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.

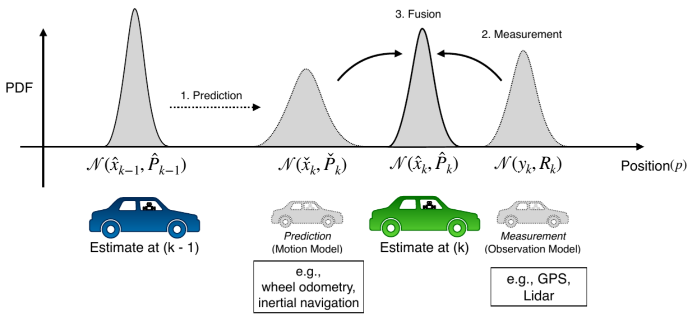

PDF
1. Prediction
3. Fusion
2. Measurement
$\mathcal{N}(\hat{x}_{k-1}, \hat{P}_{k-1})$
$\mathcal{N}(\check{x}_k, \check{P}_k)$
$\mathcal{N}(\hat{x}_k, \hat{P}_k)$
$\mathcal{N}(y_k, R_k)$
Position(p)
Estimate at (k - 1)
Prediction (Motion Model)
Estimate at (k)
Measurement (Observation Model)
e.g., wheel odometry, inertial navigation
e.g., GPS, Lidar

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.

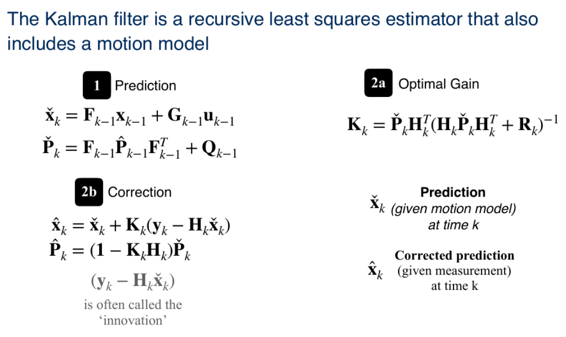

The Kalman filter is a recursive least squares estimator that also includes a motion model
1 Prediction
$\check{\mathbf{x}}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1}$
$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$
2a Optimal Gain
$\mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{H}_k^T(\mathbf{H}_k\check{\mathbf{P}}_k\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$
2b Correction
$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\check{\mathbf{x}}_k)$
$\hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)\check{\mathbf{P}}_k$
$(\mathbf{y}_k - \mathbf{H}_k\check{\mathbf{x}}_k)$
is often called the 'innovation'
$\check{\mathbf{x}}_k$ Prediction (given motion model) at time k
$\hat{\mathbf{x}}_k$ Corrected prediction (given measurement) at time k

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.

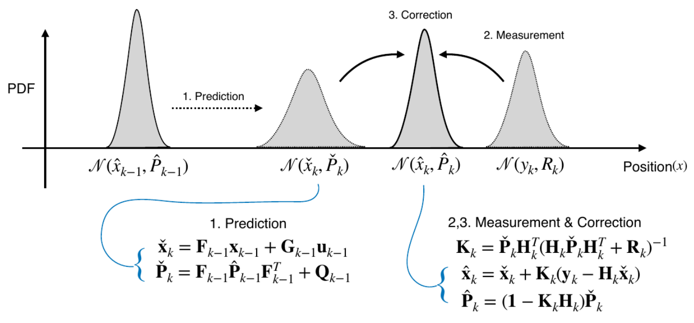

PDF
1. Prediction
3. Correction
2. Measurement
$\mathcal{N}(\hat{x}_{k-1}, \hat{P}_{k-1})$
$\mathcal{N}(\check{x}_k, \check{P}_k)$
$\mathcal{N}(\hat{x}_k, \hat{P}_k)$
$\mathcal{N}(y_k, R_k)$
Position(x)
1. Prediction
$\check{\mathbf{x}}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1}$
$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$
2,3. Measurement & Correction
$\mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{H}_k^T(\mathbf{H}_k\check{\mathbf{P}}_k\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$
$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\check{\mathbf{x}}_k)$
$\hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)\check{\mathbf{P}}_k$

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.

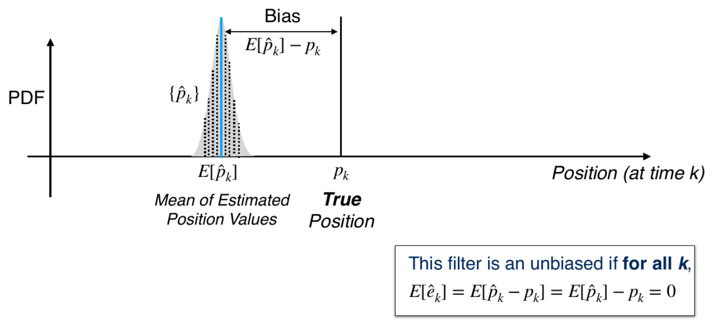

Bias
$E[\hat{p}_k] - p_k$
PDF
$\{\hat{p}_k\}$
$E[\hat{p}_k]$
Mean of Estimated Position Values
p_k
True Position
Position (at time k)
This filter is an unbiased if for all k,
$E[\hat{e}_k] = E[\hat{p}_k - p_k] = E[\hat{p}_k] - p_k = 0$

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.

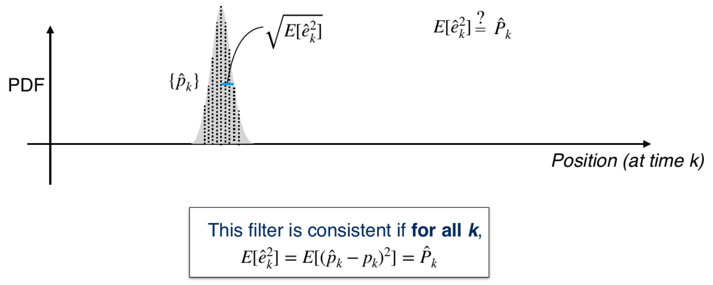

$\sqrt{E[\hat{e}_k^2]}$
$E[\hat{e}_k^2] \overset{?}{=} \hat{P}_k$
PDF
$\{\hat{p}_k\}$
Position (at time k)
This filter is consistent if **for all *k***,
$E[\hat{e}_k^2] = E[(\hat{p}_k - p_k)^2] = \hat{P}_k$

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 Bayesian derivation of the Kalman filter

