Kalman Filters for Vehicle Localization

Noah Mollerstuen

Introduction

In the field of mobile robotics, localization is the problem of determining the position and orientation of a robot relative to its environment. Often, the structure of the robot's environment is unknown, so the robot cannot rely on external cues. Instead, a robot can determine its position based on internal sensors, such as GPS, wheel encoders, gyroscopes, and accelerometers. Unfortunately, these sensors are often noisy: GPS signals are accurate only to within a few meters, wheels can slip, and inertial sensors drift over time. Furthermore, when the robot is in motion, the position to be measured is always changing, making attempts to correct for errors difficult. How then can we make accurate predictions of the robot's position based on uncertain sensor measurements that are constantly changing? The common solution is a Kalman filter, a probabilistic model which leverages our understanding of the system dynamics (the physical laws of motion) to optimally update our beliefs of the robot's position over time. In this exploration, I implement a simple Kalman filter and attempt to understand its underlying principles.

Dynamical Systems

In order to fully leverage Kalman filters, it is necessary to first create a model of the way our robot moves. This would be different for driving robots, robotic aircraft, underwater robots, etc. Modeling vehicle dynamics precisely is an entire field of engineering, but thankfully even simple models can be used in Kalman filters. For my investigation, I focused on a simple but non-trivial example: a plane flying at a constant velocity in one dimension. In this system, there are only two variables we need to track: the plane's position (x) and its velocity (v). Kalman filters operate on discrete time steps, so we write our equations in discrete form: the plane's position and velocity at each time step depend on the previous position and velocity, and on the size of the time step Δt.

$$x(t + \Delta t) = x(t) + v(t) \cdot \Delta t \tag{1}$$

$$v(t + \Delta t) = v(t) \tag{2}$$

Equations (1) and (2) are called the state-space equations of the system. Given some initial conditions at time $t = 0$, they are sufficient to predict the state of the system at every time step $t > 0$. For compactness, and to aid in later generalizing them, the equations can also be written in matrix form:

$$\begin{bmatrix} x(t + \Delta t) \\ v(t + \Delta t) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ v(t) \end{bmatrix} \tag{3}$$

Gaussian Representation

Because the Kalman filter cannot know the exact state of the system (the true position and velocity of the plane), it must instead operate on the probability distribution of the state variables. The Kalman filter makes the assumption that this probability distribution is a multivariate gaussian. Each variable has a mean, which is the filter's current "best guess" for that variable. The model's uncertainty about each variable is captured by the variance of the gaussian in that direction. It is also possible for our state variables to be correlated. In our example, if the plane is moving faster than we predict, it has probably also moved farther than we thought. These relationships are important for the model to capture, as they can allow us to infer the values of some state variables without directly measuring them. To capture this correlation between state variables, we use a covariance matrix. The mean vector μ and correlation matrix P fully describe the probability distribution of the state variables.

$$\mu = \begin{bmatrix} \mu_x \\ \mu_v \end{bmatrix} \tag{4}$$

$$P = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xv} \\ \Sigma_{vx} & \Sigma_{vv} \end{bmatrix} \tag{5}$$

Now that we can describe the distribution of possible states at a given time, we can use our state space equations from earlier to predict how the system will evolve over time. Instead of updating a single point, we transform the entire distribution simultaneously by updating the means and covariance matrix. If u_k is the mean vector at time k, the mean at the following time step can be calculated as

$$\mu_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mu_k = F_k \mu_k \tag{6}$$

To calculate the new covariance matrix, we rely on the identity $Cov(Ax) = A\,Cov(x)A^T$, where x is a probability distribution and A is any linear mapping. Applying this identity to equation (6), we find

$$P_{k+1} = F_k P_k F_k^T \tag{7}$$

Sensor Measurements

Now we can predict how the probability distribution of the states will change over time, but if we are unsure of the exact initial conditions, we can never reduce our uncertainty. To do that, we need to incorporate sensor measurements. In our example with a plane, imagine we have a radar that can measure the distance to the plane every five seconds. Unfortunately, this measurement is noisy: its reading is somewhere around the correct distance, but it could be off in either direction. The radar also can't directly tell us anything about the plane's velocity. In order to incorporate the sensor into the Kalman filter, we must model it as a gaussian and relate it to the state space equations. The expected sensor measurement z_k can be related to the current estimates by the sensor matrix Hk:

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mu_k = H_k \mu_k \tag{8}$$

A sensor matrix of $\begin{bmatrix} 1 & 0 \end{bmatrix}$ represents a sensor that directly measures the first state variable (position) and is unaffected by the second (velocity), which is an accurate model of our radar. H_k can also have multiple rows to represent many different sensors, each of which measures a different combination of the state variables. To represent the uncertainty in the measurements, we use another covariance matrix, R_k. With only one sensor, this is a single number representing the variance of the sensor measurement (sensors are assumed to sample from a gaussian centered around the "correct" value), but with n sensors, R_k would be an $n \times n$ matrix, with covariance possible between sensors. Using these two matrices, we can model the most likely states of the system based on our sensor measurement, which is just a gaussian with mean z_k and covariance R_k.

Combining Predictions

Now, whenever we receive a new sensor measurement, we have two ways to predict the next state of the system: by predicting how our previous beliefs have evolved over time, and by considering which states are likely to lead to this sensor measurement. Our new best guess of the system state should be where both predictions overlap: these states are both likely to have evolved from the previous state of the system and likely to result in the current sensor measurements. Because both predictions are in the form of a gaussian, we can find this overlap simply by multiplying the gaussians together, which happens to result in another gaussian. This then becomes our new estimate of the state. Each time a new sensor measurement is received, the Kalman filter first predicts how the system has evolved based on the time that has passed since the last measurement (using equation 6 and 7), then multiplies that prediction with the estimation of the state based on the sensor readings. This process is repeated for every new measurement.

Evaluation

To test my understanding of Kalman filters, I implemented a simple simulation of the radar and plane system we have been discussing. The plane moves at a constant velocity and the radar produces an estimate of its position every five seconds. Figure 1 show the results of applying a Kalman filter to the noisy radar measurements. Note that the position estimate is much closer to the position than the radar measurements themselves, and the model can accurately predict the plane's velocity despite having no direct way to measure it.

