Survey on the Kalman Filter and Related Algorithms

Ms. Fatema Ahmed[1], Mr. Rakesh Mandal[2]

[1]*M. Tech Scholar, Electronics & Telecommunication (Communication Eng.), SSTC, Bhilai, Chhattisgarh, India*
[2]*Asst. Professor, Electronics & Telecommunication, SSTC, Bhilai, Chhattisgarh, India*

Abstract- Bayesian state estimation is the process of recursively estimating the state of a system. In this paper we will summarize three highly influential algorithms that have been implemented in fields as diverse as signal analysis, space flight control, and robotics: the Kalman Filter, the Extended Kalman Filter, and the Particle Filter.

1. Introduction

The ultimate goal of algorithms research is to find an optimal solution for a given problem. However, there are few problems in computer science that can be considered completely solved. That is, it is rare that one can show an algorithm is completely optimal for its problem domain. One of the few areas where a provably optimal algorithm can be found, however, is in Bayesian state estimation. Here, the Kalman Filter, an algorithm that propagates a system's varying quantities over time, can be shown to be the best algorithm possible for its domain. This paper is an introduction to the Kalman Filter and several related Bayesian state estimators. In the rest of this introduction we will introduce Bayesian Filters. In the next section we will describe the Kalman Filter in detail. Then, we will detail the Extended Kalman Filter, and finally the Particle Filter. For each filter, we will provide a sketch of the algorithm, analyze its computational time complexity, and finally sketch a proof of its correctness, or analyze how well it approximates the optimal solution.

1.1. General Bayesian State Estimation

The focus of this paper is the Kalman Filter and its related algorithms. These are examples of Bayesian Filters, named after their application of Bayes' law, expressed in Equation 1.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad \text{Equation 1}$$

Stated simply, Bayes' law says the probability of estimating A given B has occurred is equal to the normalized probability of B given A has occurred, multiplied by the probability of A occurring. Bayes' law is useful when we cannot measure P(A|B) explicitly, but we can measure P(B|A) and P(A). A typical example used to illustrate Bayes' law is estimating the probability of having a disease given a positive test result. In this scenario, the probability of a false-positive and falsenegative are the observable quantities used to calculate this.

Bayesian Filters use Bayes' law to estimate an unobservable state of a given system using observable data. They do this by propagating the posterior probability density function of the state using a transition model. For example, in robotics, the system's state is typically the pose of a robot in its environment, or the configuration of joints in an actuator. This is usually denoted s_t. Also, in robotics the observable data is typically a control that tells the robot how to move, and an observation about the environment it makes with its sensors. These are usually denoted u_t and z_t respectively. The control can be velocity command give to motors, or perhaps desired angles given to joints. The observation can be a range and bearing to an obstacle. Finally, using Bayes' law and the Markov assumption (that the current sate depends only on the previous state and not any state before it) we can state the Bayesian Filter below, which is derived from Equation 1 [4].

$$P(s_t|z_t, u_t) = \eta P(z_t|s_{t-1})P(s_{t-1}|u_t) \qquad \text{Equation 2}$$

Here, η is a normalization factor, z_t is an observation made by the system, u_t is a transition that modifies the state, and s_t is the current state vector. Equation 2 is the basis of all Bayesian state estimators. The difference between them is how the probabilities are estimated, and the form of the input probability distributions. In this paper we will look at three different Bayesian filters, the Kalman Filter, the Extended Kalman Filter, and the Particle Filter. The Kalman Filter is a linear Gaussian estimator, and is optimal. The Extended Kalman Filter is a nonlinear version of the Kalman Filter that can handle more problem instances, but is not optimal. The Particle Filter is a non-parametric estimator that is more flexible than the two but is more computationally expensive.

2. The Kalman Filter

The Kalman Filter is a very rare algorithm, in that it is one of the few that are provably optimal. It was first published by Rudolf E. Kalman in his seminal 1960 paper titled *A New Approach to Linear Filtering and Prediction Problems* [1]. It is used in areas as diverse as aeronautics, signal processing, and futures trading. At its core, it propagates a state characterized by a Gaussian distribution using linear transition functions in an optimal way. Since it is optimal, it has remained relatively unchanged since it was first introduced, but has received many extensions to apply it to more than just linear Gaussian systems. In this section we sketch the Kalman Filter algorithm, and analyze its computational complexity in the time domain. Finally, we show it is an optimal algorithm, and that no algorithm can do better. This section is largely based off Kalman's original paper.

