Sensor fusion of an Inertial Motion Unit sensor and a Faulhaber decoder using the Kalman filter for improved dead reckoning

Submitted as part of a semester project titled

Improved dead reckoning using an IMU sensor and the Kalman filter

under the guidance of

Prof. Dr.- Ing Peter Fromm
Department of Electrical Engineering and Information Technology
Darmstadt University of Applied Sciences, Darmstadt

Report submitted by

Prem Jason Mendonca
Matriculation number: 1112318
Department of Electrical Engineering and Information Technology
Darmstadt University of Applied Sciences, Darmstadt

December/16/2022

Table of Contents

1 Abstract

A modern car is an electronic and electrical system formed by the interconnection of ECUs, sensors, and actuators. While the technology is moving towards autonomous driving, the sensors that measure position, speed, and orientation, among others, become crucial for guided navigation. However, these sensors are not perfect, as they are prone to environmental noise and disturbances such as vibrations, wind turbulence, etc.

A Kalman filter is an estimation algorithm used to obtain an optimal estimated sensor reading from a noisy environment. Inertial motion sensors, rate gyroscopes, accelerometers, and Global Positioning System sensors all use the Kalman filter algorithm to estimate the optimal estimation from the noisy environment. It should be noted that the Kalman filter is not perfect; however, it has the closest response to the ideal reading and can only help in optimally reducing noise based on the modelled noise.

2 Introduction

2.1 State-space representation

A state-space representation is a mathematical model of a dynamic physical system as a set of input, output, and state variables related by difference equations. State variables are variables whose values evolve over time in a way that depends on the values they have at any given time and on the externally imposed values of input variables. Output variables' values depend on the values of the state variables.

If the dynamical system is linear, time-invariant, and finite-dimensional, then the differential and algebraic equations may be written in matrix form.

2.2 Additive white Gaussian noise

Additive white Gaussian noise (AWGN) is a basic noise model used to mimic the effect of many random processes that occur in nature. The modifiers denote specific characteristics:

- additive because it is added to any noise that might be intrinsic to the system.
- white refers to the idea that it has uniform power across the frequency band for the system.
- Gaussian because it has a normal distribution in the time domain with an average time domain value of zero.

2.3 Standard deviation, variance, and covariance

The standard deviation, σ, is a measure of how much a set of values varies or disperses. A low standard deviation indicates that the values tend to be close to the mean (also called the expected value) of the set, while a high standard deviation indicates that the values are spread out over a wider range.

Variance, Var(x) or σ^2 is a measure of dispersion, which is how far apart a set of numbers is from their average value.

Covariance, Cov(x,y), is a measure of the combined variability of two random variables, x and y. If the greater values of one variable mainly correspond with the greater values of the other variable, and the same holds for the lesser values (that is, the variables tend to show similar behavior), the covariance is positive. In the opposite case, when the greater values of one variable mainly correspond to the lesser values of the other, (that is, the variables tend to show opposite behavior), the covariance is negative. As a result, the sign of the covariance indicates the tendency in the linear relationship between the variables.

The covariance is sometimes called a measure of "linear dependence" between the two random variables. When the covariance is normalized, one obtains the Pearson correlation coefficient, which gives the goodness of fit for the best possible linear function describing the relation between the variables. In this sense, covariance is a linear gauge of dependence.

2.4 Covariance matrix

A covariance matrix is a square matrix that gives the covariance between each pair of elements in a given vector. Any covariance matrix is symmetric, and its main diagonal contains variances (i.e., the covariance of each element with itself).

The covariance matrix can be used to derive a transformation matrix. This derivation is called a “whitening transformation.” A whitening transformation transforms a vector of random variables with a known covariance matrix into a set of new variables whose covariance is the identity matrix, meaning that they are uncorrelated and each have variance 1. The transformation is called "whitening" because it changes the input vector into a white noise vector.

2.5 Sensor fusion

Sensor fusion is the process of combining sensor data or data derived from disparate sources such that the resulting information has less uncertainty than would be possible when these sources were used individually. One of the sensor fusion algorithms is the Kalman filter.

2.6 Kalman filter

Kalman filter is a linear quadratic estimator or regulator that can estimate observable and unobservable parameters with great accuracy in real-time. Estimates with high accuracy are used to make precise predictions and decisions. It can use inaccurate or noisy measurements to estimate the state of that variable or another unobservable variable with greater accuracy. The real power of the Kalman filter is not smoothing measurements. It is the ability to estimate system parameters that cannot be measured or observed with accuracy. Estimates with higher accuracy in real-time systems give systems more control and thus more capabilities.

