Kalman Filter

A. Giavaras

Contents

1 Kalman Filter

Fig. 1: Rudolf E. Klmn receiving the National Medal of Science.

In this section, we'll learn about one of the most famous algorithms in all of engineering; namely the Kalman filter. In today's world of advanced machine learning, the Kalman filter remains an important tool to fuse measurements from several sensors to estimate in real-time the state of a robotic system such as a self-driving car. Concretely, in this section, we will introduce its basic linear formulation and also show why the Kalman filter is the best linear unbiased estimator.

2 Introduction

The Kalman filter algorithm was published in 1960 by Rudolf E. Kalman, a Hungarian born professor and engineer who was working at the Research Institute for Advanced Studies in Baltimore Maryland. Years later in 2009, American President Barack Obama awarded Kalman the prestigious National Medal of Science for his work on the Kalman filter and other contributions to the field of control engineering.

After its publication in 1960, the Kalman filter was adopted by NASA for use in the Apollo guidance computer.

It was this ground-breaking innovation that played a pivotal role in successfully getting the Apollo spacecraft to the moon, and to our first steps on another world. The filter helped guide the Apollo spacecraft accurately through its circumlunar orbit.

The engineers at NASA's Ames Research Center, adopted Kalman's linear theory and extended it to nonlinear models. We will discuss this specific extension in a later section. But first, let's talk about the basic linear Kalman filter.

Fig. 2: The Apollo guidance computer used Kalman filtering.

3 Linear Kalman Filter

The Kalman filter is very similar to the linear recursive least squares filter we discussed earlier. While recursive least squares updates the estimate of a static parameter, but Kalman filter is able to update and estimate of an evolving state. Thus, Kalman filtering is an iterative process that uses a set of equations and consecutive data inputs to quickly estimate the true value; e.g. position, velocity etc., of the object being measured when the measured values contain unpredicted or random error, uncertainty or variation.

The goal of the Kalman filter is to take a probabilistic estimate of this state and update it in real time using two steps; prediction and correction.

To make these ideas more concrete, let's consider a problem of estimating the 1D position of the vehicle.

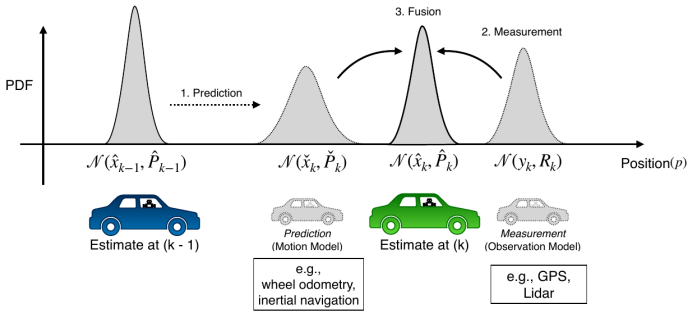

Fig. 3: The Apollo guidance computer used Kalman filtering.

Starting from an initial probabilistic estimate at time $k-1$, our goal is to use a **motion model** which could be derived from wheel odometry or inertial sensor

measurements to predict our new state. Then, we'll use the observation model derived from GPS for example, to correct that prediction of vehicle position at time k.

Each of these components, the initial estimate, the predicted state, and the final corrected state are all random variables that we will specify by their means and covariances. In this way, we can think of the Kalman filter as a technique to fuse information from different sensors to produce a final estimate of some unknown state, taking into account, uncertainty in motion and in our measurements. For the Kalman filter algorithm, we had been able to write the motion model in the following way; the estimate at time step k is a linear combination of the estimate at time step $k-1$, a control input and some zero-mean noise.

The input is an external signal that affects the evolution of our system state. In the context of self-driving vehicles, this may be a wheel torque applied to speed up and change lanes, for example. Next, we will also need to define a linear measurement model. Finally, we'll need a measurement noise as before and a process noise that governs how certain we are that our linear dynamical system is actually correct, or equivalently, how uncertain we are about the effects of our control inputs.

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{1}$$

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{2}$$

where

$$\mathbf{v}_k \sim N(0, \mathbf{R}_k), \quad \mathbf{w}_k \sim N(0, \mathbf{Q}_k) \tag{3}$$

Once we have our system in hand, we can use an approach very similar to that we discussed in the recursive least squares. Except this time, we'll do it in two steps.

