The Unscented Kalman Filter: Anything EKF can do I can do it better!

Harveen Singh

Apr 27, 2018 · 9 min read

I have just completed my Term 2 of Udacity Self Driving Car Nanodegree. I wrote about Kalman Filter and Extended Kalman Filter. Today we will look at another member of Kalman Filter Family: The Unscented Kalman Filter. So, if you read my last two posts you would be knowing my colleague Larry by now.

Summary:

Kalman Filter: *It is a tool to predict values using a bunch of mathematical equations under the assumptions that our data is in the form of Gaussian Distribution and we apply linear equations to that Gaussian distribution.*

Extended Kalman Filter: *In real world, we have non linear equations, because we may be predicting in one direction while our sensor is taking reading in some other direction, so it involves angles and sine cosine functions which are non linear. So EKF takes helps of Taylor Series (and Jacobian Matrix further) to linearly approximate a non linear function around the mean of the Gaussian and then predict the values.*

Unscented Kalman Filter

Larry: I know about Kalman Filter and Extended Kalman Filter, now what? I know the reason why Kalman Filter failed in real life and the need of Extended Kalman Filter. Now why Unscented Kalman Filter?
Me: Performance.

Larry: Performance? How come?
Me: How many points we took in EKF to approximate a new linear function from non linear function?

Larry: 1 point, that is the mean of the Gaussian.
Me: Correct, so is there a better way to linearize?

Larry: What do you mean?
Me: Have a look below what happened in EKF:

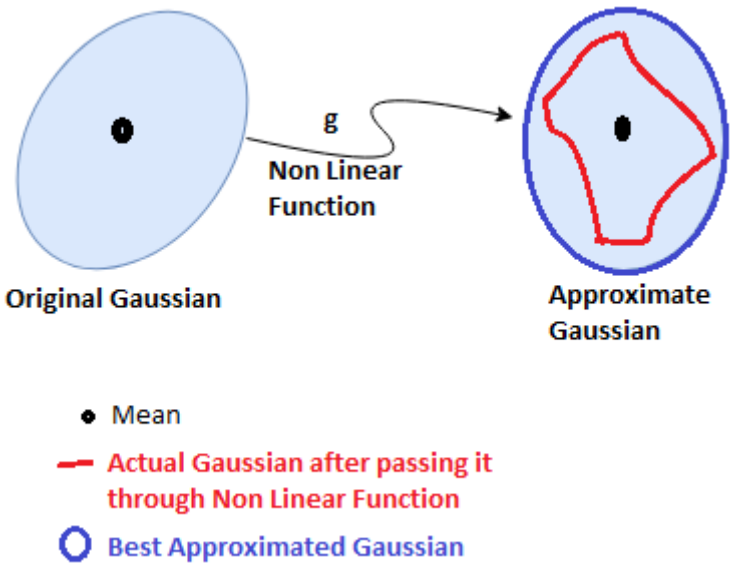


Figure 1. Scenario of Gaussian Approximation in EKF around the mean

We have just one point to approximate the Gaussian. So, is there a better way to linearize?

Larry: If I would have known that, I would not be talking to you. Tell me!
Me: What do you think will give us a better approximation? Suppose we have two scenarios to reach from a Source Gaussian to an Approximated Gaussian-:
Scenario 1: We have one point (say mean)and we approximate around one point.
Scenario 2: We have a bunch of points including the mean and we approximate around those multiple points.

Larry: My Intuition says if we have multiple points as in case of scenario 2, we will have a better approximation!
Me: Congrats! You now know the Unscented Kalman Filter.

Larry: So in that case why don't you consider all the points in source Gaussian and then transform and then approximate?
Me: That will take a lot of computational power and resources, so it may be the most trivial solution but it is not optimal.

