Extended Kalman Filter simplified— Udacity's Self-driving Car Nanodegree

Dhanoop Karunakaran Follow
Feb 9, 2018 · 7 min read

We can use a Kalman filter in any place where we have uncertain information about some dynamic system, and It can make an educated guess about what the system is going to do next. Is it interesting?

Here we go the another interesting facts:

> *In the 1960s, the Kalman filter was applied to navigation for the Apollo Project, which required estimates of the path of manned spacecraft going to the Moon and back. Also today it helps commercial airline pilots get from New York to Seattle without winding up in, say, San Diego.*

I have come across Kalman Filter and Extended Kalman Filter algorithms as part of project in term 1 of Udacity's Self-driving Car nanodegree. The goal of the project is to write Kalman filter and Extended kalman filter algorithms for accurately tracking bicycle's position and velocity. Udacity will be providing simulated lidar and radar measurements detecting a bicycle that travels around the vehicle.

Expectation of this article to give you the concept of Kalman Filter and Extended Kalman Filter algorithms in detail.

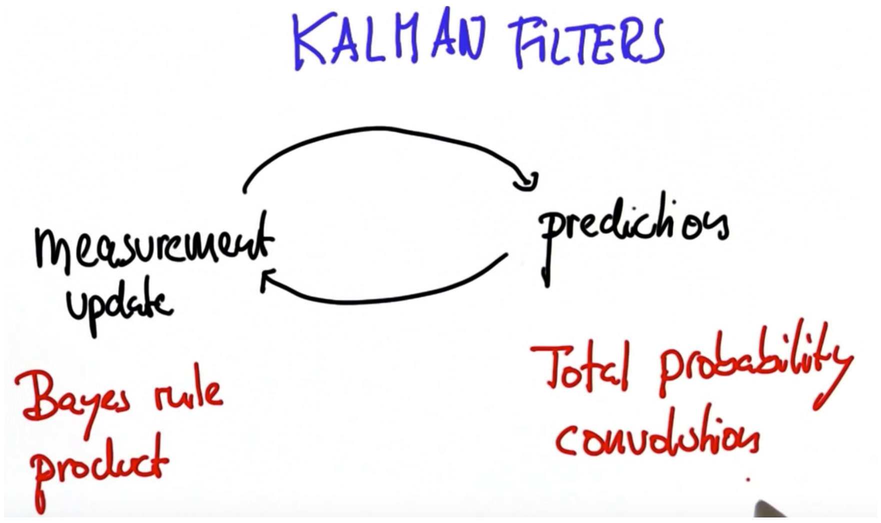
What is Kalman Filter?

A Kalman filte**r** is an optimal estimation algorithm used to estimate states of a system from indirect and uncertain measurements. A Kalman filter is only defined for linear systems. If you have a nonlinear system and want to estimate system states, you need to use a nonlinear state estimator.

Kalman filter is extensively used in estimating the car position using Lidar and Radar sensors. Measurement of these sensors are not accurate as they are subject to drift or noisy. Kalman filter can be used to fuse the measurement of these sensors to find the optimal estimation of the exact position.

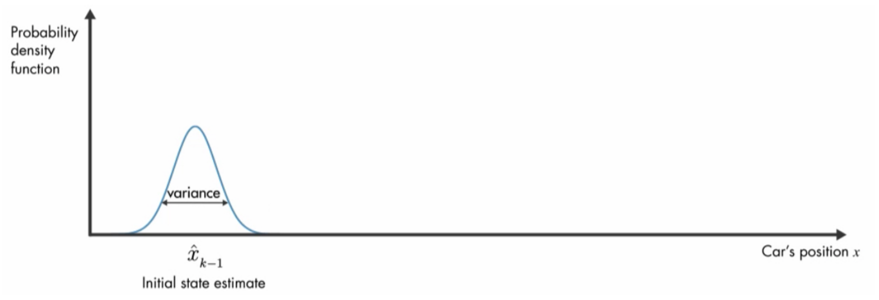
Kalman Filter Cycle

The Kalman filter represents our distributions by Gaussians and iterates on two main cycles: Prediction and Measurement update.