- First, we use the process model to predict how our states, remember, that we're now typically talking about evolving states and non-state parameters evolved since the last time step, and will propagate our uncertainty.
- Second, we'll use our measurement to correct that prediction based on our measurement residual or innovation and our optimal gain.

Finally, we'll use the gain to also propagate the state covariance from our prediction to our corrected estimate. This is shown in Figure

In our notation, the hat indicates a corrected prediction at a particular time step. Whereas a check indicates a prediction before the measurement is incorporated. If you've worked with the Kalman filter before, you may also have seen this written with plus and minus signs for the corrected and predicted quantities, respectively.

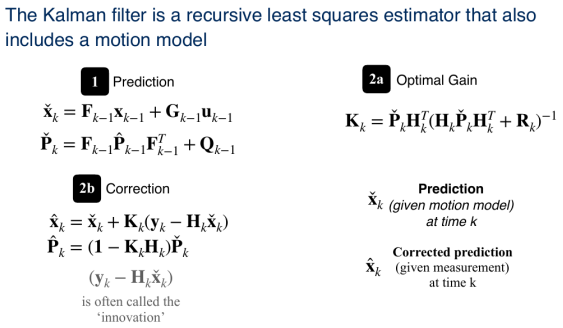

Fig. 4: The linear Kalman filter steps.

Let's recap. We start with a probability density over our states and also maybe parameters at time step $k-1$, which we represent as a multivariate Gaussian. We then predict the states at time step k using our linear prediction model and propagate both the mean and the uncertainty; the covariance, forward in time. Finally, using our probabilistic measurement model, we correct our initial prediction by optimally fusing the information from our measurement together with the prior prediction through our optimal gain matrix $\mathbf{K}$. Our end result is an updated probabilistic estimate for our states at time step k.

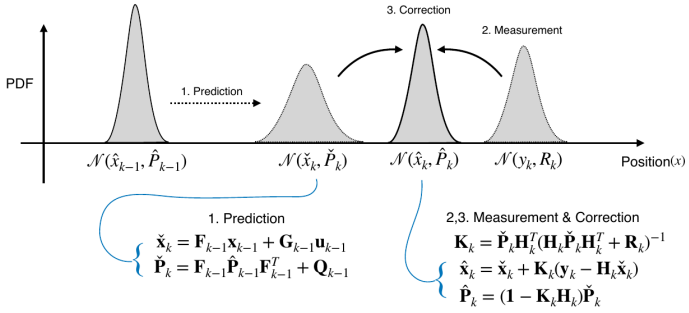

Fig. 5: The linear Kalman filter steps.

3.1 Example: 1D Vehicle Motion

The best way to become comfortable with the Kalman filter is to use it. Let's look at a simple example. Consider again the case of the self-driving vehicle estimating its own position. Our state vector will include the vehicle position and its first derivative velocity. Our input will be a scalar acceleration that could come from a control system that commands our car to accelerate forward

or backwards. For our measurement, we'll assume that we're able to determine the vehicle position directly using something like a GPS receiver. Finally, we'll define our noise variances as follows: Given this initial estimate and our data, what is our corrected position estimate after we perform one prediction step and one correction step using the Kalman filter? Here's, how we can use these definitions to solve for our corrected position and velocity estimates. Pay attention to the fact that our final corrected state covariance is smaller. That is we are more certain about the car's position after we incorporate the position measurement. This uncertainty reduction occurs because our measurement model is fairly accurate. That is, the measurement noise variance is quite small. Try increasing the measurement variance and observe what happens to the final state estimate. To summarize, the Kalman filter is similar to recursively squares, but also adds a motion model that defines how our state evolves over time. The Kalman filter works in two stages: First, predicting the next state using the motion model, and second, correcting this prediction using a measurement. But how can we be sure that the Kalman filter is giving us an accurate state estimate?

4 Kalman Filter and BLUE

We have introduced the Kalman Filter, let's discuss little bit about what makes it such an appealing estimation method.