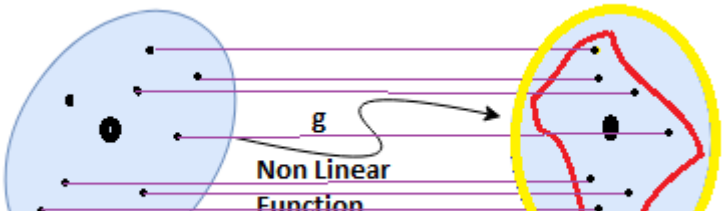
Sigma Points

Larry: Oh!! So how do we go about choosing the right number of points?
Me: So in Unscented Kalman Filter we have a concept of **Sigma Points. We take some points on source Gaussian and map them on target Gaussian after passing points through some non linear function and then we calculate the new mean and variance of transformed Gaussian.**

It can be very difficult to transform whole state distribution through a non linear function but it is very easy to transform some individual points of the state distribution, these individual points are sigma points. These sigma points are the representatives of whole distribution.

Basic Difference between EKF and UKF

Here the main difference from EKF is that in EKF we take only one point i.e. mean and approximate, but in UKF we take a bunch of points called sigma points and approximate with a fact that more the number of points, more precise our approximation will be!



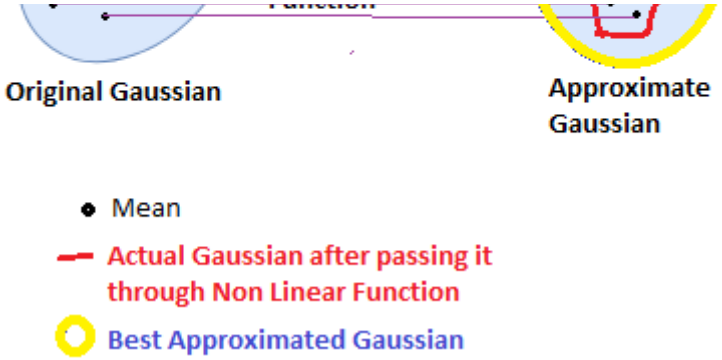


Figure 2. Scenario of Gaussian Approximation around the Mean and other sigma points

Larry: Great! Got it! Its so simple.
Me: Well, that's not the case, in addition to sigma points, these points also have weights, so these are **weighted sigma points**.

. . .

Unscented Transform

Larry: So, in that case we are giving more or less preference to some points to make our approximation better?
Me: Yup, that's correct.
When a Gaussian is passed through a non linear function, it does not remains a Gaussian anymore but we approximate the Gaussian from the resulting figure, so in UKF a process called **Unscented Transform** helps us to perform this task. To summarize here are the below steps the unscented transform performs:
1. Compute Set of Sigma Points
2. Assign Weights to each sigma point
3. Transform the points through non linear function
4. Compute Gaussian from weighted and transformed points
5. Compute Mean and Variance of the new Gaussian.

Computing Sigma Points

Larry: But how do we choose the Sigma Points?
Me: The number of sigma points depend on the dimentionality of the system. The general formula is **2N + 1**, where N denotes the dimensions.

$$\begin{aligned} \mathcal{X}^{[0]} &= \mu \\ \mathcal{X}^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\,\Sigma}\right) \qquad \text{for } i = 1, \ldots, n \end{aligned}$$

$$\mathcal{X}^{[i]} = \mu - \left(\sqrt{(n+\lambda)\,\Sigma}\right)_{i-n} \quad \text{for } i = n+1, \ldots, 2n$$

Figure 3. Choosing Sigma Points Equations

> *χ(Caligraphic X) -> Sigma Points Matrix*
> *μ -> mean of the Gaussian*
> *n-> dimentionality of system*
> *λ-> Scaling Factor*
> *Σ-> Covariance Matrix*

χ (Caligraphic X)

χ denotes the Sigma Point Matrix. An important thing to note here is that every column of χ denotes a set of sigma points. So if we are working in 2 dimensions, then the size of χ matrix will be 2 X 5. As we have 5 number of sigma points for each dimension.