2.1. Outline of the Kalman Filter

Since the Kalman Filter is a Bayesian filter, our goal is to solve Equation 2. However, we first must note the Kalman Filter comes with several assumptions:

1. The state transition is linear in the form

$$s_{t+1} = As_t + Bu_t + w_k \qquad \text{Equation 3}$$

where s_t is the state, $u_t$is the control, and w_k is added Gaussian noise.

2. The measurement is linear in the form

$$z_t = Hs_t + v_k \text{Equation 4}$$

where z_t is the observation, and v_k is added Gaussian noise.

3. The system is continuous.

While these assumptions restrict the applicability of the Kalman Filter, we will show later they also ensure its optimality.

The algorithm is structured in a predictor-corrector format. The general idea is to project the state forward, using a state transition function. Then this state is corrected by incorporating a measurement of the system's observable quantities. The algorithm can be divided into two distinct phases: a time update phase and a measurement update phase.

2.1.1. Time Update

In the time update phase, the state is projected forward using Equation 3. However, we also must propagate the uncertainty in the state forward. Since the state is a Gaussian distribution, and is fully parameterized by a mean s_t and covariance P_t, we can update the covariance as in Equation 5.

$$P_{t+1} = AP_tA^T + Q \qquad \text{Equation 5}$$

Here, A is the same matrix used to propagate the state mean, and Q is random Gaussian noise. Equations 3 and 5 exploit a general property of Gaussians: adding two Gaussians results in a Gaussian, and applying a linear transformation to a Gaussian yields a Gaussian. There properties of Gaussian are crucial to the optimality of the filter. After the time update phase, the original Gaussian characterized by s_t and P_t is a new Gaussian, now characterized by s_{t+1} and P_{t+1}. This concludes the time update phase, and represents the prediction step of the algorithm.

2.1.2. Measurement Update

The measurement update phase is the correction step of the Kalman Filter, wherein a measurement of an observable variable is made and fused with the prior distribution to estimate the posterior. First, we make a measurement of the system using our linear measurement model in Equation 4. After the measurement is made, we form what is known as the Kalman Gain, depicted in Equation 6. This is the key step of the Kalman Filter.

$$K = PH^T(HPH^T + Q)^{-1} \qquad \text{Equation 6}$$

In section 2.3 we will derive this gain, and show it is this that makes the Kalman Filter optimal.

Next, we calculate what is known as the innovation (Equation 7). This is the difference between the expected observation, and the actual observation.

$$\hat{z} = (z_t - H_t s_t) \qquad \text{Equation 7}$$

Now we are ready to calculate the posterior distribution, by combining Equations 6 and 7. Equation 8 corrects the mean while equation 9 corrects the covariance.

$$s_t^- = s_t + K\hat{z} \qquad \text{Equation 8}$$

$$P_t^- = (I - K_tH_t)P_t \qquad \text{Equation 9}$$

Here, $s_t^-$and P_t^-fully parameterize the posterior distribution. The preceding steps represent a single iteration of the Kalman filter. This output is then used as input to a subsequent observation, along with a new control and observation.

3. Extended Kalman Filter

One major limitation of the Kalman Filter is the strict set of problems to which it applies: problems with linear state transition and linear measurements with added Gaussian noise. While a great many problems can be modeled this way, it would be nice to apply the Kalman Filter to other problems, due to its optimality.

The Extended Kalman Filter was invented just for this purpose. It works through a process of linearization, where the nonlinear transition and observation functions are approximated by a Taylor Series expansion. This section derives from a seminal paper on the Unscented Kalman Filter, which is similar to the Extended Kalman Filter [2].

