Report Title:

Theory and Design Choices for EKF–GNN Radar–Camera Fusion

Report Objectives:

- Document foundational knowledge of the Kalman Filter and its applications.
- Review literature on EKF-based radar–camera data fusion systems.
- Analyze and justify engineering decisions in designing the EKF fusion pipeline.
- Summarize the full fusion and target tracking pipeline, enabling the development of the MATALB code for the system from scratch.

Work Contribution:

This document was prepared by Shayma Alteneiji

Table of Contents

Kalman Filter

In this section, we lay the foundations of the core component of the fusion algorithm: the Kalman filter. The radar and camera each provide complementary information about the scene, but both are affected by measurement noise and hardware limitations. To generate reliable and continuously updated target tracks, with the target's position and velocity, we must consider two elements: the noise in the sensor data and the dynamics of the moving targets.

Estimation methods that rely solely on measurements, such as RLS, are ineffective here because they assume static or slowly varying parameters. In contrast, targets motions are dynamic with time and must be modelled explicitly through a motion profile. However, the motion model alone cannot fully capture real-world target behavior due to the uncertainty in their motion.

The Kalman filter addresses both issues, by:

(1) predicting the state using the motion model, and
(2) correcting that prediction using sensor measurements.

This prediction–update cycle runs at every time step, gradually stabilizing the estimates and producing smooth, reliable tracks. Under the assumption of white Gaussian noise, the Kalman filter is the optimal linear estimator. Since in our system, the radar provides a nonlinear measurement of the target's radial velocity, the Extended Kalman Filter (EKF) is used, which linearizes the measurement equation around the nominal point, prior state estimate (to be defined in the upcoming subsections).

This report presents the essential theoretical background: the state space model, the discrete-time state-space model, observability considerations, LS, RLS, and the EKF prediction and update equations, the selected constant-acceleration motion model, the estimation of the sensor noise covariance matrices, the data association and target tracking block diagram.

State Space Model

A state space model is used to represent how the states and output of a system evolve with time, which is dependent on the system's internal modes or dynamics. It is also dependent on initial states and / or a control input. The system state equation and the output equation, as well as the dimensions of each vector and matrix are given below.

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \qquad \dot{\boldsymbol{x}}, \boldsymbol{x} \ni \mathbb{R}^{n\times 1}, \boldsymbol{A} \ni \mathbb{R}^{n\times n}, \boldsymbol{B} \ni \mathbb{R}^{n\times r}, \boldsymbol{u} \ni \mathbb{R}^{r\times 1}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \qquad \boldsymbol{y} \ni \mathbb{R}^{q\times 1}, \boldsymbol{C} \ni \mathbb{R}^{r\times n}, \boldsymbol{D} \ni \mathbb{R}^{q\times r}$$

What are the system states? The system states are the minimum number of variables that if known at t = t_0 together with the input for t > t_0 allows you to determine the system output for all future time t>t_0

[1]; and the state space is an n – dimensional space that represents all possible states the system can only potentially reach.

The solution to the system model, $\boldsymbol{y(t)}$ $\boldsymbol{and}$ $\boldsymbol{x(t)}$, are given below (assuming the feedforward matrix $\boldsymbol{D}$ is zero). These can be obtained by either solving the differential equation or taking Laplace or Fourier transform on both sides and their inverses to return back to the solution's time domain expression.

$$\boldsymbol{x}(t) \;=\; \boldsymbol{x}(t_0)\cdot e^{\boldsymbol{A}(t-t_0)} + \int_{t_0}^{t} e^{\boldsymbol{A}(t-\tau)}\cdot \boldsymbol{B}\cdot \boldsymbol{u}(t-\tau)d\tau$$

$$\boldsymbol{y}(t) \;= \boldsymbol{C}\cdot\; \boldsymbol{x}(t_0)\cdot e^{\boldsymbol{A}(t-t_0)} + \boldsymbol{C}\cdot \int_{t_0}^{t} e^{\boldsymbol{A}(t-\tau)}\cdot \boldsymbol{B}\cdot \boldsymbol{u}(t-\tau)d\tau$$

The $\boldsymbol{A}$ matrix is the State Transition Matrix. Since we observe the matrix $\boldsymbol{A}$ to be (1) both in the zero – input and the zero – state parts of the system solution, and (2) it is taken as an exponent, the properties of the matrix directly reflect the system's structure and determines its stability, generally describing whether the output of the system converges for bounded inputs.