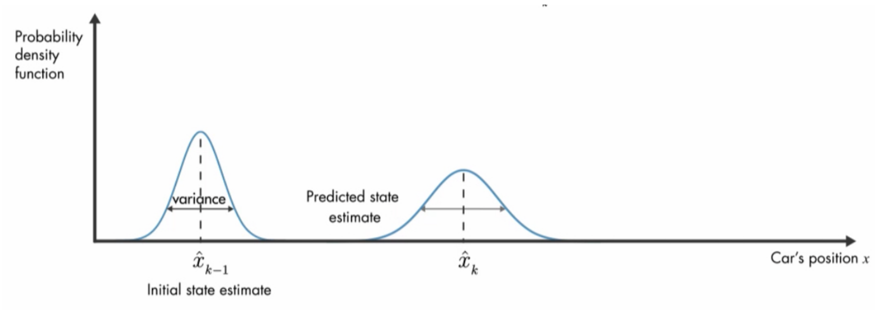


From Udacity lecture

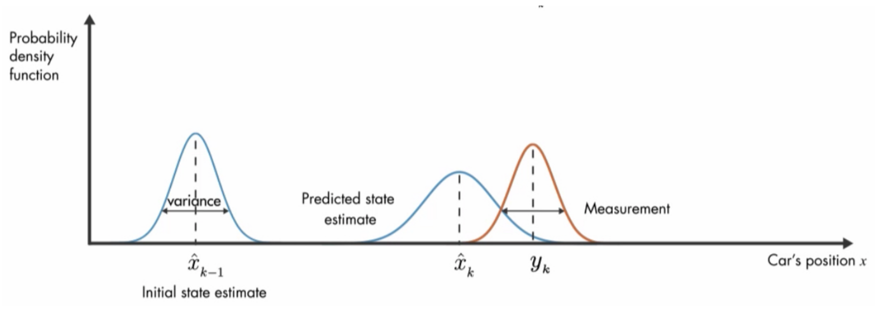
Before getting into the details of the Kalman filter algorithm cycle, consider below example of estimating the car position based on the prediction result and measurement result using Kalman filter. We can explain the working principles of kalman filters with the help of probability density function as shown below.



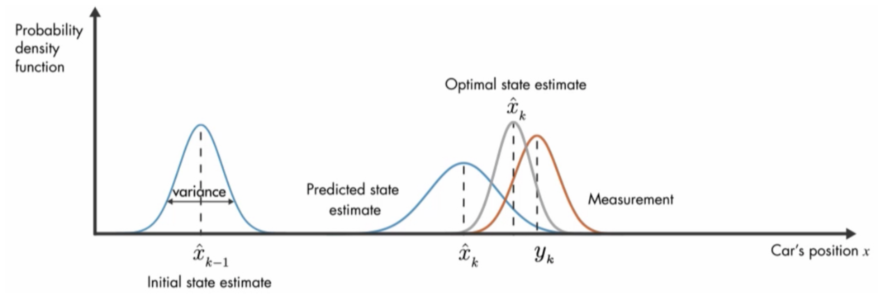
At initial time step **k-1**, car position can be in the mean position of gaussian distribution at ***xk*-1** position. At the next time step K, uncertainty at ***xk*** position increases which is shown below with larger variance.



Next one is the measurement result **y** from the sensors and noise represented in the variance of the third gaussian distribution as shown below.



Kalman filter says optimal way to estimate the position is combining these two results. This is done by multiplying these two probability functions together and the result will also be another gaussian function.



Let's get into the two cycles of Kalman filters: Prediction and Measurement update.

Prediction

Let's say we know an object's current position and velocity , which we keep in the x variable. Now one second has passed. We can predict where the object will be one second later because we knew the object position and velocity one second ago; we'll just assume the object kept going at the same velocity.

$$x' = Fx + \nu$$

But maybe the object didn't maintain the exact same velocity. Maybe the object changed direction, accelerated or decelerated. So when we predict the position one second later, our uncertainty increases.

$$P' = FPF^T + Q$$

> ***x** is the mean state vector. For an extended Kalman filter, the mean state vector contains information about the object's position and velocity that you are tracking. It is called the "mean" state vector because position and velocity are represented by a gaussian distribution with mean x.*

> ***P** is the state covariance matrix, which contains information about the uncertainty of the object's position and velocity. You can think of it as containing standard deviations.*

> ***Q*** *is the Process Covariance Matrix. It is a covariance matrix associated with the noise in states. I didn't specify the equation to calculate the matrix here.*