So in the previous lecture, we presented a solution to the filtering equations for linear and Gaussian models. For these models, both the predicted and the posterior density is Gaussian. And the mean and covariance of these are given by the Kalman filter equations. Although we gave some hints in the previous lecture for why the Kalman filter equations actually make sense, in this lecture, we will more formally derive them, and we will do so from a Bayesian perspective. Before we start, However, we need to define some prerequisites and some notation. So foremost, we assume that we have a linear and Gaussian model. So the state at the current type is equal to the state at the previous time instance, times this transition matrix, A k minus 1. To account for added uncertainty in the predicted state, we also have this additive noise process, q, which is soon to be Gaussian with zero mean and covariance q k minus 1. The observation yk is similarly describe as a linear function of the state, xk, multiplied by this measurement model matrix, Hk. And again, we have this additive noise process, r, which we can assume to be Gaussian with zero mean and covariance Rk In this case. Additionally, we also need to assume that the prior state x naught is also Gaussian, with some mean and some covariance. We should also note that an equivalent way of expressing these models here is to express them on their density form like this, where the process or motion model is defined as a density over xk, where we know the state at the previous time instance, xk minus 1. And the measurement model or likelihood is defined as a density over the observation yk if we condition on the current state xk. I would encourage you to make sure that you understand the that these two express exactly the same thing, just in two different ways. So the objective of this lecture is to use these models and the assumptions that we presented here to derive analytical expressions for the predicted density. That is the density over xk given measurements up to time k minus 1. And for the posterior density, which is then the density over xk, but now we also condition on the measurement at the current time, k. So there are many ways to derive the Kalman filter expressions. One possible way is to derive the Kalman filter equations from the filtering equations. To do this, we simply plug in the Gaussian densities that we presented in the previous slide into these equations, and then try to solve them. So for example, we need to calculate this integral here and we need to solve this product here. Although it's completely possible to do so, unfortunately this involves various matrix manipulations, and it's rather tedious. So what we're going to present here is a more standard derivation, where we instead make use of well-known, or at least well-known for statisticians, results regarding Gaussian distributions. Additionally, we hope that this will give you some better understanding for what's happening in the Kalman filter and help you understand a bit better the non-linear filters that are based on the Kalman filter that we will learn later on in this course. Let's start with the prediction step. So the objective here is to compute the predicted density. And to do so, we use two assumptions. First, we assume that the posterior density from the previous time instance– so the density over xk minus 1, given measurements up to k minus 1, is a Gaussian density with this mean and this covariance. So remember that for our linear Gaussian models all our filter densities are Gaussian. And as a Kalman filter is a recursive filter, this is a relevant assumption to make. Secondly, we assume that we have a linear and Gaussian process model like this that we presented in the previous slide. With this in mind, what we want to do is use these assumptions to calculate the mean and covariance of this density. We will base our derivation on the well-known result that if we have two independent Gaussian random variables– let's call them z 1 and z 2 in this case. A linear combination of these two variables is then also Gaussian random variable, where the mean is just a linear combination of the mean of the these variables and the covariance can be calculated like this, from the covariance of the individual Gaussian random variables. We can easily derive these expressions by using fundamental rules for the expectation and covariance operator that we have learned in the beginning of the course. So as xk minus 1 and qk minus 1 are independent Gaussian random variables, we can use this result directly on our motion model to get the predicted density. So we want to calculate our predicted density, P of xk, given measurements up to k minus 1, which equals a Gaussian density of xk where the mean is given by the expected value of xk, as described by this process model, where we condition on observations up to time k minus 1. So we have ak minus 1 times the expected value of xk minus 1, given measurements up to k minus 1, which is exactly this mean here.

And then the expected value of qk minus 1 is just zero. So it's zero mean. So we get nothing else. For the covariance, we again use this expression here. So the covariance of xk minus 1, which is this covariance here, is then multiplied by the transition matrix on both sides. We have Ak minus 1 times Pk minus 1, given k minus 1 Ak minus 1 transpose, according to this formula here. And then we need to add the covariance for our process noise, qk minus 1. So that's qk minus 1.

To map this to the notation that we have, we call this mean, x hat k given k minus 1, and we interpret this as the estimate of x at time k, given observations up to k minus 1. And similarly, for the covariance we call this Pk given k minus 1, which is the covariance of xk conditioned on measurements up to k minus 1. And we see that what we calculate here is exactly what we calculate in the prediction step of the Kalman filter. So we have now derived the prediction step in a Kalman filter. Before we tackle the update step in the Kalman filter, we need to understand an important lemma for conditional distributions of Gaussian random variables. And it goes like this. So if x and y or two Gaussian random variables with a joint probability density function like this– so we concatenate x and y into single vector. And that vector then is Gaussian distributed with mean mu x and mu y, which is simply the expected value of x. And mu y is simply the expected value of y. And that covariance matrix of this joint vector here has this structure, where P of xx is the covariance of x. And Pyy is then the covariance of y. And Pxy is then the cross covariance of x and y. We denote this as the covariance of x comma y. And where Pyx is simply the cross covariance between y and x, which is simply Pxy transpose, as a covariance matrix needs to be symmetric. So these need to be the transpose of each other. And this cross covariance is then related to how x and y are correlated to each other or dependent on each other. One should note that this structure here is general and that we haven't made any other assumptions than that x and y are jointly Gaussian. So given that two variables are jointly Gaussian, we can always structure its mean and covariance in this matter. So based on this, the lemma states that the conditional distribution of x given y, for example, is then also Gaussian, where the mean is given by the mean of x. But we slightly shift it compared to the prior mean of x, by this term here. We further see that this term here depends on the distance between the actual y and what we expect y to be. And that we wait this distance by a ratio that it's dependent on how correlated x and y are, divided by how much uncertainty we have in y. As a sanity check, we can see that if Pxy is 0– so x and y are uncorrelated. We see that this term disappears and the mean is simply the same as before. So if we observed y and they are uncorrelated, it doesn't change the mean of x, which is to be expected, right? For the conditional covariance on the other hand, again, we take the prior covariance of x, Pxx, and then we reduce it by this factor here. And here we can make two observations. First, as with the conditional mean if x and y are uncorrelated, so Pxy is zero. This term again disappears and the covariance of the conditional density is the same as the covariance for x. So observing y, and x and y are uncorrelated, doesn't change the covariance of x. It doesn't give us any information on x. Second, if on the other hand x and y are correlated, the covariance of x, if we conditional y, will always be less than the covariance of x because we reduce it by this factor here, and this factor here is always positive. And this also makes sense, right? If they are correlated, after observing y, we should be less uncertain about x than before we made observation. So in sum, the lemma states that if we have two jointly Gaussian random variables, the conditional density is also Gaussian with the mean and the covariance expressed like this. Perhaps you can already now see some patterns here for how we can use this lemma to prove the update step in the Kalman filter. But let's look at this in some more detail. So when we do the update step in the Kalman filter, we assume that we have already made the prediction step that we showed earlier. That is, we have computed the moments of the predicted density of xk given observations up to k minus 1. And we denote these moments as this. Additionally, we observe a measurement y at time k, which is related to the state at time k through this linear measurement model. So here we have two Gaussian random variables. And in order to see that they are jointly Gaussian, it is convenient that we use the measurement model and rewrite it to express the joint random variable x comma y. So we have this joint variable, xk yk, which we want to express as something times xk plus something times rk. So we want to write it on this form here. And in order for this to hold true, we see that we need to have the identity matrix here and we need to have a 0 here. And this needs to be Hk. And this needs to be the identity matrix. So now we see that this equations holds true. So xk is equal to xk plus 0 times rk. And yk is equal to Hk xk plus rk, which is what we see here. So now that we have written it on this form, it actually follows directly that we can write the joint distribution of xk yk condition on all the measurements up to k minus 1 as this Gaussian density. And it's perhaps useful to map this back to the notation that was used in lemma on the previous slide. So we have the expected value of x here. So this was– we call this mu x. And this is then mu y. This we called Pxx.