While most system's dynamic is described by continuous equations, however, they are processed by microprocessor which can operate on only a discrete set of values [1], for that, continuous system state equations are usually quantized, as well as the system measurement equation. In the next section, we show how to derive the state equation model of a discrete time system.

State Space Model of Discrete Time Systems

Assume (1) the sampling period is $\Delta t \;=\; t_k \;-\; t_{k-1}$ [1], (2) the system is piece -wise constant, meaning the control input, $\boldsymbol{u}(t)$, and the state transition matrix, $\boldsymbol{A}$, and the input matrix, $\boldsymbol{B}$, are constant over the sampling period from t_{k-1} to t_k [1], then, the state at the future time (t_k), can be expressed with respect to its current value, $\boldsymbol{x}(t_{k-1})$, as given below [1].

$$\boldsymbol{x}(t_k) \;=\; \boldsymbol{x}(t_{k-1})\cdot e^{\boldsymbol{A}(t_k-t_{k-1})} + \int_{t_k-1}^{t_k} e^{\boldsymbol{A}(t_k-\tau)}\cdot \boldsymbol{B}\cdot u(t_k-\tau)d\tau$$

To expresses the integral as a function of Δt, we use the substitution variable (α) in place of $t_k - \tau$. The resulting expression is shown below [1].

$$\boldsymbol{x}(t_k) \;=\; \boldsymbol{x}(t_{k-1})\cdot e^{\boldsymbol{A}(\Delta t)} + u(\alpha)\int_{0}^{\Delta t} e^{\boldsymbol{A}(\alpha)}\cdot \boldsymbol{B}\; d\alpha$$

Comparing this equation to the continuous time state space model equation, we can define the discrete time system state equation as follows:

$$\boldsymbol{x_k} = \boldsymbol{F} \cdot \boldsymbol{x_{k-1}} + \boldsymbol{G} \cdot \boldsymbol{u_{k-1}}$$

Were:

$$\boldsymbol{F} = e^{\boldsymbol{A}(\Delta t)}, \text{ and } \boldsymbol{G} = \int_0^{\Delta t} e^{\boldsymbol{A}(\alpha)} \cdot \boldsymbol{B}\, d\alpha$$

Depending on the system, samples maybe are taken at uniform periods or non – uniform sampling maybe performed, in which case Δt is not constant. $\boldsymbol{F}$ and $\boldsymbol{G}$ are then a function of time and written as $\boldsymbol{F_{k-1}}$ and $\boldsymbol{G_{k-1}}$. $\boldsymbol{x_{k-1}}$ represents the current state value, and $\boldsymbol{u_{k-1}}$ represents the current input applied. The output equation is given below, the discrete time matrix $\boldsymbol{H}$ matrix maps the states to their respective output measurements [2].

$$\boldsymbol{y_k} = \boldsymbol{H} \cdot \boldsymbol{x_k}$$

In the next section, we discuss observability, which discusses the conditions under which we can re-construct our states given only the output measurements.

Observability, Constructability, and the Kalman Filter

The observability of a linear system is a critical concept in a Kalman filter, since the Kalman filter is fundamentally an observer [3]. Hence for a given system, it must be verified that it is observable, to guarantee the constructability of its states from the output measurement captured [4]. For a discrete time system, the observability definition is given as follows:

"*A discrete-time system is observable if for any initial state* $\boldsymbol{x_0}$ *and some final time k the initial state* $\boldsymbol{x_0}$ *can be uniquely determined by knowledge of the input* $\boldsymbol{u_i}$ *and output* $\boldsymbol{x_i}$ *for all* $i \in [0, k]$." [1]

While there are multiple definitions for what makes a system observable [1]. The selection of the method used depends on its computational ease in a given scenario [1]. We present here the widely known definition:

"*The n-state discrete linear time-invariant system*

$$\boldsymbol{x_k} = \boldsymbol{F} \cdot \boldsymbol{x_{k-1}} + \boldsymbol{G} \cdot \boldsymbol{u_{k-1}}$$

$$\boldsymbol{y_k} = \boldsymbol{H} \cdot \boldsymbol{x_k}$$

has the observability matrix Q defined by

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{HF} \\ \vdots \\ \boldsymbol{HF^{n-1}} \end{bmatrix}$$

