# Kalman Filter Interview

Harveen Singh
Mar 30, 2018 · 8 min read

I am currently into Term 2 of my Self Driving Car Nanodegree. Recently I met one of my colleagues Larry, who is a young developer and is very excited to be a part of self driving car industry. He asked me about my understanding of Kalman Filter. So here goes the conversation.

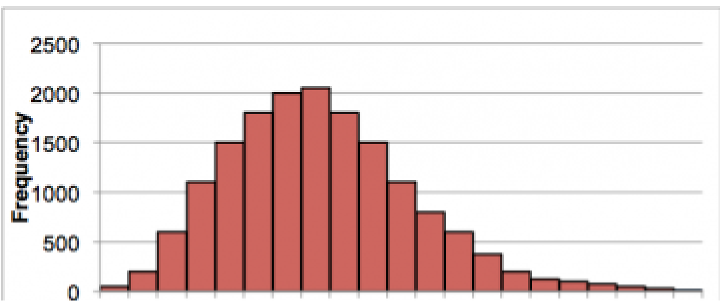
. . .

*Larry: What is a Kalman Filter?*
Me: A Kalman Filter is a tool that helps to predict values.

*Larry: Well that is cool! Means it is something sort of an astrologer?*
Me: Well, not quite. It is an iterative mathematical process that uses a set of equations and consecutive data inputs to estimate the values we are interested in associated with the object **quickly. Basically it is all quick mathematics!**

*Larry: I was not that good in Mathematics! Maybe this sounds a little bit absurd to me. Can you please go on?*
Me: Okay the first thing, kalman filters work with gaussians or normal distribution.

*Larry: Normal Distribution?*
Me: In a continuous graph data can be spread out in different ways: Either it can be spread to the left, or to the right or jumbled up.



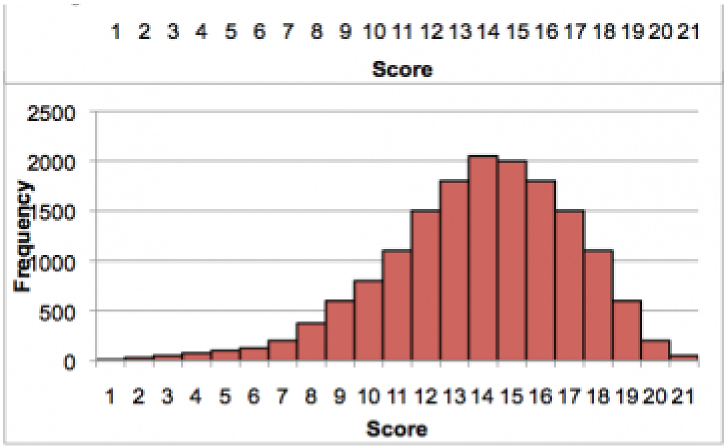



Figure 1. Example of left skewed and right skewed distribution (image source)

But there are many cases where data tends to be around a cental value with no bias left or right and the resulting distribution is called a normal distribution or a gaussian or a bell's curve.

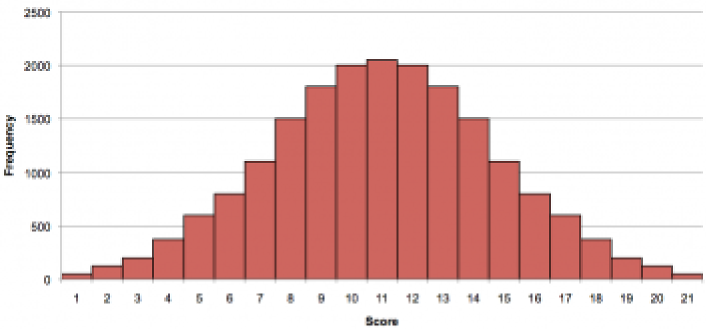



Figure 2. Example of a normal distribution, looks like a bell hence bell curve. (image source)

*Larry: Got it! As you pointed out this is a continuous curve and not a discrete valued. How do I find out say the probability of getting a 5 on a roll of dice?*
Me: Well, Good question, but to represent such distributions like probability of getting a dice we use binomial distribution as it is discrete valued. Here in normal distribution you need to define a range. Say, what is the probability of getting rain between 4.5 to 5.5 cm. So in this case we have a normal distribution, we mark the points and we calculate the area under these points which gives us the probability.