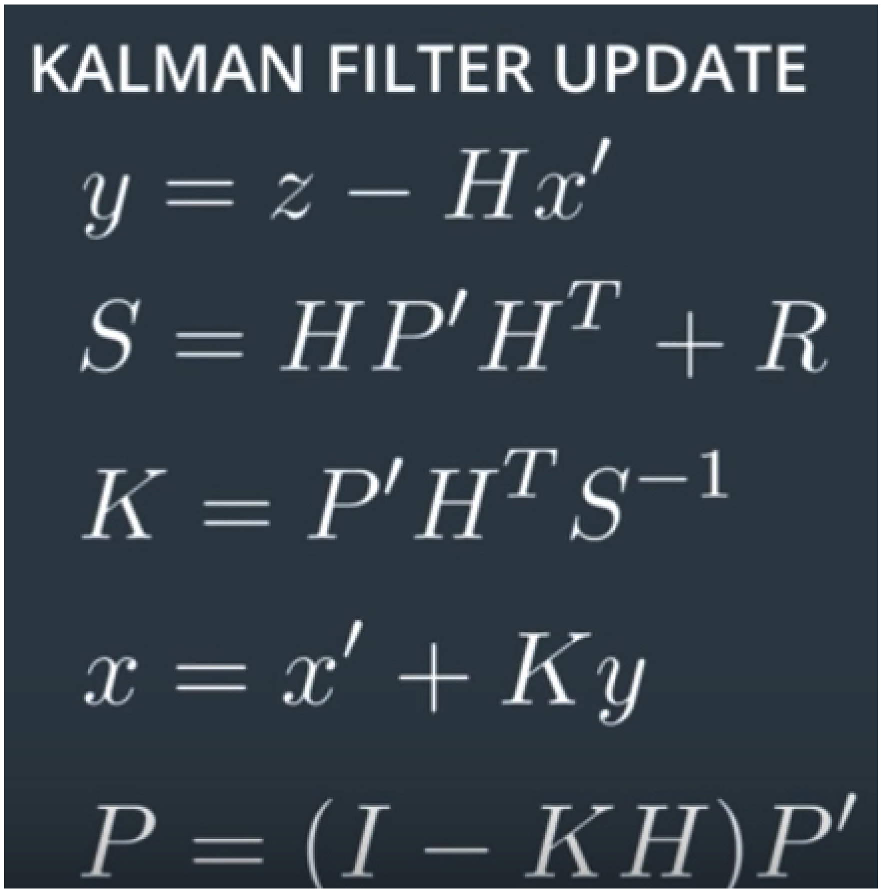
> ***F*** *is the Transition Matrix (the one that deals with time steps and constant velocities)*

Measurement Update

We are going to obtain the data from two kind of measurements (sensors):

- Laser Measurements
- Radar Measurements

Laser Measurement allows us to get the current position of the object but not its velocity. On the other hand Laser Measurements have more resolution.



KALMAN FILTER UPDATE

$$y = z - Hx'$$

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

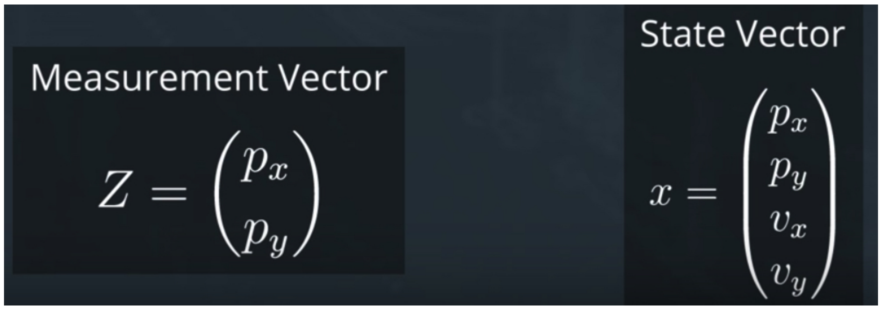
$$K = P'H^TS^{-1}$$

$$x = x' + Ky$$

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

From Udacity lecture

As shown above, this is the equation for finding the x and P. Measurement and state vector representation is shown below.