4.1 Bias

Let's dive in. First, let's discuss bias. Let's consider our Kalman Filter from the previous lesson and use it to estimate the position of our autonomous car. If we have some way of knowing the true position of the vehicle, for example, an oracle tells us, we can then use this to record a position error of our filter at each time step k. Since we're dealing with random noise, doing this once is not enough. We will need to repeat this same process over and over and record our position error at each time step. Once we've collected these errors, if they average to zero at a particular time step k, then we say the Kalman Filter estimate is unbiased at this time step. Graphically, this is what the situation may look like.

Say the particular time step, we know that the true position is the following. We build a histogram of the positions that our filter reports over multiple trials, and then compute the difference between the average of these estimates and the true position. If this difference does not approach zero, then our estimate is biased. However, if this difference does approach zero as we repeat this experiment many more times, and if this happens for all time intervals, then we say that our filter is unbiased. Although we could potentially verify this lack of bias empirically, what we'd really like are some theoretical guarantees.

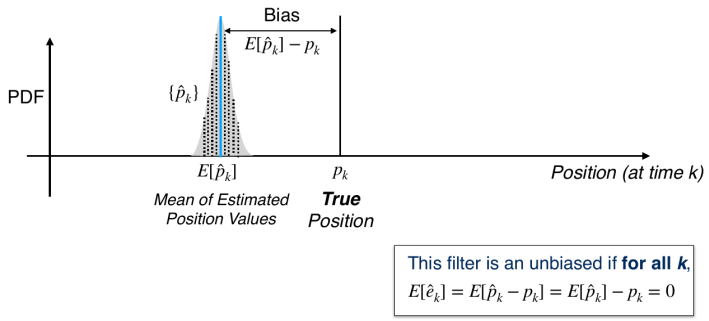

Fig. 6: The linear Kalman filter steps.

Let's consider the error dynamics of our filter. We define our predicted and corrected state errors as follows

$$\text{Predicted state error:} \quad \check{\mathbf{e}}_k = \check{\mathbf{x}}_k - \mathbf{x}_k \tag{4}$$

$$\text{Corrected estimate error:} \quad \hat{\mathbf{e}}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k \tag{5}$$

We can then use the common filter equations to write the following relations.

$$\check{\mathbf{e}}_k = \mathbf{F}_{k-1}\check{\mathbf{e}}_{k-1} - \mathbf{w}_k \tag{6}$$

$$\hat{\mathbf{e}}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\check{\mathbf{e}}_k + \mathbf{K}_k\mathbf{v}_k \tag{7}$$

For the Kalman Filter, we can show the expectation value of these errors is equal to zero exactly. Meaning that we can show

$$E[\check{\mathbf{e}}_k] = \mathbf{0}, \quad E[\hat{\mathbf{e}}_k] = \mathbf{0} \tag{8}$$

For this to be true, we need to ensure that our initial state estimate is unbiased and that our noise is white, uncorrelated, with zero mean. While this is a great result for linear systems, remember that this doesn't guarantee that our estimates will be error free for a given trial, only that the expected value of the error is zero.

4.2 Consistency

Kalman Filters are also what is called consistent. By consistency we mean that for all time steps k, the filter covariants P_k matches the expected value of the square of our error.

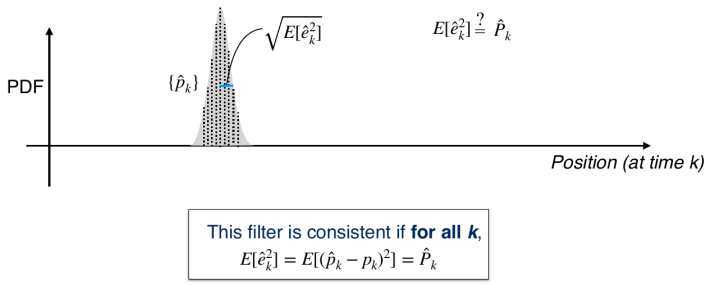

Fig. 7: The linear Kalman filter steps.

For scalar parameters, this means that the empirical variance of our estimate should match the variance reported by the filter. Practically, this means that our filter is neither overconfident, nor underconfident in the estimate it has produced.