*Larry: Oh! So the area of complete graph will be 1 as the probability can have a maximum value of 1?*
Me: Absolutely right, additionally the mean, median and mode all are equal in case of this distribution.

*Larry: But you did not represented the Gaussian in mathematical terms?*
Me: So, to define a Gaussian, we have basically two things- mean and variance.

Mean($\mu$) you obviously know, Variance($\sigma^2$) basically tells how much the numbers are spread out and how far are they from mean. Standard Deviation($\sigma$) is just the square root of Variance($\sigma^2$) given by formula-:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

*Larry: So how is a Gaussian related to Kalman Filter?*
Me: Hmm! A Gaussian in a Kalman Filter represents the predicted value with noise/error/uncertainty in our prediction often known as the variance. The predicted value is centered around the mean with the width of the Gaussian denoting the uncertainty in our value. Basically it tells how much sure we are of a certain value to be true. More the width of the Gaussian denotes more uncertainty.

*Larry: Ohk! You played a bit of polymorphism game here. You mentioned earlier that it is an iterative process?*
Me: Yup, It is basically a two step process-
1. Predict
2. Update

In Predict we just predict the new value called **predicted value** based on the initial value and then predict the uncertainty/error/variance in our prediction according to the various process noises present in the system.

In Update, we take in account the actual measurement coming from the device and we call this as **measured value**. Here we calculate the difference between our predicted value and measured value and then decide which value to keep by calculating the Kalman Gain. We then calculate the new value and new uncertainty/error/variance based on our decision made by Kalman Gain. These calculated values will finally be the predictions done by our Kalman Filter in iteration 1.

The output of the update step is again fed into the Predict State and the cycle goes on until the error/uncertainty between our predicted and actual values tends to converge to zero.

*Larry: Well, that was pretty fast. Can you please explain in terms of any example or a flowchart?*
Me: Ok, so from now on these notations will be followed:

> *x -> Mean*
>
> *P -> Variance*

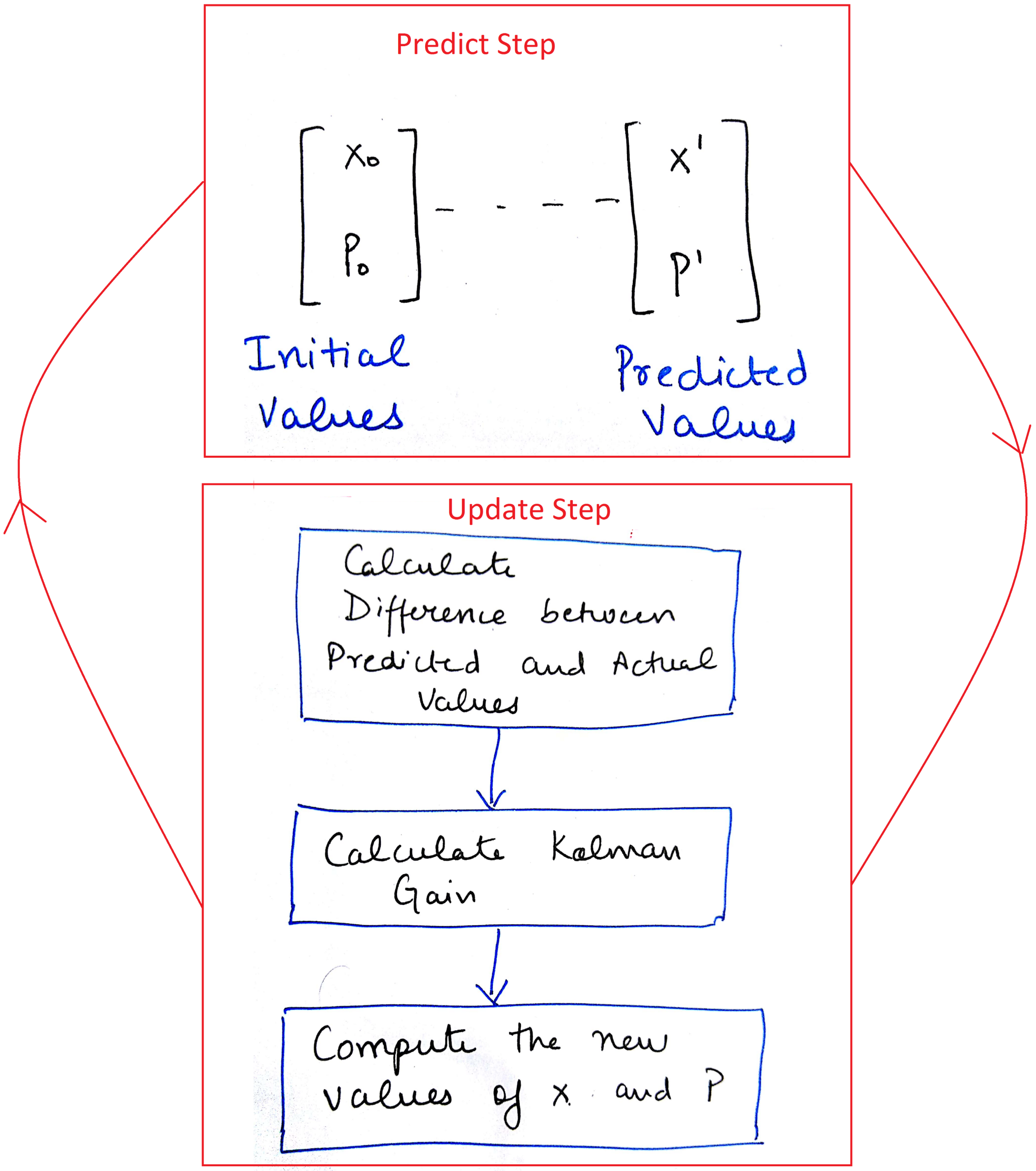



Figure 3. A Rough Flowchart for Kalman Filter

*Larry: Seems Intuitive! But what the hell is Kalman Gain?*
Me: Kalman Gain is a parameter which decides how much weight should be given to predicted value and measured value. It is a parameter that decides whether our actual value is close to predicted value or measured value.

*Larry: But how does it know how to trust either the predicted value or actual value?*
Me: It checks the uncertainty/error my friend.

> *K = Error In Prediction / (Error in Prediction + Error in Measurement)*

The value of the K ranges from 0 to 1. If we have a large error in measurement, K is nearer to 0,this means our predicted value is close to actual value. If we have a large error in prediction, K is closer to 1, this means our measured value is closer to actual value.

*Larry: Ok, Agreed! I am getting excited now. Can you please elaborate on the equations now?*
Me: Sure. I will write it down for you differently for the prediction and update step. Remember these equations are for 2D space.