And this whole expression here, we call that Pyy. And this was then the cross covariance between x and y. And this here was then the cross covariance between y and x, which is actually equal to the cross covariance between x and y transposed. Now if we use the lemma on conditional Gaussian densities, we can express the posterior density as this Gaussian density where the posterior mean, x hat k given k, is equal to– so if you use the lemma on the previous slide, we see that we can form this as mu x plus Pxy times Pyy inverse times y minus mu y. So simply using the lemma on the previous slide, we see that we can formulate the posterior mean as this expression here. And if we now identify what these are from what we see here, we see that this is equal to x hat k given minus 1 plus Pxy, this expression here, pk given k minus 1 and Hk transpose. And then Pyy is this expression here. Remember from the previous lecture, we actually call this Sk as the innovation covariance. So to simplify things, we will call it Sk in this equation. So we have Sk inverse times y is yk in this case minus the expected value of yk given observations up to k minus 1, which is given here. Hk times x hat k given k minus 1. So if we look at this, we can see that this here– this difference here is what we call the innovation in the previous lecture. And this factor here in front of it is what we call the Kalman gain. So this is actually equal to call Kalman gain times the innovation. And this is exactly how the updated mean is calculated in the Kalman filter. So how about the posterior covariance? So again, if we use the lemma, we see that our posterior covariance can be written as Pxx minus cross-covariance x and y times Pyy inverse times Pyx.

And Pyx, we see that this is just a transpose of Pxy. So we exchange it like this. So if we identify what things are here– so Pxx is just a predicted covariance.

And then, Pxy is this expression here.

And Pyy is just Sk, so Sk inverse. And then Pxy transpose is just this expression here, but transposed.

So now, if we introduce a small little trick saying that Sk times Sk inverse– so this product here is simply the identity matrix. So we can insert this into our equation here without changing anything. So if we rewrite this as–

and then insert this identity matrix here in the middle, which we can take the transpose of, as it's a symmetric matrix.

We get the following expression. And if we look at this, we can see that this factor here is actually the Kalman gain. And this factor here is simply the Kalman gain transposed. So with this, we have that the updated covariance matrix is simply the predicted covariance minus the Kalman gain times the innovation covariance times the Kalman gain transpose, which is also exactly how we presented it in the previous lecture. So now we have also derived the update step of the Kalman filter. We have now derived a Kalman filter using some well-known results regarding Gaussian densities. And you might wonder, why is this important? Well, from my perspective, I think it's important from at least two aspects. First, being able to derive the Kalman filter offers some intuition into what the Kalman filter actually does. And secondly, by being able to derive the Kalman filter, we can figure out how we need to adapt equations if the underlying assumptions changes slightly. This could, for example, be if we no longer have zero mean process and measurement noise, how do we then need to adapt these equations in order to fit this slightly different model? We end this lecture with a self-assessment question where you can try out your intuition regarding correlation and the distribution of random variables.

5 Kalman Filter Tuning

6 Kalman Filter and LMMSE Estimators

6.1 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

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

6.2 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

6.2.1 Example: Temperature Estimate

7 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 7.1. **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 7.2. **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 7.3. **Normally distributed random variable**

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

Remark 7.4. **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 7.5. **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 7.1. ***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 7.2. ***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

$$L = PC^T R_w^{-1}$$

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.

8 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 8.1. **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

8.1 Questions

1. Recall that $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \mathbf{K}_k \mathbf{v}_k$ and that $\mathbf{v}_k = \mathbf{y}_k + \mathbf{H}_k \hat{\mathbf{x}}_{k-1}$. Suppose that $y_k = x_k + r_k$ with $H_k = 1$ and $r_k \sim N(0, R)$. Which of the following statements apply?
   (a) if $R = \infty$ then $K_k \approx 0$
   (b) if $R = 0$ then $K_k = \infty$
   (c) if $R = 1$ then $K_k = 0$
   (d) if $R = 0$ then $K_k = \infty$

9 Answers

1. Recall that $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \mathbf{K}_k \mathbf{v}_k$ and that $\mathbf{v}_k = \mathbf{y}_k + \mathbf{H}_k \hat{\mathbf{x}}_{k-1}$. Suppose that $y_k = x_k + r_k$ with $H_k = 1$ and $r_k \sim N(0, R)$. Which of the following statements apply?
   (a) if $R = \infty$ then $K_k \approx 0$
   (b) if $R = 0$ then $K_k = \infty$
   (c) if $R = 1$ then $K_k = 0$
   (d) if $R = 0$ then $K_k = \infty$

   $R = \infty$ means the measurement is completely uninformative, and we should not trust it at all. Thus, the innovation should not be incorporated in the posterior, and we can note that achieves just that. $R = 1$ means the measurement is completely free from noise, and we can trust it completely. Thus, the innovation should be incorporated to 100If we get a , we have a problem, since that would mean we would take a step in the direction of the measurement that is longer than to the measurement itself. That would not make sense. Hence, the first and the fourth options are the correct ones.

10 Non-linear Filtering

We developed the Kalman filter algorithm using the following linear state space model

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{22}$$
$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{23}$$

Using this model, we then soleved the filtering problem iteratively using a predictor-corrector type of scheme

$$\text{Predict:}\ \ p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})d\mathbf{x}_{k-1} \tag{24}$$
$$\text{Update:}\ \ p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k)|\mathbf{y}_{1:k-1})}{\int p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})d\mathbf{x}_k} \tag{25}$$

For linear Gaussian models the Kalman filter provides and analytical solution.

10.1 Nonlinear models

However, many important motion and measurement models are nonlinear. For example the coordinated turn motion model is given by

$$\begin{bmatrix} x_k \\ y_k \\ v_k \\ \phi_k \\ \omega_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + \frac{2v_{k-1}}{\omega_{k-1}}\sin(\frac{\omega_{k-1}T)}{2}\cos(\phi_{k-1} + \frac{\omega_{k-1}T}{2}) \\ y_{k-1} + \frac{2v_{k-1}}{\omega_{k-1}}\sin(\frac{\omega_{k-1}T)}{2}\sin(\phi_{k-1} + \frac{\omega_{k-1}T}{2}) \\ v_{k-1}\phi_{k-1} + T\omega_{k-1} \\ \omega_{k-1} \end{bmatrix} + \mathbf{q}_{k-1} \tag{26}$$
$$y_k = \sqrt{x_k^2 + y_k^2} + r_k \tag{27}$$

for such models we, generally, cannot obtain an anlytical solution to the filtering equations. A common approach to solve this, is to find tractable approximate solutions.

Remark 10.1. **Why is nonlinear filtering hard?**

This is because of the following two reasons

- We cannot in general find an analytical solution to our filtering equations.