3 Algorithm overview

3.1 Table of Definitions

Table 3.1 identifies the variables that are used in the algorithm. Each variable listed has a structure type and category. Further in this document, use this table as a reference.

x	state variable vector	n x 1 column vector	output
P	state covariance matrix	n x n matrix	output
z	measurement variable vector	m x 1 column vector	input
R	measurement covariance matrix	m x m matrix	input
K	Kalman gain	n x m matrix	internal
A	state transition matrix	n x n matrix	system model
H	state-to-measurement matrix	m x n matrix	system model
Q	process noise covariance matrix	n x n matrix	system model

Table 3.1: Definitions of Parameters of the Algorithm.

3.2 Process Diagram

Figure 3.1 shows the step-by-step process diagram of the Kalman filter algorithm.

Step 1 uses the measurement to initialize the system estimate. Each application of the Kalman filter may do this differently (have re-initialization stages), but the goal is to have a system state estimate that can be updated for future measurement with the Kalman filter equations.

Steps 2 through 4 demonstrate how measurements are filtered in and the state estimate is updated.

Use this figure as a reference throughout this document.

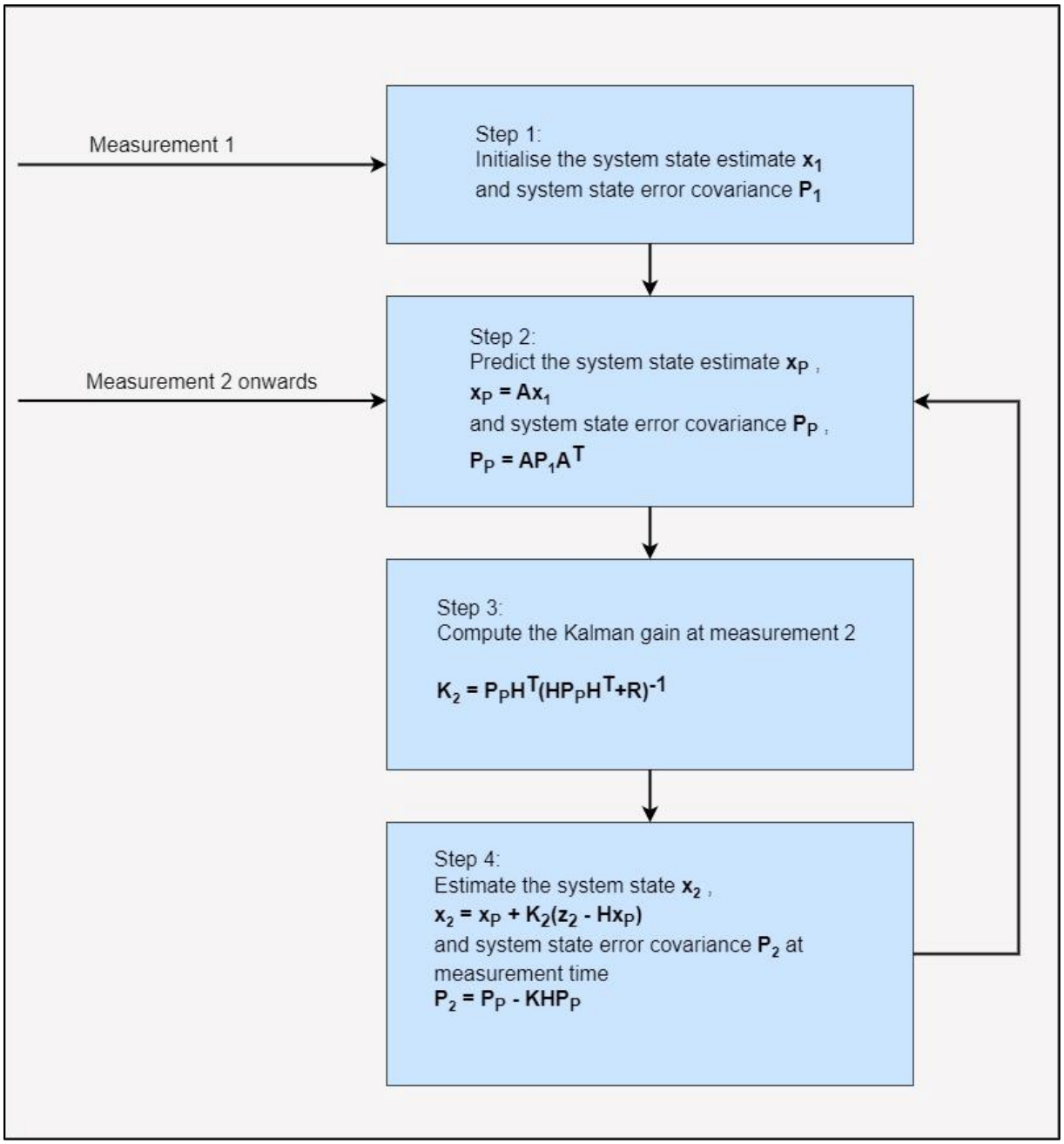

Figure 3.1: Process diagram of the Kalman filter algorithm

3.3 Covariance matrix P

An indication of uncertainty must be given for the initial state. This matrix changes throughout the filtering process, as well as during the predict and update steps. This matrix can be initialized on the basis of the sensor's accuracy. If the sensor is relatively inaccurate, large values should be used here to allow the filter to converge relatively quickly. If the sensor is very accurate, small values should be used here.

