1 Kalman Filter Introduction

You should first read Chapter 1 of *Stochastic models, estimation, and control: Volume 1* by Peter S. Maybeck (available here).

1.1 Explanation of Equations (1-3) and (1-4)

Equation (1-3) is the weighted average of z_1 and z_2, and Equation (1-4) is the variance of the weighted average. You can see how they are derived by reading Minas Spetsakis' notes (available here).

1.2 Explanation of Equation (1-12)

I am reasonably certain that Equation (1-12) contains a typographical error. Equation (1-12) is the variance of $\hat{x}(t_3^-)$ in Equation (1-11), repeated here for convenience:

$$\hat{x}(t_3^-) = \hat{x}(t_2) + u\Delta t$$

$\hat{x}(t_3^-)$ is the sum of two Gaussian random variables:

$$\begin{aligned} \hat{x}(t_2) &\sim \mathcal{N}(0, \sigma_x^2(t_2)) \\ u\Delta t &\sim \mathcal{N}(0, (\Delta t)^2\sigma_w^2) \end{aligned}$$

[Note that that the velocity $u \sim \mathcal{N}(0, \sigma_w^2)$; when it is multiplied by the constant time interval Δt the variance of $u\Delta t$ is $(\Delta t)^2\sigma_w^2$ which can be easily proved by using the definition of variance.] It can be shown that if X_1 and X_2 are independent Gaussian random variables such that

$$\begin{aligned} X_1 &\sim \mathcal{N}(\mu_1, \sigma_1^2) \\ X_2 &\sim \mathcal{N}(\mu_2, \sigma_2^2) \end{aligned}$$

then the sum $X_1 + X_2$ is also a Gaussian random variable such that

$$X_1 + X_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

Using this fact, we conclude that the variance of $\hat{x}(t_3^-)$ must be

$$\sigma_x^2(t_3^-) = \sigma_x^2(t_2) + (\Delta t)^2\sigma_w^2$$

2 Kalman Filter

The Kalman filter is an optimal data processing algorithm for linear Gaussian state-space models.

2.1 State

State variables describe the robot and its environment. Typical state variables include:

- The robot pose (its location and orientation in the world). A rigid robot moving in a planar environment has three pose variables: two variables describing its position in the plane and one variable describing its heading direction. A rigid robot moving in three dimensions would have at least six pose variables: three describing its position and at least three describing its orientation.
- The velocity and acceleration vectors of the robot (both linear and angular).
- Locations of objects in the environment (a map). If these objects are moving, then the velocities of these objects are potentially state variables.

The state variables are typically collected into a single n-dimensional vector x_k (the state at time k). We are mostly interested in the pose of a robot moving in a plane so our state vector will typically be

$$x_k = \begin{bmatrix} p_{x,k} \\ p_{y,k} \\ \theta_k \end{bmatrix}$$

where $p_{x,k}$ and $p_{y,k}$ are the position of the robot in the world at time k and θ_k is the heading direction (angle relative to the world x axis) at time k.

2.2 Observations or Measurements

A mobile robot typically has one or more sensors that produce measurements related to its internal state or environment. Examples of sensors measuring internal state include a GPS sensor that measure the position of the robot, and a compass that measures the orientation of the robot relative to the earth's magnetic field. Examples of sensors that measure the environment include range sensors such as sonar and lidar, and cameras that capture video images.

In the lost-at-sea example from Maybeck, the measured variable was the same as the state (the one-dimensional location in the world). Usually, sensor measurements do not correspond directly to the state variables but instead are some function of the state variables.

The measurements are typically collected into a single m-dimensional vector z_k (the measurements at time k).

2.3 Control Data

Control data describe the change of state of the robot or its environment. For a free falling object, gravity is the force causing the change in the position and velocity of the object. In mobile robotics, the velocity of the robot causes the pose of the robot to change.

The control data are typically collected into a single l-vector u_k that describes the change of state from time k to time $k+1$.

2.4 State-Space Model

The Kalman filter recursively produces optimal estimates of state given a description of how the state changes from one time step to the next and a description of how the measurements are related to the state variables. These descriptions come in the form of a pair of equations called the *process* or *plant model* and the *measurement model*.