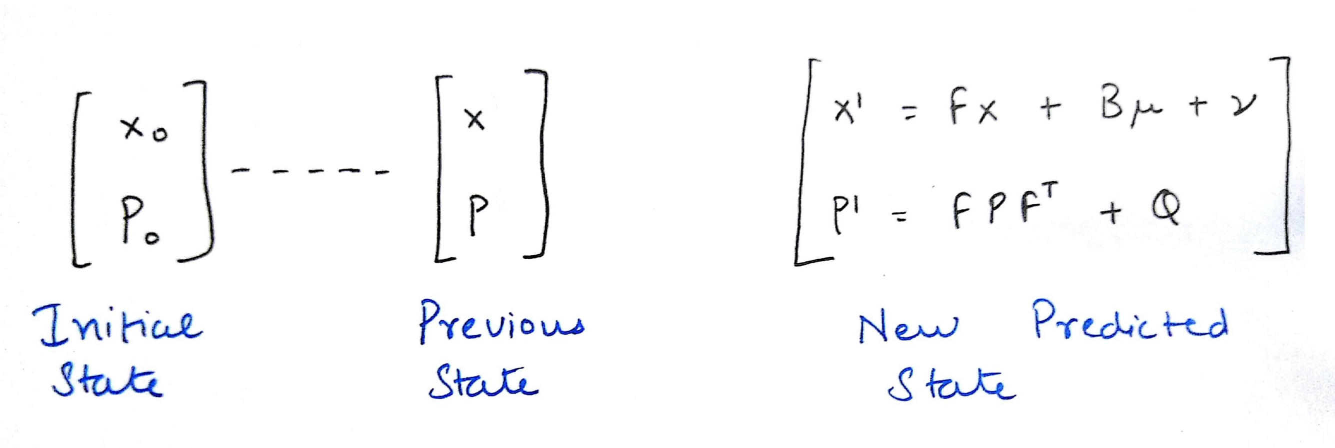



Figure 4. Equations for Predict Step

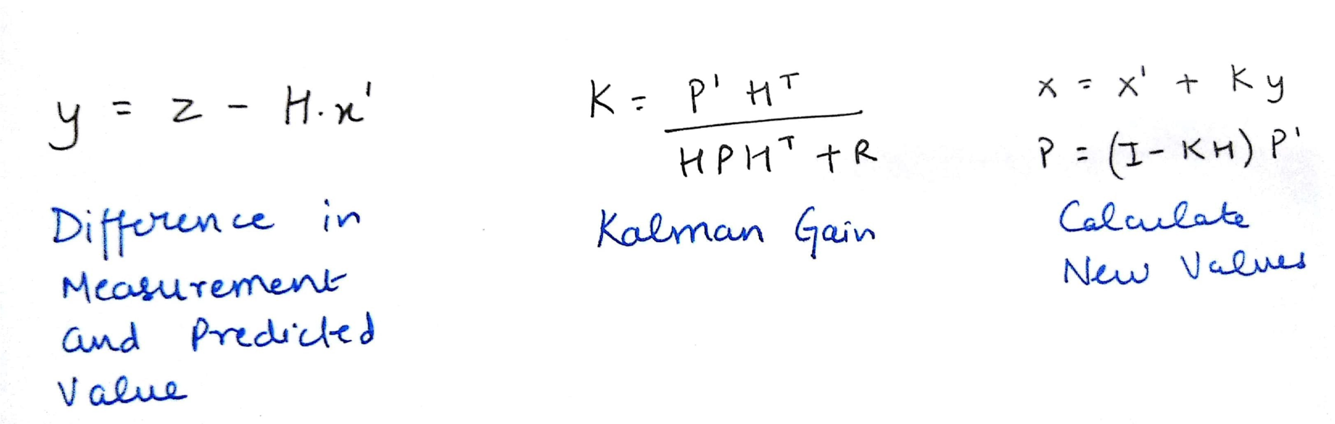



Figure 5. Equations for Update Step

. . .

*Larry: At this step, probably we should end our discussion. What is F, B, H? and there are all sorts of variables messed up. You said that's all for kalman filter, I understood the functionality but why all these unknowns now?*
Me: Don't Panic. I will explain everything in detail now.

Suppose we are want to predict the position and velocity of car from the measurements coming from different sensors.