3.4 Dynamics matrix A

The dynamics matrix forms the core of the filter and therefore must be set up with great understanding of the physical context. A dynamic matrix is defined on the basis of the equations governing the predictions of system parameters in a vehicle motion model.

3.5 Process noise covariance matrix Q

The process noise covariance matrix is introduced since the movement of the vehicle (in the sense of a superimposed, normally distributed noise) may also be disturbed. This matrix tells us about the filter, and how the system state can "jump" from one step to the next.

Imagine the vehicle that drives itself. It can be disturbed by a gust of wind or road bumps, which have a force effect. A speed change by the driver is also an acceleration that acts on the vehicle. If an acceleration now has an effect on the system state, the physical dependence is Q. The matrix is a co-variance matrix that accounts for the errors caused in the process.

3.6 Measuring matrix H

This matrix is used to tell the filter what is measured, and how it relates to the state vector. In simpler terms, it transforms the predicted parameters into the same form as the input measurement.

If the sensors measure in a different unit or the size is affected by detours, then the measuring matrix is used to map the relationships in the formula.

3.7 Measurement noise covariance matrix R

An element of uncertainty must also be added to the measurement. This measurement uncertainty indicates the degree to which the measured can be trusted. If the sensor is very accurate, small values should be used here. If the sensor is relatively inaccurate, large values should be used here

4 Illustration of sensor fusion using the Kalman filter

To illustrate sensor fusion using the Kalman filter, let us take the case of determining the car's position using a velocity sensor that aids in improving the position determined by a GPS sensor.

Kalman filter to determine the position using a velocity sensor

The measurements of velocity from the velocity sensor will be represented as a 4-by-1 column vector **z**:

$$z_k = \begin{bmatrix} v_{x_{m_k}} \\ v_{y_{m_k}} \\ 0 \\ 0 \end{bmatrix}$$

The associated variance-covariance matrix for these measurements will be provided with the time tag for when the measurement occurred, **t**.

$$R_k = \begin{bmatrix} Cov^2(v_{x_m}) & Cov(v_{x_m}, v_{y_m}) \\ Cov(v_{x_m}, v_{y_m}) & Cov^2(v_{y_m}) \end{bmatrix}$$

$$t_k = t_{m_k}$$

The subscript **m** denotes the measurement parameters, and the subscript **k** denotes the order of the measurement.

The position estimate is represented by a 4-by-1 column matrix, **x**,

$$x_k = \begin{bmatrix} v_x \\ v_y \\ x \\ y \end{bmatrix}$$

where, x and y are calculated from measurements v_x and v_y, respectively.

$$x = x + v_x * \Delta T$$

$$y = y + v_y * \Delta T$$

Its associated variance-covariance matrix is represented by a 4-by-4 matrix, **P**.

$$P_k = \begin{bmatrix} Cov^2(v_x) & Cov(v_x, v_y) & Cov(v_x, x) & Cov(v_x, y) \\ Cov(v_x, v_y) & Cov^2(v_y) & Cov(v_y, x) & Cov(v_y, y) \\ Cov(v_x, x) & Cov(v_y, x) & Cov^2(x) & Cov(x, y) \\ Cov(v_x, y) & Cov(v_y, y) & Cov(x, y) & Cov^2(y) \end{bmatrix}$$

Additionally, the state estimate has a time tag denoted as **T**.

$$T_k$$

Step 1: Initialize System State

The Kalman filter initializes the system state with the first measurement.

When using the velocity sensor, the input measurements only contain velocity information. The output system state will contain the velocity and position of the car.

When the first measurement comes, the velocity of the object at that point in time is known. The system state estimate will be set to the input velocity measurement after the first estimate, and the position will be calculated from that velocity measurement. Position is estimated with a linear approximation; position (distance traveled) is equal to the velocity times the time it took to travel to that position.

The system-state error covariance will be the measurement's velocity accuracy and an approximated position accuracy. Note that this position accuracy approximation is something that can be tuned and adjusted after running data through the filter. There are different ways of approximating these values, so it does not matter if these approximations differ from one scenario to another.

