Kalman Filters

Consider what we have learned thus far: Sensors are noisy, imperfect, and often measure state variables indirectly. Similarly, actuators and moving components in all automated systems are noisy and imperfect, and at best can accomplish requested actions to within a small margin or error. Finally, the environment whithin which a robot is operating is complex and often poorly modelled (if at all). All the sources of uncertainty compound each other and the result is that, for particular applications which require cotinued ***tracking of state variables***, the accumulated error may render our estimates useless.

For instance, consider the problem of keeping track of the position of a car that is moving along a straight line (a very simple situation). If all we have access to is the sensor that measures the car's velocity (which is usually the case), and we use this sensor to estimate the car's position, we will run into the same problem inertial navigation systems: ***drift*** will make our positione stimate less and less accurate over time, eventually becoming useless for practical purposes.

What we need is an ***estimator***. A process that will combine information provided by our sensors, a model of how our system evolves over time as a result of actions, information about the amount of noise in our sensor measurements as well as in the results of actions performed, and puts everything together into an estimate of our state variables that is significantly better than could be obtained directly even if we carefully denoise sensor input.

One such ***estimator*** that is extensively used in practice is the ***Kalman Filter***.

The Kalman filter relies on the following idea:

- Use a model of the system to ***predict*** what the state variables will be at the next time instant
- Acquire information about the state variables by using the ***sensors***
- Combine both of these sources of information ***accounting for noise in sensing and actions*** so as to obtain a better estimate than we could get from either source alone.

We have encountered the idea of prediction before – it has been used in particle filters to simulate what different particles will do and thus create a cloud of particles that progressively approximate the robot's true state. It can be used as a sanity check on sensor inputs – e.g. by checking if the sensor measurements disagree in a significant way from the prediction. And now we will see that it can be used to obtain a more accurate estimate of state variables in the presence of noise and uncertainty.

Prediction requires a ***model*** – another idea we have encoutered before in control systems, localization, and management of sensor noise. Similarly to control systems, the Kalman Filter will use ***a linear model*** to represent the robot. The linearity requirement is important because it makes derivation of the Kalman Filter's updates manageable. Most of the systems we will have to deal with in practice are not actually linear, but even so the Kalmaf Filter works well in a surprisingly wide range of situations. We will come back to this linearity requirement later.

The Kalman Filter

The material in this section has been adapted from an excellent write-up by Ramsey Faragher (***https://ieeexplore.ieee.org/document/6279585***).

The Kalman Filter (KF) is based on the following assumptions about the system:

- The system is represented by a ***linear model,*** the model includes a ***noise term***
- Sensors provide a noisy estimate of state variables, we have a ***linear model*** relating state variables to sensor observations, and the model includes a ***noise term***
- The noise in both the system model and the sensor model is ***zero-mean Gaussian***

The system and sensor models used in the KF will look familiar from what we already studied in control systems. The system model is given by

$$\vec{x_t} = A_t\vec{x}_{t-1} + B_t\vec{u_t} + \vec{w_t}, \qquad \vec{w_t} \sim \mathcal{N}(\vec{\mu}_{w_t}, Q_t)$$

Here the vector $\vec{x_t}$ contains the state variables, vector $\vec{u_t}$ represents any inputs to the system, and vector $\vec{w_t}$ represents zero mean Gaussian with covariance ***Q***. Matrix ***A*** is the state transition matrix, relating the current and previous values of the state variables; matrix ***B*** relates state variables to control inputs applied to the system.

Along with our model for the system, we will need a ***sensor model*** that describes how the values of state variables are related to the sensor readings obtained by our system. The sensor model is linear and given by

$$\vec{z_t} = H_t\vec{x_t} + \vec{v_t}, \qquad \vec{v_t} \sim \mathcal{N}(\vec{\mu}_{v_t}, R_t)$$

Here the vector $\vec{v_t}$ represents zero-mean Gaussian noise with covariance ***R***. Keep in mind that unlike our models used while studying control systems, for the KF we explicitly allow the linear model to change over time, hence the dependence on ***t*** for the matrices in the above equations.