λ

λ is the scaling factor which tells how much far from mean we should choose our sigma points. A good mathematical study suggests the optimal value of λ to be 3-n.

Obviously one of the sigma points is the mean, and the rest we calculate based on the above equations.

Larry: Wait a minute, Square root of a matrix? I never heard of that!
Me: Yup. Square of the matrix is defined if we have a matrix S that satisfies the following condition:

> *Σ = S.S or Σ = S.S_transpose*

If we are able to find S then we can say that S= √∑

. . .

Computing Weights of Sigma Points

Larry: Oh! that is a new thing to know and what about the weights assigned to these calculated sigma points?
Me: Oh, the weights too have equations:

$$w^{[0]} = \frac{\lambda}{n+\lambda}$$

$$w^{[i]} = \frac{1}{2(n+\lambda)} \quad \text{for } i = 1, \ldots, 2n$$

Figure 4. Calculating Weights of the Sigma Points

Calculating weight of the mean has a different equation than the rest of the sigma points. λ is the spreading parameter and n is the dimentionality. **An interesting point to note here is that sum of all the weights is equal to 1.**

. . .

Computing Mean and Covariance of the approximate Gaussian

Larry: Oh man! So many equations! So we have now sigma points, weights so how do we recover the gaussian after it passes from the non linear function g?
Me: Again a bunch of equations :D

$$\mu' = \sum_{i=0}^{2n} w^{[i]}\, g(\mathcal{X}^{[i]})$$

$$\Sigma' = \sum_{i=0}^{2n} w^{[i]}\, (g(\mathcal{X}^{[i]}) - \mu')(g(\mathcal{X}^{[i]}) - \mu')^T$$

Figure 5. Calculating New Mean and Covariance

> *μ' -> Predicted Mean*
> *Σ' -> Predicted Covariance*
> *w -> Weights of sigma points*
> *g -> Non Linear function*

> *χ(Caligraphic X) -> Sigma Points Matrix*
> *n -> Dimentionality*

So this was all about the Unscented Transform and how it works.

. . .

Prediction Step

Larry: Hmm, so how to use Unscented Transform in our General Predict-Update model of Kalman Filter?
Me: So the Predict Step is basically very close to what we discussed just now i.e. the Unscented Transform.

1. **Calculate Sigma Points-** using equations in Figure 3.
2. **Calculate Weights of Sigma Points-** using equations in Figure 4.
3. **Transforming Sigma Points and Calculate new Mean and Covariance-** It is very close to the equations mentioned in Figure 5. Whenever we predict what happens? Our uncertainty increases by some amount, because we become a little bit uncertain so we have to take in account the process noise.

$$\mu' = \sum_{i=0}^{2n} w^{[i]}\, g(\mathcal{X}^{[i]})$$

$$\Sigma' = \sum_{i=0}^{2n} w^{[i]}\, (g(\mathcal{X}^{[i]}) - \mu')(g(\mathcal{X}^{[i]}) - \mu')^T + R_t$$

Figure 6. Predicted mean and covariance after accounting in Noise R

. . .

Update Step

Larry: Now we have the predicted mean and covariance. So in the Update step suppose we have a measurement coming from the sensor so how we compute the difference between our predicted values of mean and covariance and actual values of mean and covariance?

Me: Well, the procedure is quite similar to the one used in Kalman Filter. So what we do here we take our predicted state to the measurement state.

Now here we have an option we can generate the sigma points again because the predicted mean and variance changed and sigma points somehow depend on them or we just continue with the same set of sigma points we generated earlier. For the time being lets take the sigma points we generated earlier only.