The equations below show the input and output values for this Kalman filter after receiving the first measurement.

$$z_1 = \begin{bmatrix} v_{x_{m_1}} \\ v_{y_{m_1}} \\ 0 \\ 0 \end{bmatrix}$$

$$R_1 = \begin{bmatrix} Cov^2(v_{x_m}) & Cov(v_{x_m}, v_{y_m}) \\ Cov(v_{x_m}, v_{y_m}) & Cov^2(v_{y_m}) \end{bmatrix}$$

$$t_1 = t_{m_1} = T_1 = \Delta T$$

$$x_1 = \begin{bmatrix} v_x \\ v_y \\ x \\ y \end{bmatrix} = \begin{bmatrix} v_{x_{m_1}} \\ v_{y_{m_1}} \\ v_{x_{m_1}} * \Delta T \\ v_{y_{m_1}} * \Delta T \end{bmatrix}, \qquad P_1 = \begin{bmatrix} Cov^2(v_x) & Cov(v_x, v_y) & 0 & 0 \\ Cov(v_x, v_y) & Cov^2(v_y) & 0 & 0 \\ 0 & 0 & Cov^2(x) & 0 \\ 0 & 0 & 0 & Cov^2(y) \end{bmatrix}$$

Step 2: Prediction: Predict the System State Estimate

When the second measurement is received, the system state estimate is propagated forward to time align it with the measurement. This alignment is done so that the measurement and state estimate can be combined.

The system model is used to perform this prediction, $\mathbf{x_p}$. In this example, a constant velocity linear motion model is used to approximate the object's position change over a time interval. Note that a constant-velocity model assuming zero acceleration is the most important consideration for the system under illustration. As per the constant velocity linear motion model, the equation states that the velocity of an object is equal to its displacement times the time period, assuming zero acceleration.

A state transition matrix represents these equations. This matrix is used to propagate the state estimate and state error covariance matrix appropriately. The reason for the state error covariance matrix propagation is that, when a state estimate is propagated in time, the uncertainty about its state at this future time step is inherently uncertain, so the error covariance grows.

$$z_2 = \begin{bmatrix} v_{x_{m_2}} \\ v_{y_{m_2}} \\ 0 \\ 0 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} Cov^2(v_{x_m}) & Cov(v_{x_m}, v_{y_m}) \\ Cov(v_{x_m}, v_{y_m}) & Cov^2(v_{y_m}) \end{bmatrix}$$

$$t_2 = t_{m_2} = T_2, \qquad \Delta T = T_2 - T_1$$

$$A = \begin{bmatrix} 0 & 0 & \Delta T & 0 \\ 0 & 0 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad x_2 = \begin{bmatrix} v_x \\ v_y \\ x \\ y \end{bmatrix} = \begin{bmatrix} v_{x_{m_2}} \\ v_{y_{m_2}} \\ x + v_{x_{m_2}} * \Delta T \\ y + v_{y_{m_2}} * \Delta T \end{bmatrix}$$

$$x_P = Ax_1 = A \begin{bmatrix} v_x \\ v_y \\ x \\ y \end{bmatrix} = \begin{bmatrix} x * \Delta T \\ y * \Delta T \\ x \\ y \end{bmatrix}$$

$$P_P = AP_{k-1}A^T + Q$$

Q Matrix

The Q matrix represents process noise for the system model. The system model is an approximation. Throughout the life of a system state, that system model fluctuates in its accuracy. Therefore, the Q matrix is used to represent this uncertainty and adds to the existing noise in the state. For this example, the system's actual accelerations and decelerations contribute to this error.

The process noise for the system model under illustration is a 4-by-4 diagonal matrix.

H Matrix

The Kalman filter uses the state-to-measurement matrix, H, to convert the system state estimate from the state space to the measurement space. It transforms the predicted state vector and covariance matrix into the same form as the input measurement.

In this illustration, the H matrix is a simple matrix that is set up to reduce the state estimate and error covariance to velocity-only (measured) values rather than velocity (measured) and position (estimated).

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 3: Compute the Kalman gain

The Kalman filter computes a Kalman gain for each new measurement that determines how much the input measurement will influence the system state estimate. In other words, when a really noisy measurement comes in to update the system state, the Kalman gain will trust its current state estimate more than this new inaccurate information.

This concept is the root of the Kalman filter algorithm and why it works. It can recognize how to properly weight its current estimate and the new measurement information to form an optimal estimate.

The Kalman gain, **K** at every measurement is updated as below:

$$K = P_P H^T (H P_P H^T + R)^{-1}$$

Step 4: Update: Estimate the System State and System State Error Covariance Matrix

The Kalman filter uses the Kalman gain to estimate the system state and error covariance matrix for the time of the input measurement. After the Kalman gain is computed, it is used to weight the measurement appropriately in two computations.

In the below computations, the first computation is the new system state estimate, x_k.

$$x_k = x_P + K(z_k - H x_P)$$

In the above equation, the factor ($z_k - Hx_P$) is the correction factor that is applied by considering the state-to-measurement mapping factor.