2.4.1 Process Model

The process model describes how the state changes from one time to the next as a function of the current state, the current control inputs, and a noise model. For a Kalman filter, the process model is

$$x_{k+1} = A_k x_k + B_k u_k + w_k$$

A_k is an $n \times n$ matrix (sometimes called the state transition model) and B_k is an $n \times l$ matrix (sometimes called the control-input model); often A_k and B_k are constant in time in which case they are denoted simply A and B respectively. Because the state and control vectors are multiplied by matrices and the resulting terms summed, the process model is linear in its arguments (the Kalman filter assumes a linear model of the change in state).

$w_k \sim \mathcal{N}(0, Q_k)$ is a Gaussian random variable (sometimes called the process noise) that models the uncertainty in the state transition. Q_k is an $n \times n$ covariance matrix. Notice that the Kalman filter assumes an additive Gaussian noise model for the process noise.

2.4.2 Measurement Model

The measurement model describes how the measured quantities are related to the state. For a Kalman filter, the measurement model is

$$z_k = H_k x_k + v_k$$

H_k is an $m \times n$ matrix that describes the linear relationship between the state and the observations.

$v_k \sim \mathcal{N}(0, R_k)$ is a Gaussian random variable (called the neasurement noise) that models the uncertainty in the measured quantities. R_k is an $m \times m$ covariance matrix. Notice that the Kalman filter assumes an additive Gaussian noise model for the measurement noise.

Example 1

Consider an object in free fall (ignoring atmospheric drag) such as a vertically thrown ball. Suppose that we are interested in the height and velocity of the object. From Newtonian physics, the height of the object at time t is

$$y(t) = y(t_0) + \dot{y}(t_0)\Delta t - \tfrac{1}{2}g(\Delta t)^2$$

where $y(t_0)$ is the initial height of the object, $\dot{y}(t_0)$ is the initial velocity of the object, $\Delta t = t - t_0$ is the elapsed time, and g is the acceleration due to gravity. The velocity of the object at time t is

$$\dot{y}(t) = \dot{y}(t_0) - g\Delta t$$

If we assume regular discrete time intervals $\Delta t = 1$ we can write the height and velocity as

$$\begin{aligned} y_{k+1} &= y_k + \dot{y}_k - \tfrac{1}{2}g \\ \dot{y}_{k+1} &= \dot{y}_k - g \end{aligned}$$

Our state vector is

$$x_k = \begin{bmatrix} y_k \\ \dot{y}_k \end{bmatrix}$$

Our control input is

$$u_k = g$$

Our process model is

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ &= A\begin{bmatrix} y_k \\ \dot{y}_k \end{bmatrix} + Bg + w_k \\ &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} y_k \\ \dot{y}_k \end{bmatrix} + \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix} g + w_k \end{aligned}$$

2.4.3 Example 2

An omnidirectional robot is a mobile robot capable of moving in any direction in the plane. Let the state be the position of the robot

$$x_k = \begin{bmatrix} p_{x,k} \\ p_{y,k} \end{bmatrix}$$

Assume that the robot is capable of controlling its velocity in both the x and y directions so that in one time step it moves by the vector

$$u_k = \begin{bmatrix} d_{x,k} \\ d_{y,k} \end{bmatrix}$$

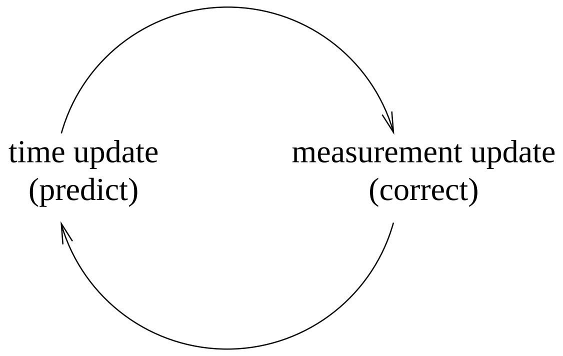
Our process model is

$$\begin{aligned}
x_{k+1} &= Ax_k + Bu_k + w_k \\
&= A \begin{bmatrix} p_{x,k} \\ p_{y,k} \end{bmatrix} + B \begin{bmatrix} d_{x,k} \\ d_{y,k} \end{bmatrix} + w_k \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{x,k} \\ p_{y,k} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_{x,k} \\ d_{y,k} \end{bmatrix} + w_k
\end{aligned}$$

