Extended Kalman Filter

Dhruv Patel July 20, 2020

Why Extended Filter is required ?

If we already have Kalman Filter, first question in mind would be why do we require another alternative to that ... ?

Well, in real life, the assumptions of *linear state transitions* and linear measurements with added *Gaussian noise* are only found on the rarest of occasions. Let's see what I mean by that using an example. " A robot that moves with constant translational and rotational velocity typically moves on a circular trajectory, which cannot be described by linear next state transitions. This fact, along with the assumption of unimodal beliefs, renders the simple Kalman Filter adequate enough for most trivial robotics problems. This Extended Kalman Filter (EKF) overcomes one of these assumptions : the linearity assumption. Here, the assumption is that the next state probability and the measurement probabilities are governed by nonlinear functions g and h, respectively :

$$x_t = g(u_t,\ x_{t-1}) + \varepsilon_t \tag{1}$$

$$z_t = h(x_t) + \delta_t \tag{2}$$

This model strictly generalizes the linear Gaussian model underlying Kalman filters, postulated in Equations 1 and 2. The function g replaces the matrices A_t and B_t in previous filter type, while h replaces the matrix C_t. **Unfortunately, with arbitrary functions g and h, the belief is no longer a Gaussian**. In fact, performing the belief update exactly is usually impossible for non-linear functions g and h, in the sense that the Bayes filter does not possess a closed-form solution.

So how come this EKF performs better than Kalman Filter?

The extended Kalman filter (EKF) calculates an approximation to the true belief. Particularly, the belief $bel(x_t)$ at time t is represented by a mean μ_t and a covariance Σ_t. Thus, the EKF inherits from the Kalman filter the basic belief representation, but it differs in that this belief is only approximate, not exact as was the case in Kalman filters.

The EKF Algorithm

The algorithm, in many ways, is similar to the KF.

---

1 Algorithm Kalman Filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
2 # Prediction Step : ref. as Motion Model
3 $\overline{\mu}_t = g(u_t, \mu_{t-1})$
4 $\overline{\Sigma}_t = G_t * \Sigma_{t-1} * G_t^T + R_t$
5 # Update Step : ref. as Measurement Model
6 $K_t = \overline{\Sigma}_t * H_t^T * (H_t * \overline{\Sigma}_t * H_t^T + Q_t)^{-1}$
7 $\mu_t = \overline{\mu}_t + K_t * (z_t - h(\mu_t))$
8 $\Sigma_t = (I - K_t * H_t) * \overline{\Sigma}_t$
9 return μ_t, Σ_t

Table 1: Comparison - Kalman Filter vs EKF

	Kalman Filter	**EKF**
state prediction (line 2)	$A_t * \mu_{t-1} + B_t * u_t$	$g(u_t, \mu_{t-1})$
measurement prediction (line 5)	$C_t * \overline{\mu}_t$	$h(\overline{\mu}_t)$

In summary, the linear predictions in Kalman filters are replaced by their nonlinear generalizations in EKF. Also, EKF uses Jacobians G_t and H_t instead of the corresponding linear system matrices A_t, B_t and C_t in Kalman counterpart. The Jacobian G_t corresponds to the matrices A_t and B_t, while the Jacobian H_t corresponds to C_t.

The Kalman Filter represents beliefs by the moments parameterization : At time t, the belief $bel(x_t)$is represented by the mean μ_t and the covariance Σ_t. The input of the Kalman filter is the belief at time $t-1$, represented by μ_{t-1} and Σ_{t-1}. To update these parameters, Kalman filters require the control u_t and the measurement z_t. The output is the belief at time t, represented by μ_t and Σ_t. This predicted belief μ_t and Σ_t is calculated representing the belief $\overline{bel}(x_t)$ one time step later, but before incorporating the measurement z_t. This belief is obtained by incorporating the control u_t. The update of the covariance considers the fact that states depend on previous states through the linear matrix A_t. This matrix is multiplied twice into the covariance, since the covariance is a quadratic matrix.

Considerations

The EKF has become just about the most popular tool for state estimation in robotics. Its strength is its simplicity and in its efficiency.

The reason for the computational efficiency is the fact that it (EKF) represents the belief by a multivariate Gaussian distribution. A Gaussian is a unimodal distribution, which can be thought of as a single guess, illustrated as an uncertainty ellipse. In many practical situations, Gaussians are robust estimators. EKF has been applied with great success to a number of state estimation problems that violate the underlying assumptions.