The second computation is the system state error covariance, P_k.

$$P_k = P_P(1 - KH)$$

Thus, the velocity sensor itself is used to predict and update the optimal values for position determination.

Sensor fusion of a velocity sensor and a GPS sensor for position determination

The GPS sensor can determine the position alone. The GPS sensor's position measurements will be represented as a 4-by-1 column vector $\mathbf{z_{GPS}}$:

$$z_{k_{GPS}} = \begin{bmatrix} 0 \\ 0 \\ x_{m_k} \\ y_{m_k} \end{bmatrix}$$

The corresponding H matrix is given by:

$$H_{GPS} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The GPS works at a different measurement rate (slower) than the velocity sensor. Further, since the GPS sensor directly provides the position value, whenever its measurement is received, the Kalman

gain is calculated, and we apply it to the update step (Step 4) of the Kalman filter, as described below:

$$K = P_P H_{GPS}^T (H_{GPS} P_P H_{GPS}^T + R)^{-1}$$

$$x_k = x_P + K\left(z_{k_{GPS}} - H_{GPS} x_P\right)$$

$$P_k = P_P (1 - K H_{GPS})$$

Thus, the sensor fusion of a velocity sensor and a GPS sensor, measured at different measurement rates, is used to determine the optimal values for the position determination.

5 Sensor fusion using the Kalman filter for improvement of dead reckoning in a car using an IMU sensor and a Faulhaber decoder

5.1 Kalman filter for position determination using an IMU sensor

The measurements of acceleration and angle along the z-axis are obtained from an accelerometer and an angle sensor, respectively, and are represented as a 10-by-1 column vector $\mathbf{z_{IMU}}$:

$$z_{k_{IMU}} = \begin{bmatrix} a_{xm} \\ a_{ym} \\ angle_z \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The associated variance-covariance matrix for these measurements will be provided with the time tag for when the measurement occurred, **t**. For this sensor, it is configured to arrive every 5 ms, as shown in Figure 5.1.

$$R_k = \begin{bmatrix} Cov^2(a_{xm_m}) & Cov(a_{xm_m}, a_{ym_m}) & Cov(a_{xm_m}, angle_{z_m}) \\ Cov(a_{xm_m}, a_{ym_m}) & Cov^2(a_{ym_m}) & Cov(a_{ym_m}, angle_{z_m}) \\ Cov(a_{xm_m}, angle_{z_m}) & Cov(a_{ym_m}, angle_{z_m}) & Cov^2(angle_{z_m}) \end{bmatrix}$$

$$t_k = t_{m_k} = 5ms$$

The subscript **m** denotes the measurement parameters, and the subscript **k** denotes the order of the measurement.

The position estimate is represented by a 10-by-1 column matrix, $\mathbf{x_{IMU}}$,

$$x_{k_{IMU}} = \begin{bmatrix} a_{xm} \\ a_{ym} \\ angle_z \\ a_x \\ a_y \\ v_x \\ v_y \\ x \\ y \\ \varphi \end{bmatrix}$$

where,

φ is established through measurement $angle_z$

a_x and a_y are established through measurements a_{xm} and a_{ym}, respectively

v_x and v_y are established through a_x and a_y, respectively

x and y are derived from established v_x and a_x, and established v_y and a_y, respectively.

Its associated variance-covariance matrix is represented by a 10-by-10 matrix, **P**.