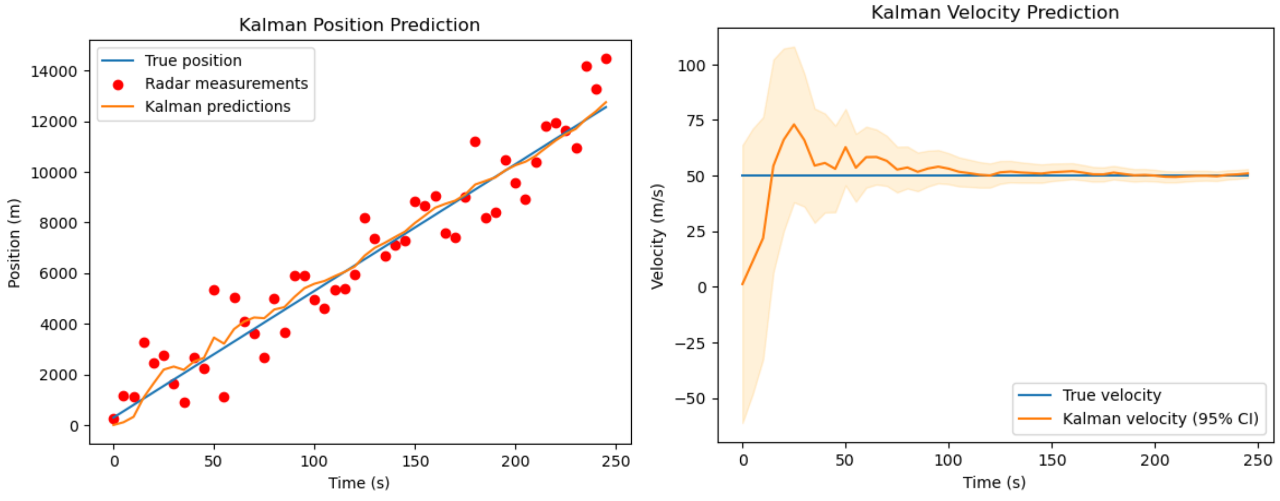

Figure 1: The true and predicted state of the plane over time. Red dots represent the noisy radar measurements. The orange shaded area is the 95% confidence interval (two sigma).

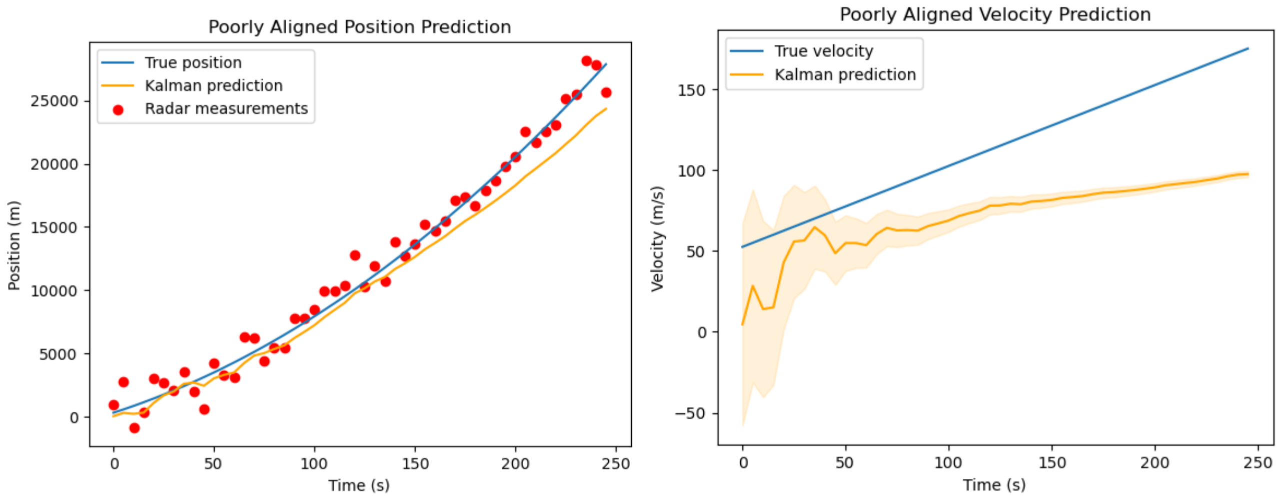

Figure 2: The state and Kalman predictions when the Kalman filter's model doesn't reflect the true dynamics of the system

Model Mismatch

Now let's look at a scenario in which our model of the world is imperfect. In the radar tracking example, we assumed the plane was traveling at a constant velocity. What would happen if the plane were actually accelerating? We can simulate this easily by updating the dynamics equations used to simulate the state (but not the Kalman filter's prediction matrix). Figure 2 shows the results. The position and velocity diverge from the true estimates, the predicted variance is still decreasing, as can be seen on the velocity plot. When the predicted variance is low, new measurements are not weighted as heavily, so the model fails to correct itself despite receiving new measurements that should reveal the model's current prediction is wrong.