3.1. Outline of the Extended Kalman Filter

With the Extended Kalman Filter, we can no longer assume a linear process update or observation model. We express this condition in Equations 14 and 15

$$s_t = g(s_{t-1}, u_t) \qquad \text{Equation 14}$$

$\hat{z}_t = h(s_{t-1})$ Equation 15

Here, the process update and observation models are characterized by two potentially nonlinear functions $g(s_{t-1}, u_t)$ and $h(s_{t-1})$. Since the Kalman Filter works on only linear inputs, we must linearize these functions to propagate the state covariance matrix forward. We do this by approximating the function as a line tangent to the actual function at the mean value. This line is found by expanding the nonlinear functions in a Taylor Series around the mean, and taking the first order approximation. This expansion is expressed as $y = f(x+\varepsilon) \approx f(x) + J\varepsilon$, where J is the Jacobian of f(x). Equation 16 demonstrates how we use the Jacobian to propagate our covariance matrices.

$C_y = E[(y-\bar{y})(y-\bar{y})^T] \approx E[J\varepsilon\varepsilon^T J^T] = JC_xJ^T$ Equation 16

Here, C_x is our current covariance; J is the Jacobian of our nonlinear state transition function; ε is zero mean Gaussian noise; and C_y is our transformed covariance matrix. However, this result is only approximate due to our use of the Jacobian. Thus, we need to linearize both equations 14 and 15 by taking the gradient of each with respect to the state s_t, as in Equations 17 and 18.

$G_t = \nabla_s g(s_{t-1}, u_t)$ Equation 17

$H_t = \nabla_s h(s_{t-1})$ Equation 18

Thus we now follow the same predictor-corrector form of the original Kalman Filter in two distinct phases.

3.1.1. Time Update

First, we propagate the mean forward to find the mean of our prior distribution. This is simply applying Equation 14. Next, we have to propagate the covariance forward, which was shown in Equation 14 to just be the covariance convolved with the appropriate Jacobian. In this case we use G_t.

$P_t = G_t P_{t-1} G_t^T + Q$ Equation 19

Here, Q is again zero mean Gaussian noise of the process model. These two equations provide us with a fully parameterized Gaussian prior. We now move to the measurement update phase.

3.1.2. Measurement Update

The measurement update phase is largely the same as in the Kalman filter, two important exceptions. First, the Kalman Gain is calculated using a Jacobian. This has the unfortunate consequence that Kalman Gain can no longer be considered optimal, and thus the Extended Kalman Filter cannot be the best possible algorithm for filtering nonlinear systems. We calculate the Kalman Gain as in Equation 6, but this time H_t is found using Equation 18.

The second difference is in calculating the innovation. In the Kalman Filter, the observation model had to be linear. This requirement was only to ensure the calculation for the Kalman Gain worked correctly. Here, our observation model can be nonlinear, but the way we calculate our innovation is straight forward.

$\hat{z} = (z_t - h(s_t))$ Equation 20

With this last equation, we can estimate the posterior in the standard Kalman Filter way.

4. Particle Filter

Until now, we have focused exclusively on parametric Bayesian Filters that manipulate Gaussian distributions, propagating with a transition model. We now turn to a nonparametric filter, with a unique approach to calculating the posterior distribution: the Particle Filter. As its name suggests, it uses a set of hypothesis called particles as guesses for the true configuration of the state. In the following section we present the basic Particle Filter algorithm. This section derives from this seminal 1993 paper by N. Gordon, D. Salmond, and A. Smith titled *Novel approach to nonlinear/non-Gaussian Bayesian state estimation* [3], where they called the algorithm the "Bootstrap Filter".

4.1. Outline of the Particle Filter

The particle filter is different from previous filters in that it is not limited by linear models or Gaussian noise. This flexibility is due to how it represents the probability density function as a set of samples known as particles. Each sample is taken from a proposal distribution, and weighted according to how well it matches a target distribution. After weighting, the particle distribution does not match the posterior, so we carry out the key step in the Particle Filter algorithm: importance sampling. Here, we pick particles from the proposal particle set and add them to the posterior particle set with a frequency proportional to their weight. 4.1.1. Sample

The first step in the particle filter is to go through every particle and sample from a proposal distribution.

$sample\ s_t^{[m]} \sim p(s_t | s_{t-1}^{[m]}, u_t)$ Equation 21

Just as the Bayesian Filters we looked at in previous sections, the Particle Filter is a recursive algorithm, so we therefore sample the current state using the previous state. Here, the superscript [m] on both state variables implies that the state s_t is derived from the same particle in the previous particle set. The same control input u_t as the previous methods is used to propagate the state forward. Since the transition function is noisy, each particle goes through a different transition, which adds variety to the particle set.