- For linear systems, it is enough to describe the posterior using the first two moments, the mean and the covariance. For nonlinear systems, we in general need infinitely many moments to describe the true posterior.

The optimal filtering equations hold also in the non-linear setting. It is just harder to solve the equations, because we cannot find analytical solutions to them (in general). When the models are linear and Gaussian, the posterior density is also Gaussian, which means it could be described completely by the first two moments only. In the non-linear setting, we need infinitely many moments to fully describe the posterior, so approximations are necessary.

Weather the Markov property holds or not decides if you can at all use recursive filters to solve your estimation problem. In this chapter we still assume that the Markov property holds.

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.

11 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.

In this section, we will discuss some nonlinear extensions of the Kalman filter. The Kalman filter that we discussed earlier, applies only to linear systems. However, a nonlinear system can be linearized, and then linear estimation techniques (such as the Kalman or H_∞, filter) can be applied.

11.1 The Extended Kalman Filter

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.

Remark 11.1. **EKF and IEKF**

Now this filter comes in two versions, you might say, the normal one and then as an iterative solution similar to what we do when we solve an optimization problem using a Gauss Newton search. The basic idea is simple and natural. if we have nonlinear models $\mathbf{f}_{k-1}$ and $\mathbf{h}_k$, why don't we simply linearize these? This will lead us to linear models and we know how to handle these.

So in the prediction step we linearize our nonlinear motion model around our posterior estimate from the previous time instance and use a normal Kalman filter prediction.

Now why do we use the posterior mean from the previous time instance as our linearaziation point? Well, our motion model $\mathbf{f}$ is a function of $\mathbf{x}_{k-1}$. Our best guess of what $\mathbf{x}_{k-1}$ is when we do the prediction is our posterior mean. So naturally, we want our linearization to be as accurate as possible, at or near where we think $\mathbf{x}_{k-1}$ actually is.

Now for the update step, we basically do the same thing. When we linearize $\mathbf{h}_k(\mathbf{x}_k$, but as the measurement model is a function of $\mathbf{x}_k$, we use our predicted estimate, $\hat{\mathbf{x}}_{k|k-1}$ as our linearization point, and then use the Kalman filter update with this linearized model.

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.

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?

12 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 12.1. **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.

$$f(x) \approx f(\alpha) + \frac{\partial f}{\partial x}|_{x=\alpha}(x-\alpha) + \frac{1}{2!}\frac{\partial^2 f}{\partial x^2}|_{x=\alpha}(x-\alpha)^2 + \dots \tag{28}$$

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.

$$\begin{aligned}\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \approx \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) &+ \\ \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_{k-1},\mathbf{0}}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) &+ \\ \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{w}_{k-1}}|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_{k-1},\mathbf{0}}\mathbf{w}_{k-1}& \end{aligned} \tag{29}$$

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k) \approx \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}) + \frac{\partial h_k}{\partial x_k}|_{\check{\mathbf{x}}_k,\mathbf{0}}(\mathbf{x}_k - \check{\mathbf{x}}_k) + \frac{\partial h_k}{\partial \mathbf{v}_k}|_{\check{\mathbf{x}}_k,\mathbf{0}}\mathbf{v}_k \tag{30}$$

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 12.2. **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.

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ x_1^2 \end{bmatrix} \tag{31}$$

The Jacobian matrix is then

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{f_1}{\partial x_1} & \frac{f_1}{\partial x_2} \\ \frac{f_2}{\partial x_1} & \frac{f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2x_1 & 0 \end{bmatrix} \tag{32}$$

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

Linearized motion model

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) + \mathbf{F}_{k-1}\left(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}\right) + \mathbf{L}_{k-1}\mathbf{w}_{k-1}$$

Linearized measurement model

$$\mathbf{y}_k = \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}) + \mathbf{H}_k\left(\mathbf{x}_k - \check{\mathbf{x}}_k\right) + \mathbf{M}_k\mathbf{v}_k$$

Prediction

$$\check{\mathbf{x}}_k = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0})$$
$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^T$$

Optimal gain

$$\mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{H}_k^T(\mathbf{H}_k\check{\mathbf{P}}_k\mathbf{H}_k^T + \mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^T)^{-1}$$

Correction

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}))$$
$$\hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)\check{\mathbf{P}}_k$$

$\check{\mathbf{x}}_k$ Prediction (given motion model) at time k

$\hat{\mathbf{x}}_k$ Corrected prediction (given measurement) at time k

Fig. 8: Summary of EKF.

13 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.

Thus, the state vector is given by the vector

$$\mathbf{x}_k = \begin{bmatrix} p \\ \dot{p} \end{bmatrix} \tag{33}$$

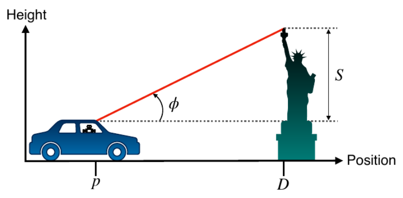


Fig. 9: Summary of EKF.

and it is propagated according to

$$\mathbf{x}_k = \begin{bmatrix}1 & \Delta t\\0 & 1\end{bmatrix}\mathbf{x}_{k-1} + \begin{bmatrix}0\\\Delta t\end{bmatrix}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{34}$$

The input signal is given by

$$\mathbf{u} = \ddot{p} \tag{35}$$

The landmark measurement model is

$$y_k = \arctan(\frac{S}{D - p_k}) + v_k \tag{36}$$

Finally, we will use the following noise densities

$$v_k \sim N(0, 0.01), \quad \mathbf{w}_k \sim N(\mathbf{0}, \begin{bmatrix}0.1 & 0\\0 & 0.1\end{bmatrix}) \tag{37}$$

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.

$$\mathbf{F}_{k-1} = \begin{bmatrix}1 & \Delta t\\0 & 1\end{bmatrix}, \quad \mathbf{L}_{k-1} = \mathbf{I}_{2\times 2}, \quad \mathbf{H}_k = \begin{bmatrix}\frac{S}{(D-\check{p})^2+S^2} & 0\end{bmatrix}, \quad M_k = 1 \tag{38}$$

Notice that the **F** matrix in this problem is exactly the same as the **F** matrix in the original problem. This is because our motion model is already linear in the state. Also notice that the noise Jacobians, **L** and **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.

$$S = 20m, D = 40m, y_1 = 30deg, u_0 = -2m/s^2, \Delta t = 0.5s \tag{39}$$

$$\hat{\mathbf{x}}_0 \sim N(\begin{bmatrix} 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}) \tag{40}$$

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.

14 Questions

1. Consider the following function

$$g(\mathbf{x}) = \sqrt{p_1^2 + p_2^2} \tag{41}$$

and $\mathbf{x} = (p_1, p_2, v_1, v_2)^T$ and $\hat{\mathbf{x}} = (1, 2, 3, 4)^T$. Calculate $g(\hat{\mathbf{x}})$ and $g^{'}(\hat{\mathbf{x}})$

15 Assignements

16 Error State Extended Kalman Filter

The previous section introduced the Extended Kalman Filter, which uses local linearization as a way to allow us to apply the Kalman filter equations to non-linear systems. In this section, we are going to look at a variant of the EKF called the Error-State Extended Kalman Filter, or ES-EKF, which has a couple of nice properties.