An important limitation : It approximates state transitions and measurements using Linear Taylor expansions. In most of the robotics problems, these functions are nonlinear. The goodness of this approximation depends on two main factors. First, it depends on the degree of nonlinearity of the functions that are being approximated. If these functions are approximately linear, the EKF approximation may generally be a good one, and EKF may approximates the posterior belief with sufficient accuracy. However, sometimes, the functions are not only nonlinear, but are also multi-modal, in which case the linearization may be a poor approximation. The goodness of the linearization also depends on the degree of uncertainty. The less certain the robot, the wider its Gaussian belief, and the more it is affected by nonlinearities in the state transition and measurement functions. In practice, when applying EKF, it is therefore important to keep the uncertainty of the state estimate small.

Note : Taylor series expansion is only one way to linearize. Two other approaches have often been found to yield superior results. One is the *unscented Kalman filter*, which probes the function to be linearized at selected points and calculates a linearized approximation based on the outcomes of these probes. Another is known as moments matching, in which the linearization is calculated in a way that preserves the true mean and the true covariance of the posterior distribution (which is not the case for EKF). Both techniques are relatively recent but appear to be superior to the EKF linearization.

The belief $\overline{bel}(x_t)$ is subsequently transformed into the desired belief $bel(x_t)$, by incorporating the measurement z_t. The variable K_t, is called *Kalman Gain*. It specifies the degree to which the measurement is incorporated into the new state estimate. The key concept here is the *innovation*, which is the difference between the actual measurement z_t and the expected measurement C_t. Finally, the new covariance of the posterior belief is calculated, adjusting for the information gain resulting from the measurement.

The *Kalman Filter* is computationally quite efficient. In many applications - such as the robot mapping applications - the measurement space is much lower dimensional than the state space, and the update is dominated by the $O(n^2)$ operations.

A brief explanation about the variables

The state transition probability $p(x_t|u_t, x_{t-1})$ must be a linear function in its arguments with added Gaussian noise. This is expressed by the following equation :

$$x_t = A_t * x_{t-1} + B_t * u_t + \varepsilon_t \tag{3}$$

Here, x_t and x_{t-1} are state vectors, and u_t is the control vector at time t. These vectors are column vectors. They are of the form

$$x_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ . \\ . \\ x_{n,t} \end{pmatrix} \text{ and } u_t = \begin{pmatrix} u_{1,t} \\ u_{t,2} \\ . \\ . \\ u_{m,t} \end{pmatrix} \tag{4}$$

A_t and B_t are matrices. A_t is a square matrix of size $n \times n$, where n is the dimension of the state vector x_t. B_t is of size $n \times m$, with m being the dimension of the control vector u_t. By multiplying the state and control vector with the matrices A_t and B_t, respectively, the state transition function becomes *linear* in its arguments. Thus, Kalman Filters assume linear system dynamics.

The random variable ε_t in (1) is a Gaussian random vector that models the uncertainty introduced by the state transition. It is of the same dimension as the state vector. It has zero mean, and its covariance will be denoted by R_t. A state transition probability of the form (1) is called a *linear Gaussian*, to reflect the fact that it is linear in its arguments with additive Gaussian noise.

The measurement probability $p(z_t|x_t)$ must also be *linear* in its arguments, with added Gaussian noise :

$$z_t = C_t * x_t + \delta_t \tag{5}$$

Here, C_t is a matrix of size $k \times n$, where k is the dimension of the measurement vector z_t. The vector δ_t describes the measurement noise. The distribution of δ_t is a multivariate Gaussian with zero mean and covariance Q_t.

Given Problem

Here, we have been given a simple target tracking problem in one-dimensional space. The state contains three components : position (one-dimensional), velocity and acceleration and it can be expressed as below.

$$x = \begin{pmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \end{pmatrix} \tag{6}$$

where, x_k is the position, $\dot{x}_k$ is the velocity and $\ddot{x}_k$ is the acceleration at time 'k'. T is the size of sample time step.

The process equation for target motion is given by the following kinematic equation:

$$\begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \end{bmatrix} = \begin{bmatrix} 1 & T & \frac{1}{2}T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x_{k-1} \\ \dot{x}k-1 \\ \ddot{x}k-1 \end{bmatrix} + \upsilon_{k-1} \tag{7}$$

and

$$\upsilon_{k-1} \sim N(0, Q) \tag{8}$$

where, $Q = \sigma^2 * \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} & \frac{T^2}{2} \\ \frac{T^3}{2} & 2T^3 & T^2 \\ \frac{T^2}{2} & T^2 & T^2 \end{bmatrix}$, σ represents intensity of Gaussian noise.

Targets are usually tracked with the help of sensors such as radars or lidars which provide the position of the target. Hence, the measurement equation can be written as:

$$y_k = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \end{bmatrix} + \omega \tag{9}$$

where, y_k is the measurement and $\omega \sim N(0, R)$ is the measurement noise.

For the given problem, obtain the estimate of its state over a period of $20\ sec$. Assume time step $T = 0.1\ sec$.