4.2.1 Overconfident filter

A filter that is overconfident, and hence inconsistent, will report a covariance that is optimistic. That is, the filter will essentially place too much emphasis on its own estimate and will be less sensitive to future measurement updates, which may provide critical information. It's easy to see how an overconfident filter might have a negative or dangerous effect on the performance of self-driving car.

So long as our initial estimate is consistent, and we have white zero mean noise, then all estimates will be consistent. Putting everything together, we've shown that given white uncorrelated zero mean noise, the Kalman Filter is unbiased and consistent. Because of these two facts, we say that the Kalman Filter is the BLUE, the Best Linear Unbiased Estimator. It produces unbiased estimates with the minimum possible variance.

To summarize, in this section we've defined the terms bias and consistency, and showed that the Kalman Filter is unbiased, consistent, and the Best Linear Unbiased Estimator, or BLUE. Remember that best here refers to the fact that the Kalman Filter minimizes the state variance. Although this is a fantastic result, most real systems are not linear. For self-driving cars, we'll generally need to estimate non-linear quantities like vehicle poses, position, and orientation in 2D and 3D. To do this, we'll need to extend the linear Kalman Filter into the non-linear domain.

4.3 Kalaman Gain

The Kalman gain $K$is used to determine how much of the new measurements to use in order to update the new estimate. In the caluclation of the Kalman gain two quantities participate:

- Error in estimate
- Error in data measurement

Thus K is given by

$$K = \frac{E_{est}}{E_{est} + E_{meas}} \tag{9}$$

From equation 9 it follows that

$$0 \leq K \leq 1 \tag{10}$$

The Kalman gain is the used used to update the current estimate $\hat{x}_t$:

$$\hat{x}_t = \hat{x}_{t-1} + K(z - \hat{x}_{t-1}) \tag{11}$$

When $K \approx 1$ or is equal to one, the measurements we are getting a very accurate however the estimates are unstable. On the other hand when K is small, the measurements we are getting a inaccurate but the estimates are stable since the error is small. The error $E_{\hat{x}}$ in the estimate is given by

$$E_{\hat{x}_t} = (1 - K)E_{\hat{x}_{t-1}} \tag{12}$$

4.4 Calculations of Kalaman Filter

The Kalman filter iterates over the following three calculations:

- Calculate K using equation 9
- Calculate the new estimate using equation 11
- Caluclate the error in the estimate using equation 12

4.4.1 Example: Temperature Estimate

5 Multidimensional Case

Let's now turn attention to the multidimensional case. Let's introduce some notation. Let X_0 and P_0 be the initial state. The matrix X_0 contains the initial state of the system. The matrix P_0 is the initial process covariance matrix. The process covariance matrix represents the error in the estimate.

Remark 5.1. **Covariance Matrix**

A covariance matrix, also known as auto-covariance matrix, dispersion matrix, variance matrix, or variancecovariance matrix, is a matrix whose element in the i, j position is the covariance between the $i - th$ and $j - th$ elements of a random vector. A random vector is a random variable with multiple dimensions. Each element of the vector is a scalar random variable. Each element has either a finite number of observed empirical values or a finite or infinite number of potential values. The potential values are specified by a theoretical joint probability distribution.

Remark 5.2. **Nonlinear Systems**

The definition above holds for nonlinear systems as well, and the results discussed here have extensions to the nonlinear case.

One of the principal uses of observers in practice is to estimate the state of a system in the presence of noisy measurements. We have not yet treated noise in our analysis, and a full treatment of stochastic dynamical systems is beyond the scope of this text. In this section, we present a brief introduction to the use of stochastic systems analysis for constructing observers. We work primarily in discrete time to avoid some of the complications associated with continuous-time random processes and to keep the mathematical prerequisites to a minimum. This section assumes basic knowledge of random variables and stochastic processes; see Kumar and Varaiya [KV86] or strm [st06] for the required material.

Consider again the LTI state-space model

$$\frac{dx}{dt} = Ax + Bu + v \quad y = Cx + Du + w \tag{13}$$

the model is augmented with additional terms representing the error or disturbance. Concretely, v is the process disturbance and w is measurements noise. Both are assumed to be normally distibuted with zero mean;

$$E[v] = 0, \quad E[vv^T] = R_v, \quad E[w] = 0, \quad E[ww^T] = R_w \tag{14}$$

Remark 5.3. **Normally distributed random variable**

A one dimensional random variable X is said to follow the normal distribution

Remark 5.4. **Nonlinear Systems**

The definition above holds for nonlinear systems as well, and the results discussed here have extensions to the nonlinear case.