By the end of this section, you should be able to describe the error state formulation of the Extended Kalman Filter, and describe the advantages of the error state EKF over the vanilla EKF that you learned about in the previous video.

The idea behind the error state EKF is really very simple. We are going to start thinking about our vehicle state, $\mathbf{x}$, as being composed of two parts;

- A large part called the nominal state, $\hat{\mathbf{x}}$
- A small part called the error state $\delta\mathbf{x}$

We can think of a simple example of tracking the position of a car over time. The green line in Figure 10 shows the true position of the car, which is the quantity we're trying to estimate.

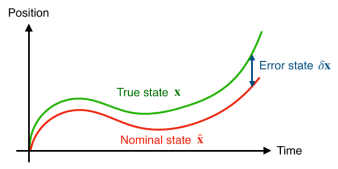

Position
Error state $\delta\mathbf{x}$
True state $\mathbf{x}$
Nominal state $\hat{\mathbf{x}}$
Time

Fig. 10: State error $\delta\mathbf{x}$.

The red line is the nominal state, or our best guess what the true state could be based on what we know about the car's motion model and acceleration and breaking inputs.

Of course, our motion model is never perfect, and there is always some random process noise. These errors build up over time as we integrate the motion model. We can think of the error state as the place where all of these modelling errors and process noise accumulate over time, so that the error state is just the difference between the nominal state and the true state at any given time. If we can figure out what the error state is, we can actually use it as a correction to the nominal state to bring us closer to the true state.

So, in the error state EKF, instead of doing Kalman filtering on the full state which might have lots of complicated non-linear behaviors, we are going to use the EKF to estimate the error state instead, and then use the estimate of the error state as a correction to the nominal state.

What this means mathematically is that we are going to rearrange our linearized motion model so that we now have an equation that can tell us how the difference between the true state at time, k, and our predicted state at time, k, is related to the same difference at time, $k-1$. These differences are exactly the error states we just talked about; $\delta\mathbf{x}_k$ and $\delta\mathbf{x}_{k-1}$,

$$\mathbf{x}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) + \mathbf{F}_{k-1}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \mathbf{L}_{k-1}\mathbf{w}_{k-1} \tag{42}$$

$$\delta\mathbf{x}_{k-1} = \mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1} \tag{43}$$

$$\delta\mathbf{x}_k = \mathbf{x}_k - \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) \tag{44}$$

Remark 16.1. **Error State Kinematics**

The equations relating $\delta\mathbf{x}_k$ and $\delta\mathbf{x}_{k-1}$ are called the error state kinematics.

We can also re-express our linearized measurement model in terms of the error state directly. We can use this error state formulation of the EKF in a very similar way to the vanilla EKF.

We start off by updating the nominal state using the non-linear motion model and our current best estimate of the state. We could do this a bunch of times before ever getting a measurement for the correction step. So, the current best estimate might be $\check{\mathbf{x}}$ or $\hat{\mathbf{x}}$.

$$\check{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) \tag{45}$$

We also need to keep track of the state covariance, which grows as we integrate more and more process noise from the motion model. Note that again, the previous covariance estimate could be $\check{\mathbf{P}}$ or $\hat{\mathbf{P}}$ depending on whether we used a measurement to do a correction step.

$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^T \tag{46}$$

We can repeat the loop updating the nominal state and the error state covariance for as long as we like until we receive the measurement and want to do a correction. When this happens, we can compute the Kalman gain

$$\mathbf{K}_k = \check{\mathbf{P}}_{k-1}\mathbf{H}_k^T(\mathbf{H}_k\check{\mathbf{P}}_k\mathbf{H}_k^T + \mathbf{R})^{-1} \tag{47}$$

Then compute the best estimate of the error state using the Kalman gain, the measurement, and our nonlinear measurement model.

$$\delta\hat{\mathbf{x}}_k = \mathbf{K}_k(\mathbf{y}_k - \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0})) \tag{48}$$

Once we have an estimate for the mean of the error state, we want to use this to update the nominal state and correct the error. We can do that by just adding our estimate of the error state to the nominal state to get the correct state estimate for the full state.

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \delta\hat{\mathbf{x}}_k \tag{49}$$

Finally, we can update the state covariance using the usual equations.

$$\hat{\mathbf{P}}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\check{\mathbf{P}}_k \tag{50}$$

Advantages of ES-EKF

This process goes on forever. So, why would we actually want to use the error state EKF in practice? Well, there are two good reasons to use it.

- One reason is that it can often work better than the vanilla EKF because the small error state is more amenable to linear filtering than the large nominal state, which we can integrate non-linearly.
- The other reason is that the error state formulation makes it much easier to work with constrained quantities like rotations. The reason for this is that we do not necessarily have to use plane vector addition to break down the state. In fact, we could use any generalized composition operation we like as long as it gives us a consistent way of incorporating small perturbations into the nominal state.

In summary, the error state formulation of the EKF separates the vehicle state into a large nominal state and a small error state. The nominal state keeps track of what the motion model predicts the states should be, while the error state captures the modelling errors and process noise that accumulate over time. In the error state EKF, we estimate this small error state and use it as a correction to the nominal state. This is the main difference between the error state EKF and the vanilla EKF, which estimates the full state. Keep in mind that both formulations still rely on local linearization. The error state EKF has a couple of advantages over the vanilla EKF. The first is that it simply performs better because the evolution of the error state tends to be closer to linear. The other is that the error state formulation makes it easier to handle special quantities like 3D rotations.

17 Questions

18 Assignements

19 Unscented Kalman Filter

In the previous section, we saw how linearization errors can cause the EKF to produce state estimates that are very different from the true value of the state and produce covariances that don't accurately capture the uncertainty in the state. This can be a big problem when we're relying on the EKF in safety critical applications like self driving cars. In this section, we will discuss the Unscented Kalman Filter, which is an alternative approach to nonlinear Kalman filtering that relies on something called the unscented transform to pass probability distributions through nonlinear functions. As we will see, the unscented transform gives us much higher accuracy than analytical EKF style linearization, for a similar amount of computation, and without needing to compute any Jacobians.

Hence, by the end of this section, you will be able to use the unscented transform to pass a probability distribution through a nonlinear function. Describe how the Unscented Kalman Filter, or UKF, uses the unscented transform in the predication and correction steps. And explain the advantages of the UKF over the EKF, as well as apply the UKF to a simple nonlinear tracking problem.

20 The unscented transform

The intuition behind the unscented transform is simple. It's typically much easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear functions. Let's see a simple example where a 1D Gaussian distribution like the one on shown on Figure, gets transforms to a nonlinear function into a more complicated a 1D distribution like the one on the right.