Measurement Vector

$$Z = \begin{pmatrix} p_x \\ p_y \end{pmatrix}$$

State Vector

$$x = \begin{pmatrix} p_x \\ p_y \\ v_x \\ v_y \end{pmatrix}$$

> ***z*** *is the measurement vector. For a lidar sensor, the* z *vector contains the* position−x *and* position−y *measurements.*

> ***H*** *is the matrix that projects your belief about the object's current state into the measurement space of the sensor. For lidar, this is a fancy way of saying that we discard velocity information from the state variable since the lidar*

> *sensor only measures position: The state vector* x *contains information about* [px, py, vx, vy] *whereas the* z *vector will only contain* [px, py]*. Multiplying Hx allows us to compare x, our belief, with z, the sensor measurement.*

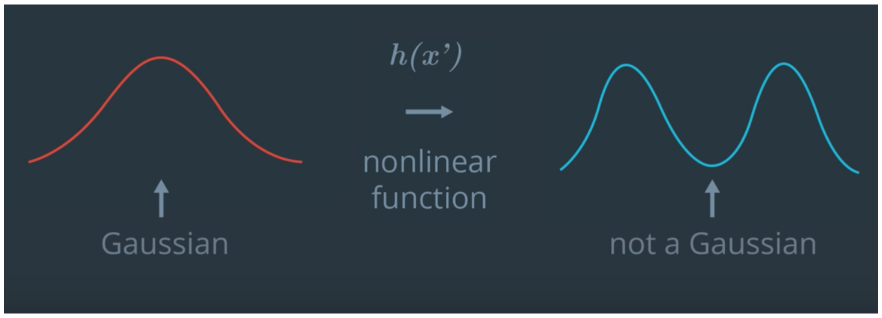
> *R is just the covariance matrix of the measurement noise. I didn't specify the equation to calculate the matrix here.*

Radar Measurement goes further and allows us to get the velocity information as well. Measurement update part of the radar will explain in Extended Kalman Filter section.

What is Extended Kalman Filters?

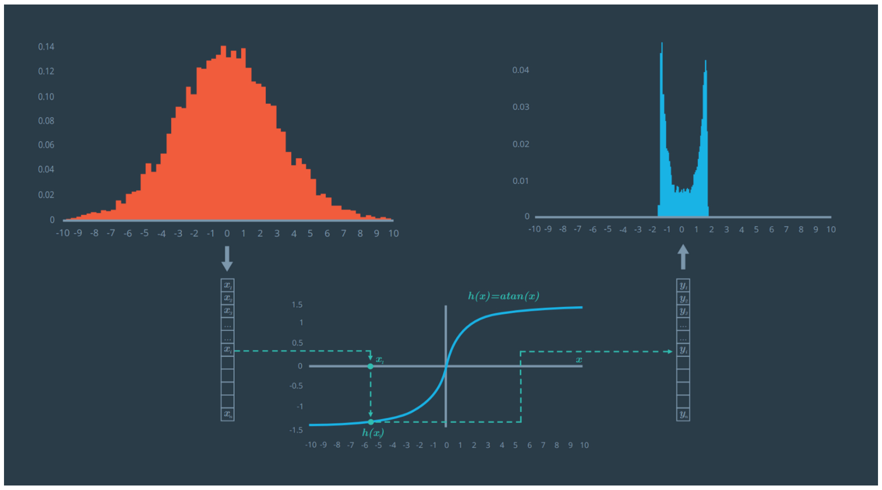
Extended Kalman Filters(EKF) linearize the distribution around the mean of the current estimate and then use this linearization in the predict and update states of the Kalman Filter algorithm.

Our predicated state **x** is represented by gaussian distribution which is mapped to Nonlinear function(Radar measurement), then resulting function will not be a gaussian function.