> *x -> mean state vector containing position and velocity.*
> *P -> covariance matrix (denoting error).*

$$\begin{bmatrix} p \\ v \end{bmatrix}$$

$$p- > Distance$$
$$v- > Velocity$$

x vector

# Prediction Step

> *Equation 1-:*
>
> ***x′ = F.x + B.μ + v***
>
> *x' -> Predicted Value*
> *F -> State Transition Matrix*
> *B -> Control Input Matrix*
> *μ -> Control Vector*
> *v -> Process Noise*

## F Matrix

F is a state transition matrix or an adaptable matrix that are required to convert matrix from one form to the other. For example- say we have a model where we are predicting the position and velocity of the object which is not accelerating. So in this case the new p and v after a time delta t is given as-:

$p' = p + v\Delta t$

$v'=v$

So in this case the F matrix will be:

$$\begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}$$

F Matrix

### B Matrix

B is the control input matrix that denotes the change in state of object due to internal or any external force. For example- the force of gravity or the force of friction to the object.

### Why B.μ = 0?

Mostly, in the context of autonomous cars the value of control product vector is equal to zero because we cannot model the external forces acting on objects from the car.

### ν

This is the noise in the process. We add random noise that might be present in the channel to make our prediction a little correct.

> *Equation 2-:*
>
> ***P′ = FPF$^T$ + Q***
>
> *P' -> Predicted Covariance*
> *F$^T$-> Transpose of State Transition Matrix*
> *Q -> Noise*

### Q Matrix

We assume that the object changed direction or maybe accelerated or decelerated. So after a time Δt, our uncertainty increases by a factor of Q which is again noise. *So we add noise into noise technically.*

So in the prediction step we get the two predicted values x′ and P′.

## Update Step

> *Equation 1:*
>
> ***y= z - H.x′***
>
> *z -> actual measurement*
> *H -> State Transition Matrix*
> *x' -> Predicted Value*
> *y -> Difference between Measured Value and Actual Value*

### z

This is the actual **measured value** that is coming from the sensor.

### H

This is again a state transition matrix. With H, we can discard information from the state variable that we do not require. Technically H is doing the same work what F was doing in Prediction Step.

> *Equation 2:*
>
> $S = HP'H^T + R$
> $K = P'H^TS^{-1}$
>
> *R -> Measurement Noise*
> *K -> Kalman Gain*
> *S-> Total Error*
> *$S^{-1}$ -> The inverse of S*

### R

R denotes the noise in the measurement. *What? So those devices are not 100% accurate?* Yup, nothing is perfect in this world and not even the devices that measure the values. All devices comes with a predefined value for R parameter that is given by the manufacturer, this value always remains constant throughout the cycle.

### K

We have a complex equation here but it is very simple. We are calculating the Kalman Gain K, formula for which was given earlier.

### S

This is the total error in the system. The error in our prediction plus the error in measurement.

*So why so complicated equation for K as earlier in the formula it was simple?*
This is because we do not have a notion of division for matrices. Hence we opt to calculate the total error first and then multiply the error in our prediction with the inverse of total error.