One solution is to update the Kalman filter's model by adding an acceleration state variable and adding its prediction rules to the prediction matrix. That would likely work in this example, but there are some cases when it's nearly impossible to fully model every aspect of the system, especially when working with real vehicles. Wind might randomly accelerate the plane in ways that are difficult to predict. If we can't model the system, we can instead incorporate our uncertainty as to the true dynamics of the system into the Kalman filter by simply adding a constant matrix Q_k to the filter's predicted covariance at each update step, so Equation (7) becomes

$$P_{k+1} = F_k P_k F_k^T + Q_k \tag{9}$$

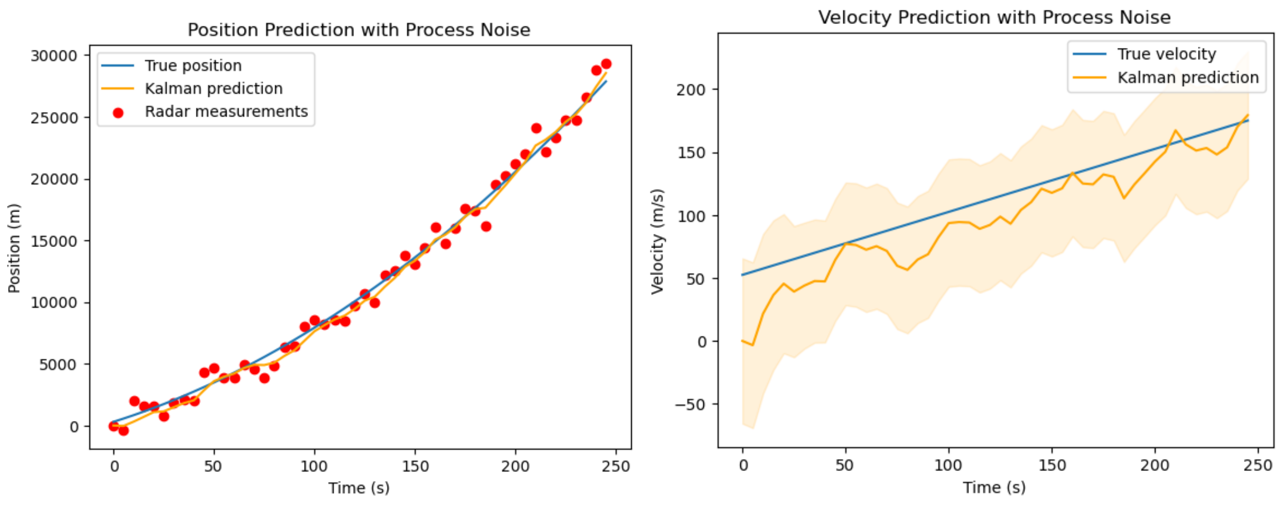

Figure 3: The true state and predictions of a kalman filter incorporating the process noise matrix Q_k

Typically the covariance matrix Q_k is diagonal, meaning that all of the covariance terms between two different variables are zero, and the variance of each variable is increased independently. Adding variance at each step prevents the uncertainty from collapsing and allows the model to incorporate new measurements that conflict with its prior beliefs. Figure (3) shows the results of applying a filter incorporating a large matrix Q_k to the previous system, without updating its prediction matrix. The predicted prediction tracks the plane's true position much more accurately, even in the presence of unexpected acceleration. The confidence interval on the velocity plot shows that the model remains uncertain, even after incorporating many sensor measurements. The cost of this uncertainty can be seen by comparing Figure (1) to Figure (3): the increased uncertainty in velocity makes the model more susceptible to noise in each individual sensor measurement.

The entries of the process noise matrix Q_k are parameters of the Kalman filter that can be tuned for optimal performance on different systems. Larger entries in the matrix mean the model is more responsive to each new measurement, which allows it to track unexpected behaviors, but also makes the predictions noisier.

Takeaways

The Kalman filter is able to accurately predict the plane's position from a series of independent noisy measurements, despite the underlying position changing continuously over time. More impressively, the model is able to predict the plane's velocity, despite no sensor measuring velocity directly. These capabilities, however, are dependent on an accurate dynamic model of the vehicle. In the event that the model does not fully describe the system, it is possible to capture these unexpected differences at the cost of increased sensitivity to noise. In the future, I hope to apply Kalman filters to more complex systems, such as robots moving in multiple dimensions and equipped with multiple different sensors.