The system is observable if and only if p(Q) = n." [1]

The matrix Q has full rank if *either* **(1)** the H matrix has full rank [1, 5], meaning in the Jordon form of matrix $\boldsymbol{H}$ ($\bar{\boldsymbol{H}} = \mathbf{HM}, \bar{\mathbf{F}} = \mathbf{M^{-1}FM}, \bar{\mathbf{G}} = \mathbf{M^{-1}G}$, where the $\mathbf{M}$ matrix is the matrix of linearly independent eigenvectors of the matrix F) its states are coupled to the output through a non – zero row in $\bar{\boldsymbol{H}}$ [1, 5]. The rank of the H matrix (linearly independent rows or columns) determines the number of directions in which we can be observe our states in $\mathbb{R}^{n\times 1}$ [5].

(2) When H is not full rank, the system dynamics defined by F can help allow the states of the system become observable after some n dynamic steps in time [5]. The Linearly independent columns or rows of F help complement those of H such that all states in (n) possible different directions can be observed over time [5].

The estimation algorithms such as LS, weighted LS, or RLS are generalizations of the Kalman filter which assumes dynamic systems. In the next section we discuss these algorithms before moving forward with discussing the relations between them and the Kalman filter in detail.

Least Squares and Weighted Least Squares Estimation Algorithms

There is a vector $\boldsymbol{x}$ is an n – dimensional column vector that we attempt to estimate. We cannot directly measure $\boldsymbol{x}$ but we can measure $\boldsymbol{y}$, yet since no measurement device achieved infinite precision, there is a noise v_k associated with every measurement y_k. A matrix $\boldsymbol{H}$ describes the mapping from the unknow vector $\boldsymbol{x}$ to vector $\boldsymbol{y}$.

$\boldsymbol{x} \ni \mathbb{R}^{\boldsymbol{n\times 1}}, \boldsymbol{y} \ni \mathbb{R}^{\boldsymbol{k\times 1}}, \boldsymbol{H} \ni \mathbb{R}^{\boldsymbol{k\times n}}, \boldsymbol{v} \ni \mathbb{R}^{\boldsymbol{k\times 1}}$. When k > n, our system becomes over constrained, or overdetermined, we have more measurements than we have variables [1, 6]. The maximum dimension that can be spanned by H is min (k, n) = n. We, therefore, usually cannot express $\boldsymbol{y}$ to be exactly equal to $\boldsymbol{Hx}$ due to the presence of noise, we always tend to obtain as many measurements (k) to improve our estimate. H can only span a subspace of the space $\mathbb{R}^{\boldsymbol{k\times 1}}$. We express y as shown below. We obtain the best approximate for x, when the error is minimized.

$$\boldsymbol{y} = \boldsymbol{H\hat{x}} + \boldsymbol{e}$$

Envision the geometric vector $\boldsymbol{y}$, represented as a sum its projection into the column space of $\boldsymbol{H}$, which is the first term $\boldsymbol{H\hat{x}}$ [6] (the coordinates of $\hat{\boldsymbol{x}}$ forms a linear combination of the columns of $\boldsymbol{H}$, and hence $\boldsymbol{H\hat{x}}$ is an element of the Col(A) space). The vector $\boldsymbol{e}$ joins the end of $\boldsymbol{H\hat{x}}$ vector to the $\boldsymbol{y}$ vector [6].

Moreover, we can geometrically envision that the optimal approximation of $\boldsymbol{x}$ is one such that $\boldsymbol{H\hat{x}}$ is the orthogonal projection of $\boldsymbol{y}$ into Col(H), and $\boldsymbol{e}$ being orthogonal to space spanned by the Columns of H [6, 7]. This is also illustrated in Figures 1 and 2. All other vectors forms of e for different $\boldsymbol{H\hat{x}}$ would yield a greater distance e [7].