R_v and R_w are the covariance matrices for the process disturbance v and the measurement noise w respectively. Furthermore, we assume that the variables v, w are not correlated i.e

$$E[vw^T] = 0 \tag{15}$$

Remark 5.5. **Corralated random variables**

Two random variables X and Y are said to be linearly correlated

The initial condition is also modeled as a Gaussian random variable

$$E[x(0)] = x_0, \quad E[x(0)x^T(0)] = P_0 \tag{16}$$

Implementation of the state-space model in a computer requires discretization. Thus the system can be written as discrete-time linear system with dynamics governed by

$$x_{t+1} = Ax_t + Bu_t + Fv_t, \quad y_t = Cx_t + w_t \tag{17}$$

Given the measurements $\{y(\tau), 0 \leq \tau \leq t\}$, we would like to find an estimate $\hat{x}_t$ that minimizes the mean square error:

$$E[(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T] \tag{18}$$

Theorem 5.1. ***Kalman 1961***

Consider the random process x_t with dynamics described by

$$x_{t+1} = Ax_t + Bu_t + Fv_t, \quad y_t = Cx_t + w_t$$

and noise processes and initial conditions described by 14, 15 and 16. The observer gain L that minimizes the mean square error is given by

$$L_t = AP_tC^T(R_w + CP_tC^T)^{-1}$$

where

$$P_{t+1} = (ALC)P_t(ALC)^T + R_vLR_wL^T, \quad P_0 = E[x_0x_0^T\} \tag{19}$$

A proof of this result can be found in [1]. We, note, however the following points:

- the Kalman filter has the form of a recursive filter: given mean square error P_t at time t, we can compute how the estimate and error change. Thus we do not need to keep track of old values of the output.
- Furthermore, the Kalman filter gives the estimate $\hat{x}_t$ and the error covariance P_t, so we can see how reliable the estimate is. It can also be shown that the Kalman filter extracts the maximum possible information about output data. If we form the residual between the measured output and the estimated output,

$$e_t = y_t - C\hat{x}_t \tag{20}$$

we can show that for the Kalman filter the error covariance matrix R_e is

$$R_e(i,j) = E(e_je_k^T) = W_t\delta_{jk} \tag{21}$$

In other words, the error is a white noise process, so there is no remaining dynamic information content in the error.

Theorem 5.2. ***Kalman-Bucy 1961***

The optimal estimator has the form of a linear observer

$$\frac{d\hat{x}}{dt} = A\hat{x} + Bu + L(y - C\hat{x}), \quad \hat{x}(0) = E(x(0))$$

where L is given by

$$L = PC^TR_w^{-1}$$

where P

$$P(t) = E((x(t) - \hat{x}(t))(x(t) - \hat{x}(t))^T)$$

All matrices A, B, C, R_v, R_w, P and L can be time varying. The essential condition is that the Riccati equation (8.30) has a unique positive solution.

6 Kalmans Decomposition of a Linear System

We have already seen that two fundamental properties of a linear input/output system are:

- reachability
- observability

It turns out that these two properties can be used to classify the dynamics of a system. The key result is Kalmans decomposition theorem, which says that a linear system can be divided into four subsystems:

- Σ_{ro} which is reachable and observable
- $\Sigma_{r\bar{o}}$ which is reachable but no observable
- $\Sigma_{\bar{r}o}$ which is not reachable but is observable
- $\Sigma_{\bar{r}\bar{o}}$which is neither reachable nor observable

Thus, from the input/output point of view, it is only the reachable and observable dynamics that matter.

Remark 6.1. **Kalman's decomposition for state-space**

The general case of the Kalman decomposition is more complicated and requires some additional linear algebra; see the original paper by Kalman, Ho, and Narendra [KHN63]. The key result is that the state space can still be decomposed into four parts, but there will be additional coupling so that the equations have the form

7 Extended Kalman Filter

All of our discussion to this point has considered linear filters for linear systems. Unfortunately, linear systems do not exist. All systems are ultimately nonlinear.

