW, Australia.