We already know the mean and standard deviation of the input Gaussian, and we want to figure out the mean and standard deviation of the output distribution using this information and the nonlinear function. The unscented transform gives us a way to do this. The basic idea in the unscented transform has three steps.

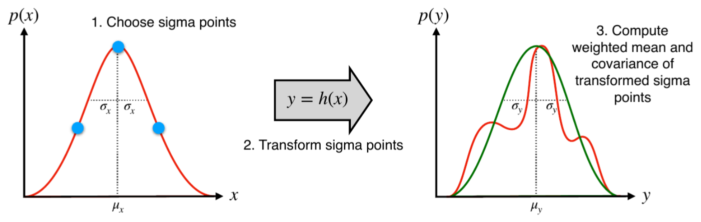


Fig. 11: Summary of EKF.

- We choose a set of sample points from our input distribution. These are not random samples, they are deterministic samples, chosen to be a certain number of standard deviations away from the mean. For this reason, these samples are called sigma points and the unscented transform is sometimes called the sigma point transform.
- Once we have our set of carefully chosen sigma points, the second and easiest step is to pass each sigma point to our nonlinear function producing a new set of sigma points belonging to the output distribution.
- Finally, we can compute the sample mean and covariance of the output sigma points with some carefully chosen weights, and these will give us a good approximation of the mean and covariance of the true output distribution.

Now, that we have seen the basic idea of the unscented transform, let's look at each of these steps in detail.

The first thing you might be wondering is, how many sigma points do we need and which points are in fact sigma points? In general, for an N-dimensional probability distribution $N(\mu, \Sigma)$, we need $2N + 1$ sigma points. One for the mean, and the rest symmetrically distributed about the mean. The diagrams on the left show the sigma points for one dimensional and two dimensional examples.

In 1D, we need two sigma points, and in 2D, we need five. The first step in determining where the sigma point is taking something called the Cholesky decomposition of the covariance matrix associated with the input distribution. The Cholesky decomposition is basically a square root operation that operates on symmetric positive definite matrices, such as covariance matrices.

$$\Sigma = LL^T \tag{51}$$

where L is a lower triangular matrix.

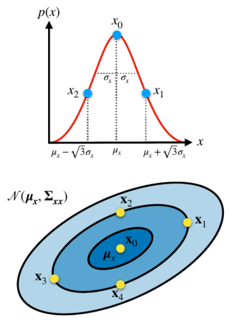


Fig. 12: Summary of EKF.

Remark 20.1. **Cholesky Decomposition**

In fact, if the input PDF is one dimensional, the Cholesky decomposition really is just the square root of the variance which is also known as the standard deviation. We will not go into the details of how to compute a Cholesky decomposition.

Once we have decomposed the covariance matrix, we can choose our first sigma point to be the mean of the distribution and the remainder to be the mean plus or minus some factor times each column of the matrix L that we got from the Cholesky decomposition.

$$\mathbf{x}_0 = \boldsymbol{\mu} \tag{52}$$
$$\mathbf{x}_i = \boldsymbol{\mu} + \sqrt{N+\kappa}\,\mathrm{col}_i L, \quad i = 1, \ldots, L \tag{53}$$

The value of N here again is the number of dimensions of the probability distribution. The parameter κ is a tuning parameter that you're free to set yourself. For Gaussian distributions, which is what we will be working with, setting $N+\kappa$ equal to three is a good choice.

$$\mathbf{x}_{i+N} = \boldsymbol{\mu} - \sqrt{N+\kappa}\,\mathrm{col}_i L, \quad i = 1, \ldots, L, \kappa = 3 - N \tag{54}$$

So now we have our set of sigma points. The next step is straight forward, just pass each of the sigma points through our nonlinear function $\mathbf{h}(\mathbf{x})$ to get a new set of transform sigma points. Now all that is left is to recombine the transform sigma points to find our output mean and output covariance. We do this using the standard formulas for the sample mean and covariance that you would have seen in you introductory statistics courses.

$$\boldsymbol{\mu} = \sum_{i=0}^{2N} \alpha_i \mathbf{y}_i \tag{55}$$

$$\Sigma = \sum_{i=0}^{2N} \alpha_i (\mathbf{y}_i - \boldsymbol{\mu})(\mathbf{y}_i - \boldsymbol{\mu})^T \tag{56}$$

$$\alpha_i = \begin{cases} \frac{\kappa}{N+\kappa}, i = 0 \\ \frac{1}{2}\frac{1}{N+\kappa}, \text{otherwise} \end{cases} \tag{57}$$

The trick is that each of the points gets a specific weight in the mean and covariance calculations, and that weight depends on the parameter κ and the dimension of the input distribution N.

21 Unscented Kalman filter example

To see the unscented transform in action, let get back to our example from the Extended Kalman Filter. Where we nonlinearly transformed a uniform distribution in polar coordinates into Cartesian coordinates. Let us see how the unscented transform compares to the analytical linearization approach.

So here again, we have the true mean and covariance of the two distributions in green. Now let's apply the unscented transform.

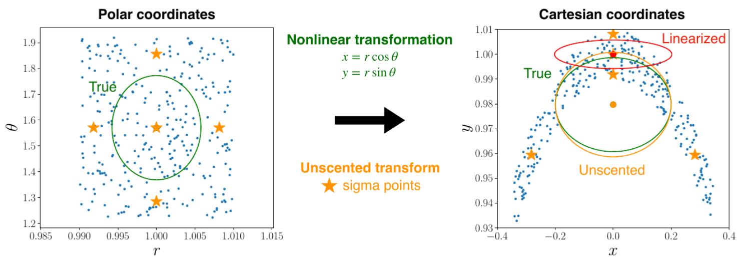


Fig. 13: Summary of EKF.

The dimension of our input distribution is two so we need five sigma points which we have shown as orange stars. Passing these sigma points to our non-linear function puts them here. The mean and covariance we compute from the transformed sigma points look like this shown in orange. Note that our estimate for the mean using the unscented transform is almost exactly the same as the true nonlinear mean. Our estimate for the covariance almost exactly matches the true covariance. Compare that to the analytically linearized transformed mean and covariance in red, which are both very different from the true mean and true covariance. It is easy to see from this example the unscented transform gives us some much better approximation of the output PDF without requiring much more work than the analytical linearization approach.

22 Implementation

Now that we have seen how the unscented transform works, we can easily use it in our Kalman filtering framework to work with nonlinear models.

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}), \mathbf{w}_k \sim N(\mathbf{0}, \mathbf{Q}_k) \tag{58}$$
$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k), \mathbf{v}_k \sim N(\mathbf{0}, \mathbf{R}_k) \tag{59}$$

This variant of the Kalman filter is called the Unscented Kalman Filter or UKF. You may also hear it called the sigma point Kalman filter. The main idea of the UKF is that instead of approximating the system equations by linearizing them, like the EKF does. We use the unscented transform to approximate the PDFs directly.