However, many systems are close enough to linear that linear estimation approaches give satisfactory results. But close enough can only be carried so far. Eventually, we run across a system that does not behave linearly even over a

small range of operation, and our linear approaches for estimation no longer give good results. In this case, we need to explore nonlinear estimators.

Nonlinear filtering can be a difficult and complex subject. It is certainly not as mature, cohesive, or well understood as linear filtering. There is still a lot of room for advances and improvement in nonlinear estimation techniques. However, some nonlinear estimation methods have become (or are becoming) widespread. These techniques include nonlinear extensions of the Kalman filter, unscented filtering, and particle filtering.

In this section, we will discuss some nonlinear extensions of the Kalman filter. The Kalman filter that we discussed earlier in this book directly applies only to linear systems. However, a nonlinear system can be linearized as discussed in Section 1.3, and then linear estimation techniques (such as the Kalman or H_∞, filter) can be applied. This chapter discusses those types of approaches to nonlinear Kalman filtering.

7.1 The Linearized Kalaman Filter

In this section, we will show how to linearize a nonlinear system, and then use Kalman filtering theory to estimate the deviations of the state from a nominal state value. This will then give us an estimate of the state of the nonlinear system.

We will derive the linearized Kalman filter from the continuowtime viewpoint, but the analogous derivation for discretetime or hybrid systems are straightforward.

7.2 The Extended Kalman Filter

The previous section obtained a linearized Kalman filter for estimating the states of a nonlinear system. The derivation was based on linearizing the nonlinear system around a nominal state trajectory. The question that arises is, How do we know the nominal state trajectory?

In some cases it may not be straightforward to find the nominal trajectory. However, since the Kalman filter estimates the state of the system, we can use the Kalman filter estimate as the nominal state trajectory.

This is sort of a bootstrap method. We linearize the nonlinear system around the Kalman filter estimate, and the Kalman filter estimate is based on the linearized system. This is the idea of the extended Kalman filter (EKF), which was originally proposed by Stanley Schmidt so that the Kalman filter could be applied to nonlinear spacecraft navigation problems.

However, the linear Kalman filter cannot be used directly to estimate states that are nonlinear functions of either the measurements or the control inputs.

For example, the pose of the car includes its orientation which is a nonlinear quantity, orientations in 3D live on a sphere, in fact.

Thus, we need to look for something else. The EKF is designed to work with nonlinear systems and it's often considered one of the workhorses of state estimation because it's used in all sorts of applications including self-driving cars.

The filter works by first predicting the mean and covariance of the updated state estimate at some time step k based on the previous state and any inputs we give to the system, such as the position of the accelerator pedal. The filter then uses a measurement model to predict what measurements should arrive based on the state estimate and compares these predictions with the measurements that actually arrive from our sensors. The Kalman gain K tells us how to weight all of these pieces of information, so that we can optimally combine them into a corrected estimate, that is, a new state and an updated co-variance. This is sometimes called a predictor-corrector architecture. As we already know, the Kalman filter is actually the best of all possible estimators for linear systems. Unfortunately, there is a catch. Linear systems don't exist in reality. Even a very simple system like a resistor with a voltage applied is not truly linear, at least not all the time. For a certain range of voltages, the current is a linear function of the voltage and follows Ohm's Law. But as the voltage gets higher, the resistor heats up which alters the resistance in a nonlinear way. Since the systems that we encounter in practice are nonlinear, this raises an important question. Can we still use the Kalman filter for nonlinear systems? If so, how?

8 Linearization

The key concept in the Extended Kalman Filter is the idea of linearizing a nonlinear system. For this reason, the EKF is sometimes referred to as the Linearized Kalman filter. Linearizing a system just means picking some operating point α and finding a linear approximation to the nonlinear function f in the neighborhood of α. In two dimensions, this means finding the tangent line to the function $f(x)$ when $x = \alpha$. Mathematically, we do this by taking the Taylor series expansion of the function.