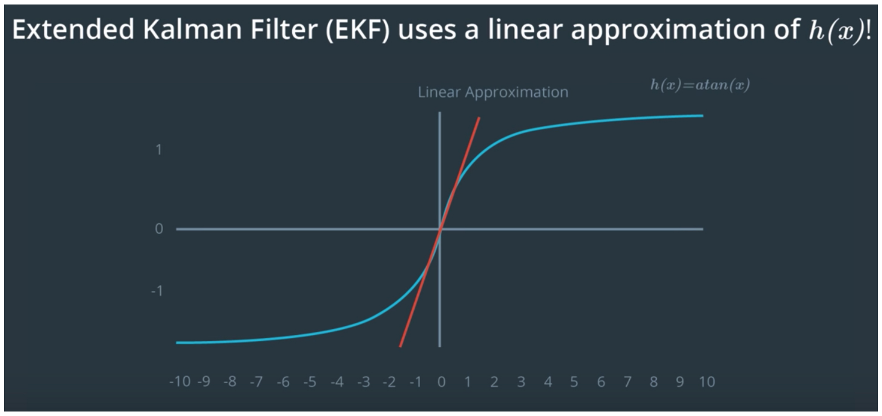
From Udacity lecture

So Kalman filter is not applicable any more as it works with Gaussian functions only.



From Udacity lecture

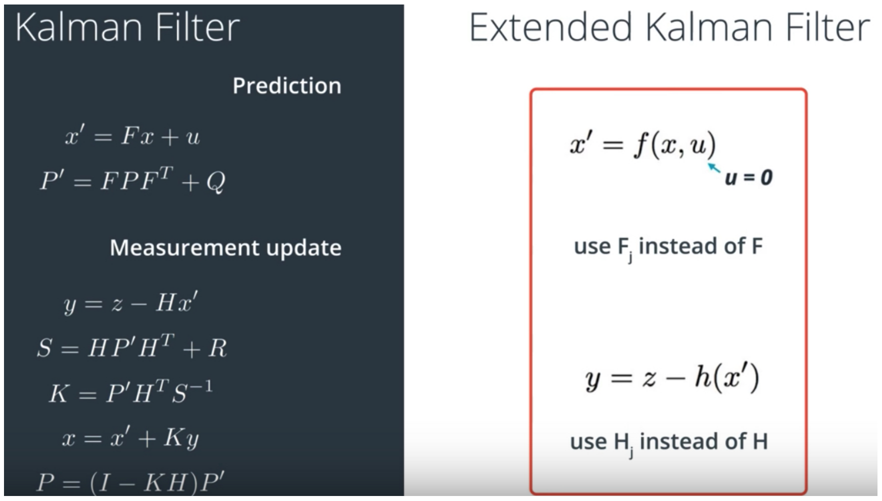
One of the solution is to linearise the h(x) function and that's the key of EKF.



From Udacity lecture

EKF uses a method called **First Order Taylor Expansion** to linearise the function.

Although the mathematical proof is somewhat complex, it turns out that the Kalman Filter equations and EKF equations are very similar.



Kalman Filter

Prediction

$$x' = Fx + u$$
$$P' = FPF^T + Q$$

Measurement update

$$y = z - Hx'$$
$$S = HP'H^T + R$$
$$K = P'H^TS^{-1}$$
$$x = x' + Ky$$
$$P = (I - KH)P'$$

Extended Kalman Filter

$$x' = f(x, u)$$

u = 0

use F_j instead of F

$$y = z - h(x')$$

use H_j instead of H

From Udacity lecture

The main differences are:

- The *F* matrix will be replaced by *Fj* when calculating *P′*.
- The *H* matrix in the Kalman filter will be replaced by the Jacobian matrix *Hj* when calculating *S*, *K*, and *P*.
- To calculate *x′*, the prediction update function, *f*, is used instead of the *F* matrix.
- To calculate *y*, the *h* function is used instead of the *H* matrix.

There are few drawbacks to EKF:

- It is difficult to calculate jacobians if they need to be found analytically.
- There is a high computational cost if the jacobians found numerically.
- EKF only works on the system that have a differentiable model.

Coming back to the measurement update Part we explained before. As **Radar measurement** is non-linear, we need to use EKF on M**easurement Update** cycle, but we can use the same formula in prediction cycle.