Let's look at the prediction step of the UKF. In order to propagate the state and covariance through the motion model from time $k-1$ to time k, we apply the unscented transform using the current best guess for the mean and covariance of the state. First, we decompose the estimated state covariance from time $k-1$, then we calculate our sigma points centered around the estimated mean state from time $k-1$. Second, we propagate our sigma points through our nonlinear motion model to get a new set of sigma points for the predicted state at time k. Finally, we calculate the predicted mean and covariance for the state at time k. At this point, it is important to account for the process noise by adding its covariance to the covariance of the transformed sigma points to get the final predicted covariance.

- Compute sigma points

$$\check{\mathbf{L}}_{k-1}\check{\mathbf{L}}_{k-1}^T = \hat{\mathbf{P}}_{k-1} \tag{60}$$
$$\hat{\mathbf{x}}_{k-1}^0 = \hat{\mathbf{x}}_{k-1} \tag{61}$$
$$\hat{\mathbf{x}}_{k-1}^i = \hat{\mathbf{x}}_{k-1} + \sqrt{N+\kappa}\,\mathrm{col}_i\check{\mathbf{L}}_{k-1}, i = 1, \ldots, N \tag{62}$$
$$\hat{\mathbf{x}}_{k-1}^{i+N} = \hat{\mathbf{x}}_{k-1} - \sqrt{N+\kappa}\,\mathrm{col}_i\check{\mathbf{L}}_{k-1}, i = 1, \ldots, N \tag{63}$$

- Propagate sigma points

$$\check{\mathbf{x}}_k^i = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}^i, \mathbf{u}_{k-1}, \mathbf{0}), i = 0, \ldots 2N \tag{64}$$

- Compute predicted mean and covariance

$$\alpha_i = \begin{cases} \frac{\kappa}{N+\kappa}, i = 0 \\ \frac{1}{2}\frac{1}{N+\kappa}, \text{otherwise} \end{cases} \tag{65}$$

$$\check{\mathbf{x}}_k = \sum_{i=0}^{2N} \alpha_i \check{\mathbf{x}}_k^i \tag{66}$$

$$\check{\mathbf{P}}_k = \sum_{i=0}^{2N} \alpha_i (\check{\mathbf{x}}_k^i - \check{\mathbf{x}}_k)(\check{\mathbf{x}}_k^i - \check{\mathbf{x}}_k)^T + \mathbf{Q}_{k-1} \tag{67}$$

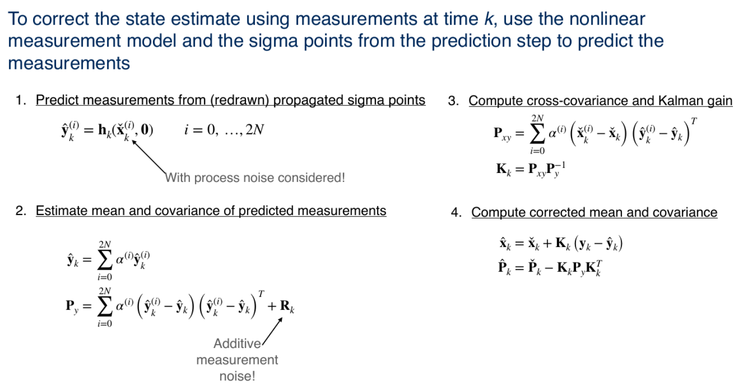
This equation looks a bit different if the process noise is not additive. But most of the time, we will be dealing with additive noise in your models unless there's a good reason not to. For the correction step, we are going to follow a similar procedure, this time with a nonlinear measurement model. First, we will need to redraw our sigma points using the predicted covariance matrix. We need to do this a second time because we added process noise at the end of the last step. This will modify the positions of some of the sigma points. Next, we are going to plug these new sigma points one by one into our nonlinear measurement model to get another set of sigma points for the predicted measurements. Then we can estimate the mean covariance of the predicted measurements using the sample mean and covariance formulas.

Again, take note that we are adding in the measurement noise covariance to get the final covariance of the predicted measurements. Also note that this formula only applies to additive noise. In order to compute the Kalman gain, we are also going to need the cross covariance between the predicted state and the predictive measurement which tells us how the measurements are correlated with the state. You can calculate this using the standard formula for the cross-covariance and using the same weights as before. Then all that is left is to use the Kalman gain to optimally correct the mean and covariance of the predicted states, and that's it. The UKF follows the same prediction correction pattern as the EKF, but we have just replaced the analytical linearization step with the unscented transform. The above is summarized in Figure 14

23 UKF example

Let's try applying the UKF to the same example driving scenario we worked through with the EKF. We are again trying to track the position and velocity of a moving car that we're controlling by pressing on the gas pedal or the brake. The car has a sensor onboard that measures the angle between a distant landmark and the horizon.

The motion model is linear



To correct the state estimate using measurements at time *k*, use the nonlinear measurement model and the sigma points from the prediction step to predict the measurements

1. Predict measurements from (redrawn) propagated sigma points

$$\hat{\mathbf{y}}_k^{(i)} = \mathbf{h}_k(\check{\mathbf{x}}_k^{(i)}, \mathbf{0}) \qquad i = 0, \dots, 2N$$

With process noise considered!

2. Estimate mean and covariance of predicted measurements

$$\hat{\mathbf{y}}_k = \sum_{i=0}^{2N} \alpha^{(i)} \hat{\mathbf{y}}_k^{(i)}$$

$$\mathbf{P}_y = \sum_{i=0}^{2N} \alpha^{(i)} \left(\hat{\mathbf{y}}_k^{(i)} - \hat{\mathbf{y}}_k\right)\left(\hat{\mathbf{y}}_k^{(i)} - \hat{\mathbf{y}}_k\right)^T + \mathbf{R}_k$$

Additive measurement noise!

3. Compute cross-covariance and Kalman gain

$$\mathbf{P}_{xy} = \sum_{i=0}^{2N} \alpha^{(i)} \left(\check{\mathbf{x}}_k^{(i)} - \check{\mathbf{x}}_k\right)\left(\hat{\mathbf{y}}_k^{(i)} - \hat{\mathbf{y}}_k\right)^T$$

$$\mathbf{K}_k = \mathbf{P}_{xy}\mathbf{P}_y^{-1}$$

4. Compute corrected mean and covariance

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right)$$

$$\hat{\mathbf{P}}_k = \check{\mathbf{P}}_k - \mathbf{K}_k\mathbf{P}_y\mathbf{K}_k^T$$

Fig. 14: UKF correction step.