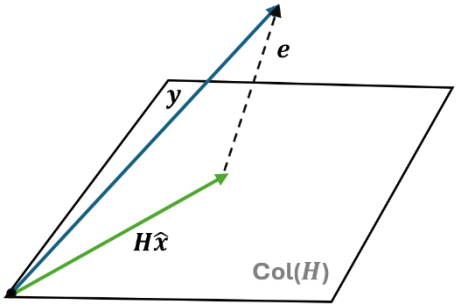


Figure 1. Projection of the y matrix on the column space of H, when e is not orthogonal

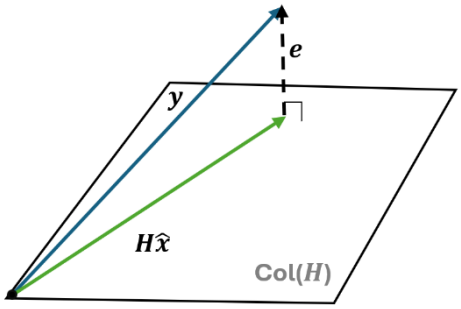


Figure 2. Projection of the y matrix on the column space of H, when e is not orthogonal

We can formulate the problem mathematically to obtain the expression for $\hat{\boldsymbol{x}}$. We formulate the problem of minimize the error distance $\boldsymbol{e}$ as follows. We define the cost function (J) of which we want to minimize to the be the sum of the squares of the difference between the measured values y and the approximation $H\hat{x}$, **J** is given as shown below [1]. We choose minimizing the sum of the squares instead of for example minimizing the absolute values, since it is differentiable while the latter is not.

$$\boldsymbol{J} = (y_1 - H\hat{x}_1)^2 + (y_2 - H\hat{x}_2)^2 + (y_3 - H\hat{x}_3)^2 + \ldots + (y_k - H\hat{x}_k)^2$$

$$\boldsymbol{J} = ||\, \boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{x}} \,||^2 = (\boldsymbol{y} - H\hat{\boldsymbol{x}})^{\boldsymbol{T}} \cdot (\boldsymbol{y} - H\hat{\boldsymbol{x}})$$

$$\boldsymbol{J} = \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{y} - \hat{\boldsymbol{x}}^{\boldsymbol{T}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{y} - \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{H}\hat{\boldsymbol{x}} + \hat{\boldsymbol{x}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\hat{\boldsymbol{x}}$$

Next, to minimize J as a function $\hat{\boldsymbol{x}}$ we find its partial derivative and set it to zero (refer to matrix calculus formulas) [1]. For $\hat{\boldsymbol{x}}$ to exist, $\boldsymbol{H}$ must be full rank, the measurement needs to be linearly independent [1].

$$\frac{\boldsymbol{\partial J}}{\boldsymbol{\partial \hat{x}}} = -2\boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{H} + 2\hat{\boldsymbol{x}}^{\boldsymbol{T}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H} = \boldsymbol{0}$$

$$\left(\hat{\boldsymbol{x}}^{\boldsymbol{T}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\right)^{\boldsymbol{T}} = \left(\boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{H}\right)^{\boldsymbol{T}}$$

$$\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\hat{\boldsymbol{x}} = \boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{y}$$

$$\hat{\boldsymbol{x}} = \left(\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\right)^{-\boldsymbol{1}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{y}$$

In Weighted Least Squares, the variance of the noise ($\boldsymbol{v}$) associated with each measurement is assumed to be known [1]. The noise is also assumed to be zero-mean and independent [1]. The autocovariance matrix of the noise is given by,

$$R = E((\boldsymbol{v} - \bar{v})(\boldsymbol{v} - \bar{v})^T) = E(vv^T)$$

$$R = \begin{bmatrix} E(v_1^2) & \ldots & E(v_1 v_k) \\ \vdots & \ldots & \vdots \\ E(v_k v_1) & \ldots & E(v_k^2) \end{bmatrix} = \begin{bmatrix} E(v_1^2) & \ldots & E(v_1)E(v_k) \\ \vdots & \ldots & \vdots \\ E(v_k)E(v_1) & \ldots & E(v_k^2) \end{bmatrix}$$

Therefore, R is simplified as,

$$R = \begin{bmatrix} \sigma_1^2 & \dots & 0 \\ \vdots & \dots & \vdots \\ 0 & \dots & \sigma_k^2 \end{bmatrix}$$