Note that we do not explicitly state how to obtain this sample. This is dependent on the exact system, and is one of the requirements of the Particle Filter, that a proposal distribution must be available to sample from. In many applications, this is readily available. If not the case, it sometimes suffices to sample from a Gaussian distribution with appropriate mean and covariance.

4.1.2. Weight

Next each particle is weighted based on how well it matches the posterior distribution. This is expressed in Equation 22.

$$w_t^{[m]} = p(z_t|x_t^{[m]})$$ Equation 22

This is equivalent to the measurement update phase in the Kalman Filter, where the observation is incorporated into the belief. There are many different methods to weight particles, but most involve estimating the following quotient

$$w^{[m]} = \frac{f(s_t^{[m]})}{\quad}$$ Equation 23 t $g(s_t^{[m]})$

Where $g(s_t^{[m]})$ is our proposal distribution, and $f(s_t^{[m]})$ is our target distribution. When we have completed these two steps, we are finally free to add the new weighted particle to a temporary particle set.

4.1.3. Resample

Also known as importance sampling, the resample step uses the newly generated temporary particle set to generate the final posterior distribution. Just as with weighting the particles, there are many ways to accomplish importance sampling, but doing this incorrectly can lead to the wrong posterior distribution. Essentially, the resampling step reduces variance in the particle set, which decreases the accuracy of the posterior approximation. In general, we sample with particles with replacement with a frequency proportional to their weight. However there are myriad variations on this general method.

5. Discussion and Conclusions

So far, we have presented three different Bayesian Filters: The Kalman Filter, the Extended Kalman Filter, and the Particle Filter. In this final section, we will compare the different filters and discuss their applicability in the context of robotics, but with implications for other fields.

The following table summarizes the major aspects of each algorithm.

	Kalman Filter	Extended Kalman Filter	Particle Filter
Input	$s_{t-1}, P_{t-1}, u_t, z_t$	$s_{t-1}, P_{t-1}, u_t, z_t$	Y_{t-1}, u_t, z_t
Output	s_t^-, P_t^-	s_t^-, P_t^-	Y_t^-
Assumption	Linear transition Linear observation Gaussian noise	Nonlinear transition Nonlinear observation Gaussian noise	Nonparametric
State representation	s_t, P_t	s_t, P_t	Y_t
Hypothesis	Single	Single	Multiple
Accuracy	Exact	Approximate, fixed by model	Approximate, but accuracy can be increasing by adding particles
Asymptotic Time Complexity	O(n2.376)	max(O($n^{2.376}$, O($g(s_{t-1}, u_t)$), O($h(s_t)$))	O(M^n)

The first algorithm, the Kalman Filter, is one of the few algorithms researchers can call solved; it solves state estimation for linear Gaussian systems in an optimized manner. For this reason, even though the algorithm is over half a century old, it is still used extensively today. The second algorithm was an extension of the Kalman Filter. It attempts to re-lax the requirements for a linear Gaussian system, so it is applicable to more systems. It does this by expanding the transition and observation models in a Taylor series approximation, and using their Jacobians to propagate covariances. The Ex-tended Kalman Filter is no longer optimal, but the robustness of the original Kalman Filter allows for a great deal of latitude in the degree of nonlinearity of the model functions.

The final algorithm was the Particle Filter, a nonparametric Bayesian State Estimator. This algorithm differs from the previous two in that it has no requirements on the transition and observation models. It manages this because samples are taken from a proposal posterior, and then weighted according to observations in order to approximate the true posterior. The beauty in the Particle Filter is found in the resampling stage, where particles are chosen for the posterior according to their respective weights. Several factors affect the performance of the Particle Filter, including variance in the particle set, frequency of resampling, and how well the proposed posterior matches the target posterior. This flexibility comes at the price of high computational complexity in the size of the state space.

6. References

[1] Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. Transaction of the ASME - Journal of Basic Engineering, 35-45.

[2] Juler, S., & Uhlmann, J. (n.d.). A New Extension of the Kalman Filter to Nonlinear Systems. Storage and Retrieval for Image and Video Databases.

[3] Gordon, N., Salmond, D., & Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEEE Proceedings-F, (pp. 107-113).

[4] Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic Robotics. Cambridge, MA: MIT Press.