$$P_k = \begin{bmatrix} Cov^2(a_{xm}) & Cov(a_{xm},a_{ym}) & Cov(a_{xm},angle_z) & Cov(a_{xm},a_x) & Cov(a_{xm},a_y) & Cov(a_{xm},v_x) & Cov(a_{xm},v_y) & Cov(a_{xm},x) & Cov(a_{xm},y) & Cov(a_{xm},\varphi) \\ Cov(a_{xm},a_{ym}) & Cov^2(a_{ym}) & Cov(a_{ym},angle_z) & Cov(a_{ym},a_x) & Cov(a_{xm},a_y) & Cov(a_{ym},v_x) & Cov(a_{ym},v_y) & Cov(a_{ym},x) & Cov(a_{ym},y) & Cov(a_{ym},\varphi) \\ Cov(a_{xm},angle_z) & Cov(a_{ym},angle_z) & Cov^2(angle_z) & Cov(angle_z,a_x) & Cov(angle_z,a_y) & Cov(angle_z,v_x) & Cov(angle_z,v_y) & Cov(angle_z,x) & Cov(angle_z,y) & Cov(angle_x,\varphi) \\ Cov(a_{xm},a_x) & Cov(a_{ym},a_x) & Cov(angle_z,a_x) & Cov^2(a_x) & Cov(a_x,a_y) & Cov(a_x,v_x) & Cov(a_x,v_y) & Cov(a_x,x) & Cov(a_x,y) & Cov(a_x,\varphi) \\ Cov(a_{xm},a_y) & Cov(a_{ym},a_y) & Cov(angle_z,a_y) & Cov(a_x,a_y) & Cov^2(a_y) & Cov(a_y,v_x) & Cov(a_y,v_{y)} & Cov(a_y,x) & Cov(a_y,y) & Cov(a_y,\varphi) \\ Cov(a_{xm},v_x) & Cov(a_{ym},v_x) & Cov(angle_z,v_x) & Cov(a_x,v_x) & Cov(a_y,v_x) & Cov^2(v_x) & Cov(v_x,v_y) & Cov(v_x,x) & Cov(v_x,y) & Cov(v_x,\varphi) \\ Cov(a_{xm},v_y) & Cov(a_{ym},v_y) & Cov(angle_z,v_y) & Cov(a_x,v_y) & Cov(a_y,v_{y)} & Cov(v_x,v_y) & Cov^2(v_y) & Cov(v_y,x) & Cov(v_y,y) & Cov(v_y,\varphi) \\ Cov(a_{xm},x) & Cov(a_{ym},x) & Cov(angle_z,x) & Cov(a_x,x) & Cov(a_y,x) & Cov(v_x,x) & Cov(v_y,x) & Cov^2(x) & Cov(x,y) & Cov(x,\varphi) \\ Cov(a_{xm},y) & Cov(a_{ym},y) & Cov(angle_z,y) & Cov(a_x,y) & Cov(a_y,y) & Cov(v_x,y) & Cov(v_y,y) & Cov(x,y) & Cov^2(y) & Cov(y,\varphi) \\ Cov(a_{xm},\varphi) & Cov(a_{ym},\varphi) & Cov(angle_x,\varphi) & Cov(a_x,\varphi) & Cov(a_y,\varphi) & Cov(v_x,\varphi) & Cov(v_y,\varphi) & Cov(x,\varphi) & Cov(y,\varphi) & Cov^2(\varphi) \end{bmatrix}$$

Additionally, the state estimate has a time tag denoted as **T**. For this sensor, it is configured to arrive every 5 ms.

$$T_k = 5ms$$

Step 1: Initialize System State

The Kalman filter initializes the system state with the first measurement. Here, the matrices z_1, R_1, x_1, and P_1 must be updated. The equations of the laws of motion may be applied to determining x_1, wherever applicable. T_1 is configured as 5 ms, i.e., the reception frequency of the IMU sensor.

Step 2: Prediction: Predict the System State Estimate

From the second measurement onward, the system state estimate is propagated forward to time align it with the measurement. This alignment is done so that the measurement and state estimate can be combined.

The system state estimate parameters x_p and P_P are computed at this state as follows:

$$x_P = Ax_{k-1}$$

$$P_P = AP_{k-1}A^T + Q$$

Q Matrix

The Q matrix represents process noise for the system model. The process noise for the system model using an IMU sensor is a 10-by-10 diagonal matrix.

H Matrix

The Kalman filter uses the state-to-measurement matrix, H, to convert the system state estimate from the state space to the measurement space. It transforms the predicted state vector and covariance matrix into the same form as the input measurement.

The state-to-measurement matrix for the IMU sensor is a 10-by-10 matrix called H_{IMU}, as only a_{xm}, a_{ym} and $angle_z$ are measured from the IMU sensor.

$$H_{IMU} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 3: Compute the Kalman gain

The Kalman filter computes a Kalman gain for each new measurement that determines how much the input measurement will influence the system state estimate. In other words, when a really noisy measurement comes in to update the system state, the Kalman gain will trust its current state estimate more than this new inaccurate information.

The Kalman gain, **K** at every measurement is updated as below:

$$K = P_P H_{IMU}^T (H_{IMU} P_P H_{IMU}^T + R)^{-1}$$

Step 4: Update: Estimate the System State and System State Error Covariance Matrix

After the Kalman Gain is computed, the new system state estimate, x_k, and system error covariance, P_k, are computed in the following order:

$$x_k = x_P + K(z_{k_{IMU}} - H_{IMU} x_P)$$

$$P_k = P_P(1 - K H_{IMU})$$

Thus, the IMU sensor itself is used to predict and update the optimal values for the position determination every 5 ms, independent of the position values from the Faulhaber decoder.