2.5 Algorithm

Given a state-space model, the Kalman filter produces state estimates in two stages. In stage one, the current state estimate is projected forward in time (using the process model). This is the time-update stage and it is performed using only the model of how the state evolves (and not the actual measurement that was performed). Because the measurement is not used, this stage can also be thought of as being a prediction of the state.

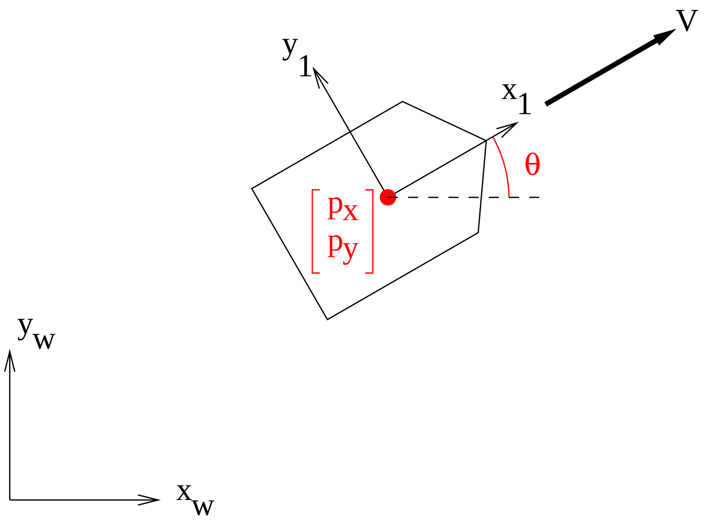
In stage two, the measurement is incorporated into the projected state to obtain an improved estimate. This is the measurement-update stage. Because the measurement is used to improve the estimate obtained by projecting the state forward in time, this stage can also be thought of as being a correction of the predicted state.



State-space model	
$x_k = Ax_{k-1} + Bu_{k-1} + w_k$	
$z_k = H_k x_k + v_k$	
Initialization	
$\hat{x}_0 = E[x_0]$	estimated state at time $k = 0$
$P_0 = E[(x_0 - E[x_0])(x_0 - E[x_0])^T$	estimated covariance of $\hat{x}_0$
Prediction-correction for time $k = 1, 2, 3, \ldots$	
Prediction	
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$	predict state using process model
$P_k^- = AP_{k-1}A^T + Q_k$	state uncertainty estimate increases due to prediction
Correction	
$\hat{z}_k = H\hat{x}_k^-$	expected measurements from measurement model
$r_k = z_k - \hat{z}_k$	difference between actual and expected measurement
$S_k = HP_k^- H^T + R_k$	measurement covariance
$K_k = P_k^{-1} H^T S_k^{-1}$	the weight to give to the measurement
$\hat{x}_k = \hat{x}_k^- + K_k r_k$	new state estimate incorporating the measurement
$P_k = P_k^- - K_k H P_k^-$	state uncertainty estimate decreases due to correction

3 Extended Kalman Filter

One problem with using a Kalman filter in mobile robotics applications is that the process model and observation models are frequently non-linear. Consider the mobile robot shown below at time k moving with linear velocity V_k and angular velocity ω_k.