Remark 8.1. **Taylor Series Expansion**

The Taylor series expansion is a way of representing a function as an infinite possibly, some whose terms are calculated from the function's derivatives at a single point.

$$f(x) \approx f(\alpha) + \frac{\partial f}{\partial x}|_{x=\alpha}(x-\alpha) + \frac{1}{2!}\frac{\partial^2 f}{\partial x^2}|_{x=\alpha}(x-\alpha)^2 + \dots \tag{22}$$

For linearization, we're only interested in the first order terms of the Taylor

series expansion.

Let's return to our general nonlinear motion and measurement models and try to linearize them. What should we choose as the operating point for our Taylor's expansion? Ideally, we would like to linearize the models about the true value of the state but we can't do that because if we already knew the true value of the state, we wouldn't need to estimate it. Instead, let's pick the next best thing, the most recent estimate of the state. For our emotion model, we'll linearize about the posterior estimate of the previous state and for the measurement model, we'll linearize about our prediction of the current state based on the motion model.

$$\begin{aligned}\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \approx \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) + \\ \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \\ \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{w}_{k-1}}|_{\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}} \mathbf{w}_{k-1}\end{aligned} \tag{23}$$

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k) \approx \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}) + \frac{\partial h_k}{\partial x_k}|_{\check{\mathbf{x}}_k, \mathbf{0}}(\mathbf{x}_k - \check{\mathbf{x}}_k) + \frac{\partial h_k}{\partial \mathbf{v}_k}|_{\check{\mathbf{x}}_k, \mathbf{0}} \mathbf{v}_k \tag{24}$$

So, now we have a linear system in state space and the matrices F, L, H and M are called Jacobian matrices of the system. Computing these matrices correctly is the most important and difficult step in the Extended Kalman filter or EKF algorithm, and it's also the most common place to make mistakes.

Remark 8.2. **Jacobian Matrix**

In vector calculus, a Jacobian or Jacobian matrix is the matrix of all first-order partial derivatives of a vector valued function. Each column of the Jacobian contains the derivatives of the function outputs with respect to a given input. For example, if your function takes a three-dimensional vector and spits out a two-dimensional vector, the Jacobian would be a two by three matrix. Intuitively, the Jacobian matrix tells you how fast each output of your function is changing along each input dimension, just like how the derivative of a scalar function tells you how fast the output is changing as you vary the input. The Jacobian is really just a generalization of the first derivative to multiple dimensions.

The Jacobian matrix captures the first derivatives of each of the two output variables with respect to each of the two input variables. The best way to get comfortable with driving Jacobian is just practice.

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ x_1^2 \end{bmatrix} \tag{25}$$

The Jacobian matrix is then

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{f_1}{\partial x_1} & \frac{f_1}{\partial x_2} \\ \frac{f_2}{\partial x_1} & \frac{f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2x_1 & 0 \end{bmatrix} \tag{26}$$

Now, we know how to compute the Jacobian matrices needed for the EKF, and all that's left is to plug them into our standard Kalman filter equations. There are a couple of differences to notice in the EKF equations compared to the Kalman filter equations.

- First, in the prediction and correction steps, we're still using the nonlinear models to propagate the mean of the state estimate and to compute the measurement residual or innovation. That's because we linearized our motion model about the previous state estimate, and we linearized the measurement model about the predicted state. By definition, the linearized model exactly coincides with the nonlinear model at the operating points.
- The second difference is the appearance of the L and M Jacobians related to the process and measurement noise. In many cases, both of these matrices will be identity since noise is often assumed to be additive but this is not always the case. So far, this has all been very abstract.

Figure 8 summarizes the Extended Kalman filter algorithm

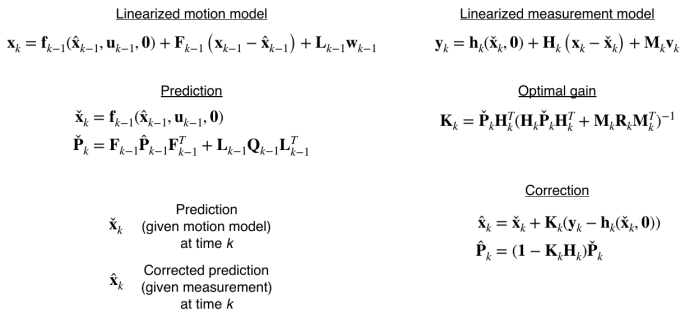

Fig. 8: Summary of EKF.