5.2 Sensor fusion using the Kalman filter for improved dead reckoning in the Faulhaber decoder using an IMU sensor

Only the car position parameters x, y and φ can be determined by the Faulhaber decoder. The Faulhaber decoder's position measurements will be represented as a 10-by-1 column vector $\mathbf{z_{FH}}$:

$$z_{k_{FH}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ x \\ y \\ \varphi \end{bmatrix}$$

The corresponding H matrix is given by:

$$H_{FH} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

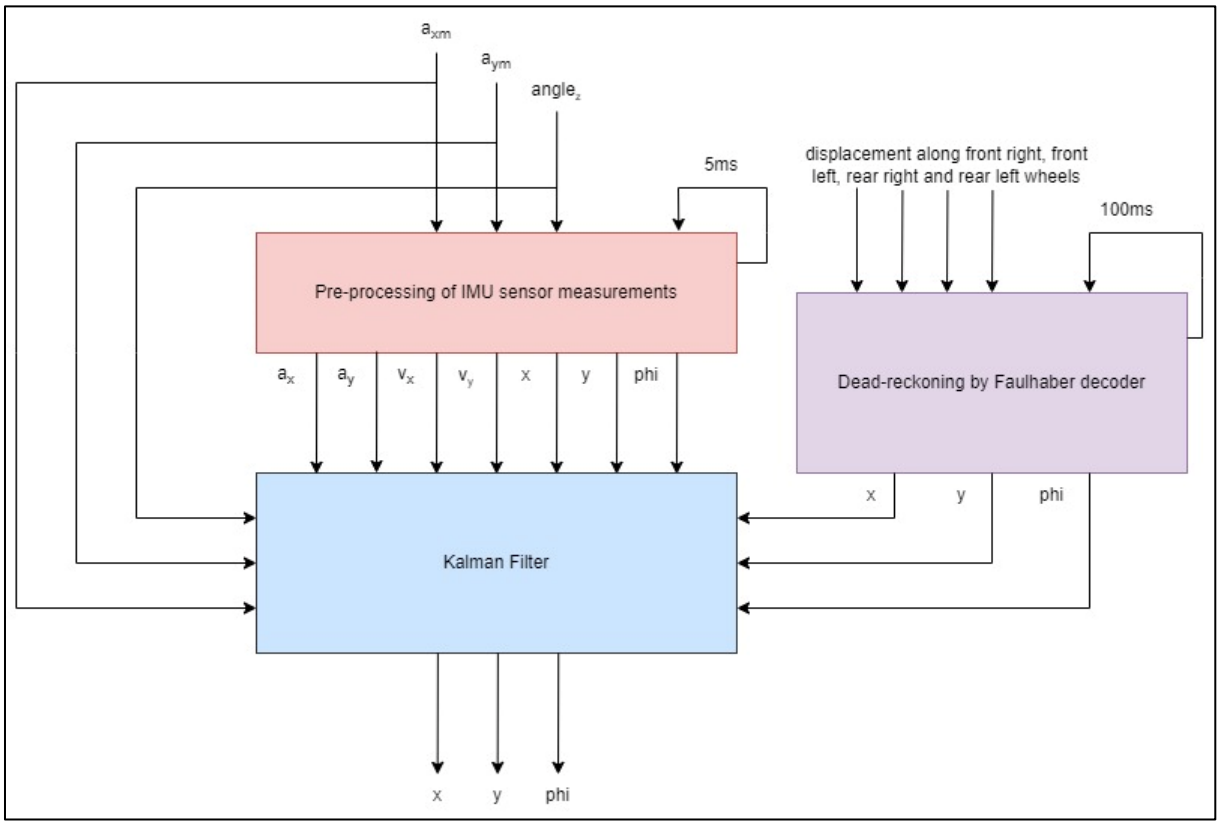

Figure 5.1: Sensor fusion of IMU sensor and Faulhaber decoder using Kalman filter for position determination.

The Faulhaber decoder determines the position every 100 ms, i.e., slower than the IMU sensor, which determines the position every 5 ms, as shown in Figure 5.1. Further, since the Faulhaber decoder directly provides the position value whenever its measurement is received, we recalculate the Kalman gain and compute the update step (Step 4) of the Kalman filter only, in the order described below:

$$K = P_P H_{FH}^T (H_{FH} P_P H_{FH}^T + R)^{-1}$$

$$x_k = x_P + K\left(z_{k_{FH}} - H_{FH} x_P\right)$$

$$P_k = P_P (1 - K H_{FH})$$

It is important to note that the prediction x_P used in the computation at this step is the prediction computed using the measurement from the IMU sensor.

Thus, the sensor fusion of an IMU sensor and a Faulhaber decoder, measured at different measurement rates, aids in the improvement of the dead reckoning done by the Faulhaber decoder.

5.3 The merit of using the Kalman filter

The state estimate computed is the only state history the Kalman Filter retains. As a result, Kalman filters can be implemented on machines with low memory restrictions.

6 Summary

A Kalman filter is an algorithm used for estimating the state of the system using past observations and current measurements, where both the observations and measurements are possibly noisy.

The Kalman filter could be understood as a loop executing prediction and updating for position determination, as illustrated in Figure 6.1.