In the world frame, the robot's pose is

$$x_k = \begin{bmatrix} p_{x,k} \\ p_{y,k} \\ \theta_k \end{bmatrix}$$

and it's velocity is

$$u_k = \begin{bmatrix} V_k \cos\theta_k \\ V_k \sin\theta_k \\ \omega_k \end{bmatrix}$$

If we are only interested in the robot's pose, we might consider using the process model

$$\begin{aligned} x_{k+1} &= \text{(current pose)} + \text{(current velocity)} * \text{(change in time)} + \text{noise} \\ &= x_k + u_k \Delta T + w_k \\ &= \begin{bmatrix} p_{x,k} + V_k \Delta T \cos\theta_k \\ p_{y,k} + V_k \Delta T \sin\theta_k \\ \theta_k + \Delta T \omega_k \end{bmatrix} + w_k \\ &= f(x_k, u_k) + w_k \end{aligned}$$

where the function f is non-linear in the state variable θ_k (the cosine and sine terms).

Recall that in the time-update stage of the Kalman filter we need to project the state forward in time to obtain the predicted state $\hat{x}^-_{k+1}$. In this case, we can just use f to predict the state

$$\hat{x}^-_{k+1} = f(\hat{x}_k, u_k) \qquad \text{instead of} \qquad \hat{x}^-_{k+1} = A\hat{x}_k + Bu_k$$

Recall that we also need to predict the state covariance; for a linear process model the Kalman filter uses

$$P^-_{k+1} = AP_kA^T + Q_k$$

For our non-linear process model, we do not have the matrix A to predict the state covariance.

One solution to this problem is to linearize the function f; that is, replace f with a linear approximation of f. This can be accomplished using the following fact: the best linear approximation to a differentiable function at a given point is given by the Jacobian.

Let F be the Jacobian (with respect to the state x) of f:

$$F = \begin{bmatrix} \frac{\partial f_1}{\partial p_x} & \frac{\partial f_1}{\partial p_y} & \frac{\partial f_1}{\partial \theta} \\ \frac{\partial f_2}{\partial p_x} & \frac{\partial f_2}{\partial p_y} & \frac{\partial f_2}{\partial \theta} \\ \frac{\partial f_3}{\partial p_x} & \frac{\partial f_3}{\partial p_y} & \frac{\partial f_3}{\partial \theta} \end{bmatrix}$$

where

$$\begin{aligned} f_1 &= p_{x,k} + V_k \Delta T \cos\theta_k \\ f_2 &= p_{y,k} + V_k \Delta T \sin\theta_k \\ f_3 &= \theta_k + \Delta T \omega_k \end{aligned}$$

The Jacobian matrix F is of size $n \times n$ where n is the dimension of the state vector. Computing the partial derivatives yields

$$F = \begin{bmatrix} 1 & 0 & -V_k \Delta T \sin\theta_k \\ 0 & 1 & V_k \Delta T \cos\theta_k \\ 0 & 0 & 1 \end{bmatrix}$$

The predicted state covariance can now be computed as

$$P^-_{k+1} = FP_kF^T + Q_k$$

which allows us to complete the time-update step.

Note that the measurement model is also often non-linear in the state variables for mobile robotics applications. Instead of the linear measurement model

$$z_k = Hx_k + v_k$$

we might have some non-linear equation h such that

$$z_k = h(x_k) + v_k$$

The lack of the matrix H makes it impossible to compute the equations required in the measurement-update step of the Kalman filter. We can use the same linearization technique with h, computing its Jacobian (with respect to the state variables), J

$$J = \begin{bmatrix} \frac{\partial h_1}{\partial p_x} & \frac{\partial h_1}{\partial p_y} & \frac{\partial h_1}{\partial \theta} \\ \vdots & \vdots & \vdots \\ \frac{\partial h_m}{\partial p_x} & \frac{\partial h_m}{\partial p_y} & \frac{\partial h_m}{\partial \theta} \end{bmatrix}$$

The Jacobian matrix J is of size $m \times n$ where m is the dimensionality of the observation vector z_k. The measurement-update equations can then be computed using J in place of H.

When the non-linear process and measurement models are approximated using linearization as described above the resulting filter is called an *extended Kalman filter* (EKF). The EKF is probably the most popular tool for state estimation in robotics.

Because the process and measurement models are approximated using linearized equations, the EKF is not an optimal estimator. If the functions f and h being approximated are strongly non-linear the EKF can perform very poorly. Another problem with the linearization is that the partial derivatives in the Jacobian matrices may be expensive (or impossible) to compute.