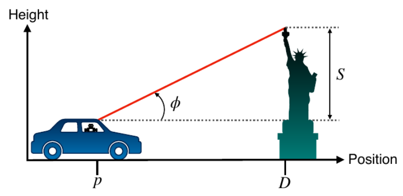


Fig. 15: UKF example setting.

$$\mathbf{x}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{68}$$

but the measurement model is nonlinear.

$$y_k = \arctan(\frac{S}{D - p_k}) + v_k \tag{69}$$

Finally, we will use the following noise densities

$$v_k \sim N(0, 0.01), \quad \mathbf{w}_k \sim N(\mathbf{0}, \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}) \tag{70}$$

Try using the data given in equation 78 to estimate the position of the vehicle at time one using the UKF.

$$S = 20m, D = 40m, y_1 = 30deg, u_0 = -2m/s^2, \Delta t = 0.5s \tag{71}$$

Here is the Cholesky decomposition of the initiate covariance matrix and the five sigma points we'll use to represent the state and its covariance.

$$\hat{\mathbf{L}}_0 \hat{\mathbf{L}}_0^T = \hat{\mathbf{P}}_0 \tag{72}$$

$$\begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix}^T = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix} \tag{73}$$

$$\hat{\mathbf{x}}_0^{(0)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} \tag{74}$$

$$\hat{\mathbf{x}}_0^{(1)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} + \sqrt{3} \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 5 \end{bmatrix} \tag{75}$$

$$\hat{\mathbf{x}}_0^{(2)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} + \sqrt{3} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 6.7 \end{bmatrix} \tag{76}$$

$$\hat{\mathbf{x}}_0^{(3)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} - \sqrt{3} \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.2 \\ 5 \end{bmatrix} \tag{77}$$

$$\hat{\mathbf{x}}_0^{(4)} = \begin{bmatrix} 0 \\ 5 \end{bmatrix} - \sqrt{3} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3.3 \end{bmatrix} \tag{78}$$

The result of the prediction step for the mean of the state.

$$\check{\mathbf{x}}_1^{(0)} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} \tag{79}$$

$$\check{\mathbf{x}}_1^{(1)} = \begin{bmatrix} 2.7 \\ 4 \end{bmatrix} \check{\mathbf{x}}_1^{(2)} = \begin{bmatrix} 3.4 \\ 5.7 \end{bmatrix} \tag{80}$$

$$\check{\mathbf{x}}_1^{(3)} = \begin{bmatrix} 2.3 \\ 4 \end{bmatrix} \tag{81}$$

$$\check{\mathbf{x}}_1^{(4)} = \begin{bmatrix} 1.6 \\ 2.3 \end{bmatrix} \tag{82}$$

$$\check{\mathbf{x}}_k = \sum_{i=0}^{2N} \alpha_i \check{\mathbf{x}}_k^{(i)} = \frac{1}{3} \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} + \ldots + \frac{1}{6} \begin{bmatrix} 1.6 \\ 2.3 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} \tag{83}$$

And finally, the predicted covariance.

$$\check{\mathbf{P}}_k = \sum_{i=0}^{2N} \alpha_i (\check{\mathbf{x}}_k^i - \check{\mathbf{x}}_k)(\check{\mathbf{x}}_k^i - \check{\mathbf{x}}_k)^T + \mathbf{Q}_{k-1} = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \tag{84}$$

Remark 23.1. Notice that the predicted mean and covariance are identical to what we would've found with the linear Kalman filter and the extended Kalman filter. This is because the motion model actually is linear.

For the correction step, these are the sigma points for the predicted measurements and the mean and covariance of that distribution.

And finally, the cross-covariance Kalman gain, and our final answer for the position of the vehicle at time $k = 1$.

$$\hat{y}_1^i = h_1(\check{\mathbf{x}}_1^i, 0), i = 0, \ldots, 2N \tag{85}$$

This gives us

$$\hat{y}_1^{(0)} = \arctan(\frac{20}{40 - 2.5}) = 28.1 \tag{86}$$
$$\hat{y}_1^{(1)} = \arctan(\frac{20}{40 - 2.7}) = 28.7 \tag{87}$$
$$\hat{y}_1^{(2)} = \arctan(\frac{20}{40 - 2.5}) = 28.1 \tag{88}$$
$$\hat{y}_1^{(3)} = \arctan(\frac{20}{40 - 2.3}) = 27.4 \tag{89}$$
$$\hat{y}_1^{(4)} = \arctan(\frac{20}{40 - 2.5}) = 28.1 \tag{90}$$

$$\hat{y}_1 = \sum_{i=0}^{2N} \alpha_i \hat{y}_1^i = 28.1 \tag{91}$$

$$P_y = \sum_{i=0}^{2N} \alpha_i (\hat{y}_1^i - \hat{y}_k)(\hat{y}_1^i - \hat{y}_k)^T + R_k = 0.16 \tag{92}$$

$$\mathbf{P}_{xy} = \sum_{i=0}^{2N} \alpha_i (\check{\mathbf{x}}_k^i - \check{\mathbf{x}}_k)(\hat{y}_1^i - \hat{y}_k)^T + R_k = \begin{bmatrix} 0.23 \\ 0.32 \end{bmatrix} \tag{93}$$

$$\mathbf{K}_1 = \mathbf{P}_{xy} P_y^{-1} = \begin{bmatrix} 1.47 \\ 2.05 \end{bmatrix} \tag{94}$$

$$\hat{\mathbf{x}}_1 = \check{\mathbf{x}}_1 + \mathbf{K}_1(y_1 - \hat{y}_1) = \begin{bmatrix} 5.33 \\ 7.93 \end{bmatrix} \tag{95}$$

To summarize this video, we looked at the Unscented Kalman Filter, or UKF, which uses the unscented transform to adapt the Kalman filter to nonlinear systems. The unscented transform works by passing a small set of carefully chosen samples through a nonlinear system and computing the mean and covariance of the outputs. It often does a much better job of approximating the output distribution than the local, analytical linearization technique used by the EKF for similar computational cost.

Let's recap what we've learned in this module. We started off by discussing the linear Kalman filter which is a form of recursive least-squares estimation that allows us to combine information from a motion model with information from sensor measurements to estimate the vehicle state. The Kalman filter falls a prediction correction architecture. The motion model is used to make predictions of the state and the measurements are used to make corrections to those predictions. We also saw than the Kalman filter is the Best Linear Unbiased Estimator or BLUE. That is the Kalman filter is the best unbiased estimator that uses only a linear combination of measurements.

Of course, linear systems don't really exist in reality. So we needed to develop techniques for handling nonlinear systems. In this module, we looked at three different approaches to nonlinear Kalman filtering. The Extended Kalman Filter or EKF, the error state formulation of the EKF, and the Unscented Kalman Filter or UKF. As we've discussed, the main difference is that the EKF relies on local analytical linearization to propagate PDFs through nonlinear functions. Whether using the full state or the error state formulation. In contrast, the UKF relies on the unscented transform to handle nonlinear functions. For most systems that are only mildly nonlinear, the EKF will give accurate results, but the UKF will be more accurate in cases where a linearization error is problem for the EKF. The error state Kalman filter performs somewhere in between. One of the biggest advantages of the UKF, over any EKF formulation, is that the UKF doesn't require you to compute any derivatives of your nonlinear models, which can be prone to human error or numerical instability. Finally in terms of speed, the EKF wins out by a small margin for typical estimation problems. But in general, the EKF and the UKF, require very similar amounts of computation. Because of its accuracy and simplicity, we recommend using the UKF over the EKF whenever possible in your projects. If you must use the EKF, our advice is to use the error state formulation, be wary of linearization error, and take extra care to ensure your Jacobians are correct.

24 Questions

25 Assignements

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*