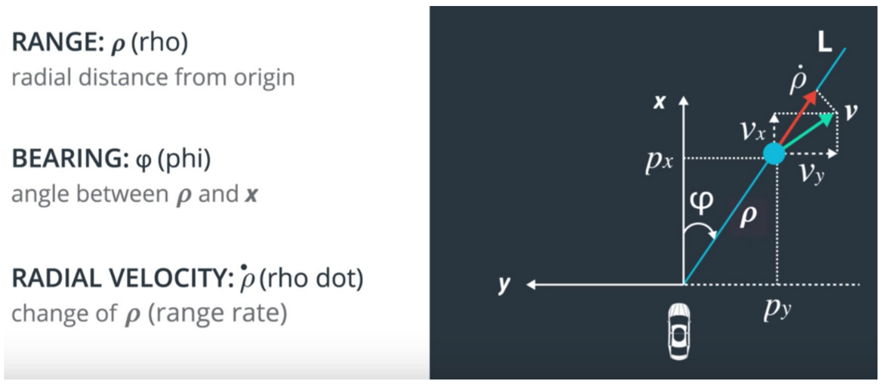
The *H* matrix from the laser measurement and $h(x)$ equations from the radar measurement are actually accomplishing the same thing; they are both needed to solve $y=z-Hx'$ in the update step.

But for radar, there is no *H* matrix that will map the state vector *x* into polar coordinates; instead, you need to calculate the mapping manually to convert from cartesian coordinates to polar coordinates.

Here is the *h* function that specifies how the predicted position and speed get mapped to the polar coordinates of range, bearing and range rate.

$$h(x') = \begin{pmatrix} \rho \\ \phi \\ \dot{\rho} \end{pmatrix} = \begin{pmatrix} \sqrt{p'^2_x + p'^2_y} \\ \arctan(p'_y / p'_x) \\ \frac{p'_x v'_x + p'_y v'_y}{\sqrt{p'^2_x + p'^2_y}} \end{pmatrix}$$

Hence for radar $y = z - Hx'$ becomes $y = z - h(x')$.

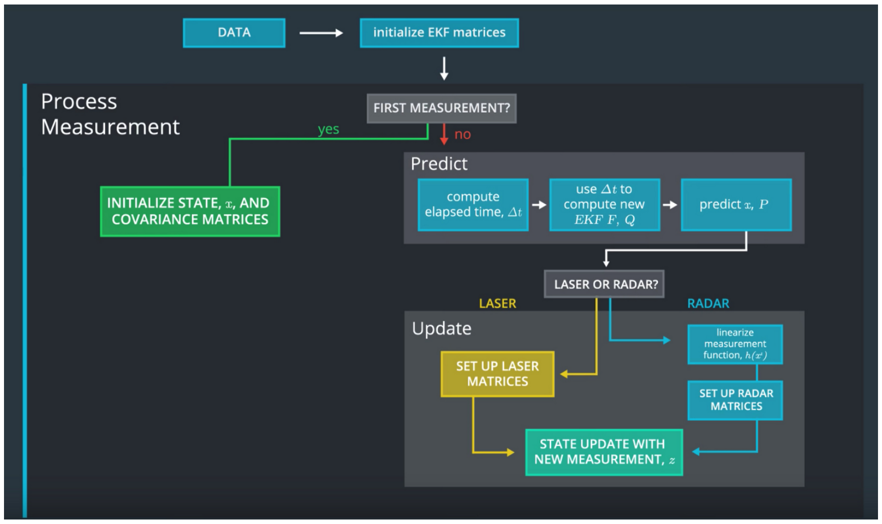


> *The range, (ρ), is the distance to the pedestrian. The range is basically the magnitude of the position vector ρ which can be defined as ρ=sqrt(px2 +py2).*

> *φ=atan(py/px). Note that φ is referenced counter-clockwise from the x-axis, so φ from the video clip above in that situation would actually be negative.*

> *The range rate, ρ˙, is the projection of the velocity, v, onto the line, L.*

And here is the High level architecture of EKF project of Udacity's Self-driving Car nanodegree.


DATA
initialize EKF matrices
Process Measurement
FIRST MEASUREMENT?
yes
no
INITIALIZE STATE, x, AND COVARIANCE MATRICES
Predict
compute elapsed time, Δt
use Δt to compute new $EKF\ F,\ Q$
predict $x,\ P$
LASER OR RADAR?
LASER
RADAR
Update
SET UP LASER MATRICES
linearize measurement function, $h(x')$
SET UP RADAR MATRICES
STATE UPDATE WITH NEW MEASUREMENT, z

High level architecture from Udacity lecture

If you would like to see the code in action, visit my github project page below.

dkarunakaran/carnd-extended-kalman-filter-term2-p1

carnd-extended-kalman-filter-term2-p1 — Kalman filtering, also known as linear quadratic estimati…

github.com