The lower the variance, the more compact (less spread) the distribution of out measurement and the more certain we are about the accuracy of the measurement, and vice versa. The cost function must be penalized when the error is greater for more accurate measurements than for noisier measurements. We incorporate the measurement noise into the cost function J as shown below [1].

$$\boldsymbol{J} = \frac{(y_1 - H\hat{x}_1)^2}{\sigma_1^2} + \frac{(y_2 - H\hat{x}_2)^2}{\sigma_2^2} + \frac{(y_3 - H\hat{x}_3)^2}{\sigma_3^2} + \dots + \frac{(y_k - H\hat{x}_k)^2}{\sigma_k^2}$$

The inverse of a diagonal matrix is matrix with the reciprocal of the elements along the diagonal, hence:

$$\boldsymbol{J} = (\boldsymbol{y} - \boldsymbol{H\hat{x}})^{\boldsymbol{T}} \boldsymbol{R^{-1}} (\boldsymbol{y} - \boldsymbol{H\hat{x}})$$

Performing the same steps for the LS, we obtain the expression for $\hat{\boldsymbol{x}}$ as [1]:

$$\hat{\boldsymbol{x}} = \left(\boldsymbol{H^T R^{-1} H}\right)^{-1} \boldsymbol{H^T R^{-1} y}$$

For the expression of $\hat{\boldsymbol{x}}$ to exist, the noise autocovariance matrix must be non-singular, all its eigenvalues must non-zero, meaning all measurements must have some non-zero noise [1], and as with LS, $\boldsymbol{H}$ must be full rank. In the next section, we discuss the recursive least squares, a computationally efficient method of determining the optimal estimate of $\boldsymbol{x}$.

Recursive Least Squares Estimation Algorithm

Assume we are solving the problem of obtaining the parameters of a discrete time FIR (finite impulse response) filter, which could have M = 64, 128, 256, or 1024 parameters. The parameters represent the $\boldsymbol{x}$ vector we attempt to approximate; the $\boldsymbol{y}$ vectors represent the data obtained of the filter response to a know input.

The equation below expresses the response of a filter as function of discrete finite input values and finite discrete response values using convolution. Using LS to approximate the w[k] parameter would involve finding the inverse of large matrices, and as the number of measurements increases, the computation costs increase significantly. Therefore, it is an inefficient algorithm.

$$y[n] = \sum_{k=0}^{M-1} w[k]\, x[n-k]$$

The RLS algorithm to estimate the x vector using only the current measurement and the previous iteration estimation [1], hence recursively improving the estimation of x as more measurements arrive without require as much computational cost.

The RLS estimator is written in the form shown below:

$$y_k = \boldsymbol{H_k x} + v_k$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k-1}} + K_k(\boldsymbol{y_k} - \boldsymbol{H_k}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}})$$

$\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}$ is the previous estimate, $y_k - \boldsymbol{H_k}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}$ is the correction term, and $K_k$is the estimation gain matrix [1]. The gain matrix, K_k determines how much trust, or weight is put on the new measurement y_k when updating the estimate of $\boldsymbol{x}$ for t = k. Intuitively, therefore, the gain matrix is expected to be a an inverse function of the current measurement noise, the greater the noise present in the measurement, the smaller the gain, the smaller the effect the new measurement has on updating our estimate of $\boldsymbol{x}$.

Moreover, we track how close our estimate is to the true value of $\boldsymbol{x}$ using the autocovariance of the RV $\boldsymbol{x}$ that we attempt to estimate. In order to ensure that the covariance decays as more measurements are obtained, we determine the optimal value of $\boldsymbol{K_k}$ which shrinks it at every iteration [1]. We recursively update $\boldsymbol{K_k}$ based on previous update covariance and current covariance. The greater the gain, $\boldsymbol{K_k}$, the more reliable the measurement is, the more correction applied, and the faster our convergence to the smallest possible uncertainty of the $\hat{\boldsymbol{x}}_{\boldsymbol{k}}$ RV estimate.