9 Example: EKF

So, let's walk through a concrete example of actually using the EKF. We will use the same example from module two but with a twist. We're going to track the

position and velocity of a car moving along a rail. But now, instead of receiving periodic GPS measurements that tell us our position, we're going to use an on-board sensor like a camera to measure the altitude of distant landmarks relative to the horizon. We will keep the same linear motion model as in the original example, and assume we know both the height of the landmark and its position in a global reference frame. See for example Figure 9.

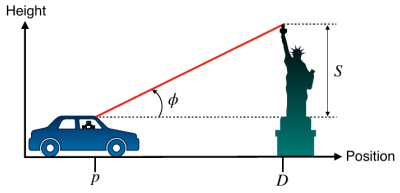

Fig. 9: Summary of EKF.

Thus, the state vector is given by the vector

$$\mathbf{x}_k = \begin{bmatrix} p \\ \dot{p} \end{bmatrix} \tag{27}$$

and it is propagated according to

$$\mathbf{x}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{28}$$

The input signal is given by

$$\mathbf{u} = \ddot{p} \tag{29}$$

The landmark measurement model is

$$y_k = \arctan(\frac{S}{D - p_k}) + v_k \tag{30}$$

Finally, we will use the following noise densities

$$v_k \sim N(0, 0.01), \quad \mathbf{w}_k \sim N(\mathbf{0}, \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}) \tag{31}$$

Because our sensor is measuring an angle, our measurement model has a nonlinear dependence on the position of the car. We are going to need to linearize the

measurement model and use it in our Extended Kalman Filter. The Jacobians for this problem look like this.

$$\mathbf{F}_{k-1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \quad \mathbf{L}_{k-1} = \mathbf{I}_{2\times 2}, \quad \mathbf{H}_k = \begin{bmatrix} \frac{S}{(D-\check{p})^2+S^2} & 0 \end{bmatrix}, \quad M_k = 1 \tag{32}$$

Notice that the $\mathbf{F}$ matrix in this problem is exactly the same as the $\mathbf{F}$ matrix in the original problem. This is because our motion model is already linear in the state. Also notice that the noise Jacobians, $\mathbf{L}$ and $\mathbf{M}$, are both identity since both the emotion and the measurement model have additive noise.

We will be using the following data to estimate the position of the vehicle at time one using the EKF.

$$S = 20m, D = 40m, y_1 = 30deg, u_0 = -2m/s^2, \Delta t = 0.5s \tag{33}$$

$$\hat{\mathbf{x}}_0 \sim N(\begin{bmatrix} 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}) \tag{34}$$

Here is the result of the prediction step for the mean and covariance of the state. Notice that the result is identical to the linear Kalman filter case because the motion model actually is linear. For the correction step, this is what you should get. Keep in mind that you should use the nonlinear measurement model to compute the measurement residual, not the linearized model. Also note that in this case, even though the corrected mean at the state estimate is different from the predicted mean, the corrected co-variance didn't change that much from the predicted co-variance. This is because the Azimuth angle changes slowly at this distance and doesn't provide much information about the vehicle state compared to a GPS measurement.

In summary, the Extended Kalman Filter or EKF uses linearization to adopt the Kalman filter to nonlinear systems. We will encounter several different nonlinear systems to which we can apply the EKF or its cousin, the UKF, in the upcoming course project. Linearization works by computing a local linear approximation to a nonlinear function using a first-order Taylor series expansion about an operating point. This requires several Jacobian matrices which contain the set of first-order partial derivatives. In the next section, we will discuss an alternative formulation of the EKF called the error state Extended Kalman Filter. This would be a useful tool later in the course when we talk about estimating the vehicle orientation in 3D space.

References

[1] Åström K. J., Murray R. M. *Feedback Systems. An Introduction for Scientists and Engineers*

[2] Philip , Florent Altche1, Brigitte dAndrea-Novel, and Arnaud de La Fortelle *The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles?* HAL Id: hal-01520869, https://hal-polytechnique.archives-ouvertes.fr/hal-01520869

[3] Marcos R. O., A. Maximo *Model Predictive Controller for Trajectory Tracking by Differential Drive Robot with Actuation constraints*