> *Equation 3:*

> *x = x′ + K.y*
> *P = (I- KH)P′*

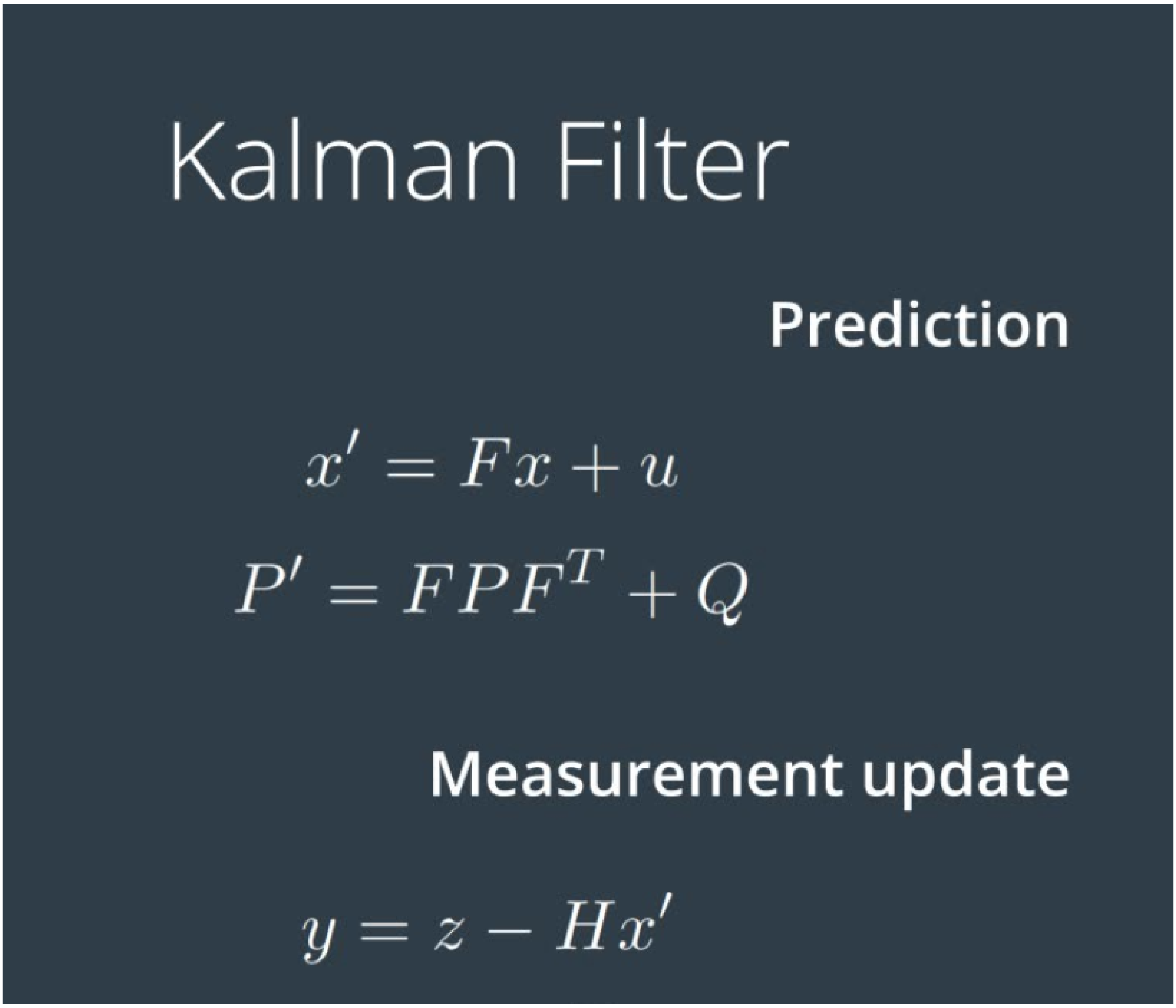
### Final Step

This is the final step where we update our x and P according to the calculations done by the Kalman Gain. Note- On the LHS, we have x and P and not x′ and P′ because we are now setting x and P for the next prediction step, hence we need to find their values.

*Larry: Well, this will take some time to digest! What are the additional resources I must go through to understand it completely?*
Me: The universe thanks Michel van Biezen for his wonderful contribution to Kalman Filters through his YouTube channel. Watch his videos for more examples and insights.

. . .

To summarize all the equations-:



$$S = HP'H^T + R$$

$$K = P'H^TS^{-1}$$

$$x = x' + Ky$$

$$P = (I - KH)P'$$

Figure 6. Taken from Udacity Nanodegree

## To be continued….(EKF and UKF in the store)

Note for Readers- If you notice any mistake in this article, please feel free to comment or leave a private note.

## Update 1- Extended Kalman Filter

## Update 2- Unscented Kalman Filter

Robotics Udacity Self Driving Cars Artificial Intelligence Kalman Filter