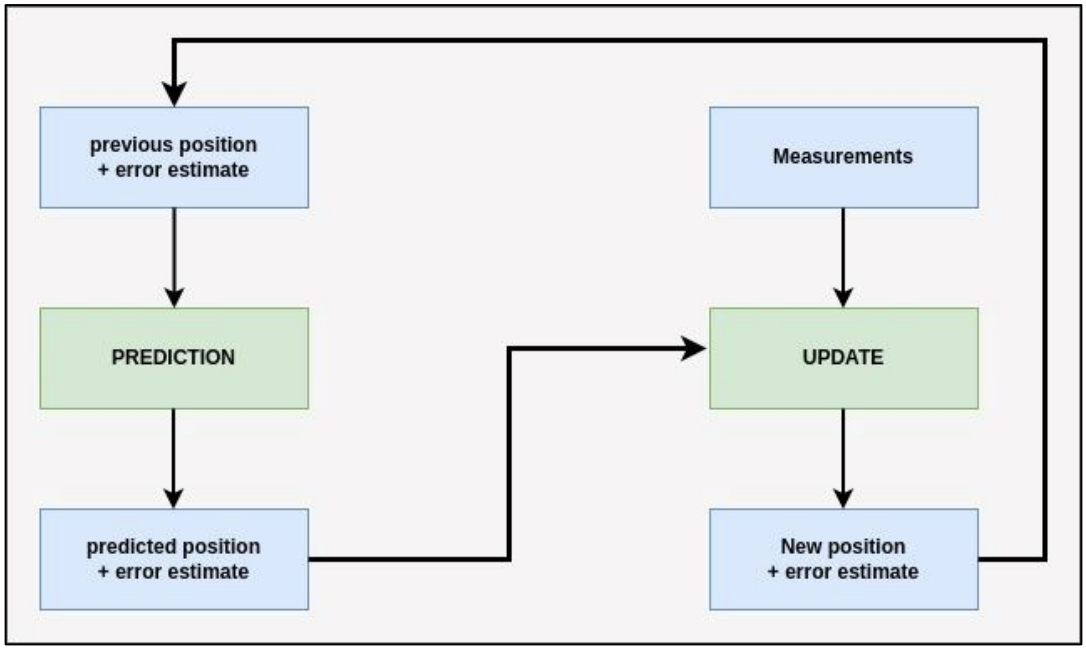

Figure 6.1: Elaboration of a Kalman filter as a loop for position determination.

The algorithm is achieved in two steps: prediction and update.

1. Prediction
   i) The observations are made using the IMU sensor alone when measurements are received every 5 ms.
   ii) The measurement parameters ax, ay, and anglez are used to predict the position coordinates "x" and "y" as well as the angle data "phi."
   iii) The noise from the IMU sensor is also modeled and taken into consideration.

2. Update
   There are two different ways in which the update step is executed in the process.
   i) In the absence of the Faulhaber decoder, the IMU sensor-based predictions themselves go through the update step, in order to determine the optimal position coordinates "x" and "y" and angle data "phi."
   ii) In the presence of the Faulhaber decoder, the Faulhaber decoder's measurements of position coordinates "x" and "y" and angle data "phi," at every 100 ms, are used to update the predictions made by the IMU sensor, thus determining the optimal position coordinates "x" and "y" and angle data "phi."

At update step 2.ii, a sensor fusion of an IMU sensor and a Faulhaber decoder using the Kalman filter is achieved.

7 Conclusion

The sensor fusion of an IMU sensor measuring every 5 ms and a Faulhaber decoder measuring every 100 ms successfully determines the most optimal car position coordinates x, y, and φ, using a Kalman filter. This in turn improves the dead reckoning determined by the Faulhaber decoder.

8 Future Scope

The system model that determines the position using the Faulhaber decoder can be improved by designing a corresponding Kalman filter.

9 References

1. Kalman Filter Explained Simply, https://thekalmanfilter.com/kalman-filter-explained-simply/
2. Kalman Filter Python Example – Estimate velocity from position, https://thekalmanfilter.com/kalman-filter-python-example/
3. Visually Explained: Kalman Filters, https://www.youtube.com/watch?v=IFeCIbljreY
4. Understanding Sensor Fusion and Tracking, https://www.youtube.com/watch?v=6qV3YjFppuc&list=RDCMUCgdHSFcXvkN6O3NXvif0-pA&start_radio=1&rv=6qV3YjFppuc&t=4
5. Introduction to State-Space Equations, https://www.youtube.com/watch?v=hpeKrMG-WP0
6. The Kalman Filter: An algorithm for making sense of fused sensor insight, https://towardsdatascience.com/kalman-filter-an-algorithm-for-making-sense-from-the-insights-of-various-sensors-fused-together-ddf67597f35e