Finally, the KF keeps track of the ***process covariance P*** – this matrix describes the uncertainty in the state variable estimates, and changes over time. Intuitively, if we do nothing other than apply our linear model to obtain estimates for our state variables, uncertainty should grow over time. Each update is affected by random noise, and the noise at time ***t=c*** will be factored into every update for ***t>c***. Noise compounds over time, and thus uncertainty grows.

The KF consists of two steps

Prediction

Obtain an estimate for the state variables using the linear model, and obtain an estimate for the process covariance also from the linear model. ***Notation:*** In what follows, we use

$\vec{x_t}$ - the actual value of the state variables

$\check{x}_t$ - the ***predicted value*** of the state variables from the KF's first step
$\hat{x}_t$ - the ***final estimate*** of the state variablesafter completing both steps of the KF

The equations that yield the predictions for the first step of the KF are

$$\check{x}_t = A_t\hat{x}_{t-1} + B_t\vec{u}_t \qquad (1)$$

$$\check{P}_t = A_t\hat{P}_{t-1}A_t^T + Q_t \qquad (2)$$

Notice that the predicted state variables at time ***t*** use the final Kalman estimate of the state variables at time ***t-1***. Similarly, the prediction for the ***process covariance*** at time ***t*** uses the Kalman estimate of the process covariance at time ***t-1***[1].

Equation (1) simply applies the linear system to our previous Kalman estimate for the state variables, including any control inputs applied at time ***t***. Notice there is no noise term, we're not trying to simuluate the effect of the Gaussian noise on the state variables here.

Measurement Update

Given the prediction from step 1, the KF uses sensor measurements to update the estimates for our state variables, the update is given by

$$\hat{x}_t = \check{x}_t + K_t(\vec{z}_t - H_t\check{x}_t) \qquad (3)$$

$$\hat{P}_t = \check{P}_t - K_tH_t\check{P}_t \qquad (4)$$

Let's take a moment to understand what is going on here. In equation (3), the second term is comparing the difference between the sensor measurements we obtained at time ***t***, and the ***predicted value of the sensor measurements*** that we obtain from using our estimate from equation (1) and the sensor model (without the noise term). This difference is then scaled using the ***Kalman gain*** given by matrix ***K***.

The Kalman gain is updated at every step, and balances the uncertainty in the process with the uncertainty in the sensor readings giving more weight to whichever of these has smaller uncertainty. In equation (3) if the difference between our predicted state variables and the corresponding sensor measurements is small, the adjustment will be small. Conversely, if the difference is large, then a more significant adjustment will be carried out.

The Kalman gain is given by

1 The derivation of the update for the process covariance (2) is not so straightforward, for reference, please see equations 2.5 through 2.13 in https://www.control.utoronto.ca/people/profs/kwong/ece1639/2010/notes/chap2.pdf

$$K_t = \check{P}_t H_t^T (H_t \check{P}_t H_t^T + R_t)^{-1}$$

This is derived by optimizing over the ***error covariance*** of the Kalman estimate[2] so as to obtain the estimate that is closest to the true value of the state variables.

Finally, equation (4) reflects the expectation that the measurement update will ***reduce the uncertainty*** in our process.

Summary

The Kalman Filter is a powerful estimator for state variables in dynamical systems. It relies on a linear model to predict the values of the state variables, and performs an adjustment based on sensor measurements in an amount that is determined by taking into account the uncertainties in the process and the measurements.

The KF is implemented as a loop – given the initial estimates for the state variables, process covariance (this can be a matrix with all zeroes), and the covariances for noise in the process and measurement, the KF carries out the prediction and measurement update for each succeeding step.

Limitations

The KF relies on the assumptions of zero-mean Gaussian noise, and a linear system. In practice the latter assumption is often not accurate. Despite this, the KF remains useful over a wide range of problems, including some non-linear ones.

If bad performance is encountered due to nonlinearities, the ***extended Kalman filter (EKF)*** may be of use. The EKF ***linearizes the system around the current mean and covariance***, and then performs estimation. Implementing an EKF can be non-trivial. As an alternative, particle filters (already discussed for linearization) can be used to estimate the state of highly non-linear systems.

2 See Welch & Bishop “An Introduction to the Kalman Filter”, SIGGRAPH 2001 course notes, section 4.1.2