We take our state from our state space to measurement state space.

$$\mathcal{Z} = h(\mathcal{X})$$

$$\hat{z} = \sum_{i=0}^{2n} w^{[i]} \mathcal{Z}^{[i]}$$

$$S = \sum_{i=0}^{2n} w^{[i]} (\mathcal{Z}^{[i]} - \hat{z})(\mathcal{Z}^{[i]} - \hat{z})^T + Q$$

Figure 7. Update Step considering the measurement spacce

> *Z -> transformed sigma points in **measurement** space*
> *χ(Caligraphic X) -> Sigma Points Matrix*
> *$\hat{Z}$ -> Mean in measurement space*
> *S -> Covariance in measurement space*
> *Q-> Noise*
> *h-> is a function that maps our sigma points to measurement space*

Important: Z is our measurement space i.e. the measurement that is coming from the sensor. So we need a function h which can transform our state space to measurement space so that we can equate them in same units.

Larry: Great! and how do we compute the Kalman Gain here? ***I see we do not have a Jacobian anymore here because we are not linearizing the function here!***
Me: That is a very important concept, we are not linearizing the function anymore! Regarding the Kalman Gain, there is a bit of change here.

To calculate Error in Prediction: We need to calculate the cross-correlation between sigma points in state space and sigma points in the measurement space.

$$T = \sum_{i=0}^{2n} w^{[i]}(\mathcal{X}^{[i]} - \mu')(\mathcal{Z}^{[i]} - \hat{z})^T$$

$$K = T.S^{-1}$$

Figure 8. Calculating Kalman Gain

> *T -> Cross Co-relation Matrix between state space and predicted space*
> *S-> Predicted Covariance Matrix*
> *K-> Kalman Gain*

Larry: Ohh man! The equations changed a lot!
Me: If you observe closely it is not! Have a look below:

$$S = H_j P' H_j^T + R$$
$$K = P' H_j^T S^{-1}$$

⇒ *EKF*

$$T = \sum_{i=0}^{2n} w^{[i]}(\mathcal{X}^{[i]} - \mu')(\mathcal{Z}^{[i]} - \hat{z})^T \Rightarrow P' H_j^T$$
$$S = \sum_{i=0}^{2n} w^{[i]}(\mathcal{Z}^{[i]} - \hat{z})(\mathcal{Z}^{[i]} - \hat{z})^T + Q$$
$$K = T.S^{-1}$$

UKF

Figure 9. Comparison of Calculating Kalman Gain in EKF and UKF

Larry: Yup, the resemblance is the same ! And even the equations for calculating final state will be same as well?

Me: Yup, almost same.

$$\mu = \mu' + K(z - \hat{z})$$
$$\Sigma = (I - K\,T\,)\Sigma'$$

Figure 10. Predicting Final State after calculating Kalman Gain

> *μ -> Mean*
> *Σ -> Covariance*
> *μ' -> Predicted Mean*
> *Σ' -> Predicted Covariance*
> *K -> Kalman Gain*
> *z-> Actual Measurement Mean coming from the sensor*
> *$\hat{Z}$ -> Mean in measurement space*
> *T -> It is the same as H in Kalman Filter and H_j in EKF. Here it is cross co-relation matrix*

Larry: One last question. Why it is called Unscented Kalman Filter?
Me: Well, the guys who invented UKF thought that EKF stinks because it was a very poor idea to linearize a non linear function around a single point i.e. mean. So they thought that if they sample the data and pick some points then it would lead to better results. However, their professor was a big fan of EKF and he did not approved the idea of UKF. So that guys published the research paper and called it Unscented on purpose so that they can tell the world that EKF stinks!!

Larry: Great! Thanks for your time!

. . .

Well, that's all folks. I have tried to explain Larry the Kalman Filter Family in a very simple manner. Hope you gained something after reading the Posts. I will try to provide the code for Kalman Filter Family in an upcoming posts.

Kalman Filter

Extended Kalman Filter

In case you find any error you can contact me @LinkedIn here.

Machine Learning Self Driving Cars Artificial Intelligence Kalman Filter Udacity

About Help Legal