Source [1] shows the detailed mathematical expression for $\boldsymbol{P_k}$ equation given below (pg. 84 - 85), simplifying the expression $E[(\boldsymbol{x} - \hat{\boldsymbol{x}}_{\boldsymbol{k}})^T(\boldsymbol{x} - \hat{\boldsymbol{x}}_{\boldsymbol{k}})]$ as:

$$\boldsymbol{P_k} = (\boldsymbol{I} - \boldsymbol{K_k H_k})\boldsymbol{P_{k-1}}(\boldsymbol{I} - \boldsymbol{K_k H_k})^{\boldsymbol{T}} + \boldsymbol{K_k R_k K_k^T}$$

This equation assumes that **(1)** the noise v_k is zero – mean. **(2)** The noise vector and the error of the estimation at $t_{k-1}$are independent, thus the expect value of there multiplication is zero.

As stated above, the equation for $\boldsymbol{K_k}$ is determined such that the variances of the n- elements RV $\boldsymbol{x}$ is minimized, which is the also the trace of $\boldsymbol{P_k}$ matrix [1]. We define **J**, the cost function as follows,

$$J_k = E[(x_1 - \hat{x}_1)^2] + \cdots + E[(x_k - \hat{x}_k)^2] = E[(\boldsymbol{x} - \hat{x}_k)^T(\boldsymbol{x} - \hat{x}_k)] = E\big[Tr[(\boldsymbol{x} - \hat{x}_k)(\boldsymbol{x} - \hat{x}_k)^T]\big]$$

$$\boldsymbol{J_k} = Tr\,\boldsymbol{P_k}$$

$\boldsymbol{K_k}$ is found by finding the partial derivative of $\boldsymbol{J_k}$ with respect $\boldsymbol{K_k}$ and set to zero. Solving for $\boldsymbol{K_k}$ we obtain the following equation,

$$\boldsymbol{K_k} = \boldsymbol{P_{k-1}H_k^T} \cdot (\boldsymbol{H_k P_{k-1} H_k^T} + \boldsymbol{R_k})^{-1}$$

Next, we explore the bias of our RLS estimator, which is through finding the mean of the estimation error at time *k*, the simplified equation from [2] is given below,

$$E(\boldsymbol{x} - \hat{x}_k) = (\boldsymbol{I} - \boldsymbol{K_k H_k}) E(\boldsymbol{x} - \hat{x}_{k-1}) - \boldsymbol{K_k} E(v_k)$$

The RLS estimator has zero bias for the same conditions made for the equation $\boldsymbol{P_k}$, which means that the RLS estimation of x should on average be equal to the true value of x [1]. This is satisfied for all values of the gain Kk, hence, with the gain optimized to minimize the cost function, the estimation error should also be regularly close to zero [1]. Moreover, the RLS algorithm assumes that the noise at time k is uncorrelated to the noise at time i for all integers expect when i =k, that is, $E(\boldsymbol{v_k v_i}) = R_k \delta_{k-i}$ [1] (R_k being the autocorrelation for the noise signal), this means the noise is white and the augmented R matrix for all measurements is diagonal. In the next section, we explore generalizing from the RLS estimator to the Kalman filter under dynamic systems.

RLS Estimation to Kalman Filter

As mentioned in the introduction, the RLS algorithm estimates a constant or a slowly varying random variable (RV), while the Kalman filter assumes the system is dynamic, meaning the variable we attempt to estimate is governed by dynamics we can model and are somewhat certain about. For that reason, we refer to the variable x in the Kalman filter as a stochastic process, a random variable whose characteristics or distribution vary with time [1]. The random variable state may be multivariate, meaning it involves multiple variables of different characteristics considered jointly.

In RLS, the algorithm has one stage comprising three steps: (0) initialize the estimate for the states x and the estimation error covariance matrix P0; (1) calculate the gain matrix Kk; (2) estimate x using Kk, the correction term, and the previous estimate; (3) calculate the covariance matrix of the estimate at t=k. In the second iteration, the gain matrix is updated, taking into account the previous iteration's covariance matrix and the current measurement noise.

In the Kalman filter, however, we have two stages: the prediction step and the measurement (or time-update) step. Since the state we are tracking is governed by a known, modelled dynamic, we cannot simply refine the estimate based on measurements and previous values. The state is not constant, its previous values do not provide sufficient information about where it could be in the future.

This highlights the importance of the prediction step: we use the model we have of the state, which may include some modelled noise denoted as Q, and refine this estimate using the measurement. How much we allow the new measurements to contribute to estimating the state at t=k depends on how noisy the measurement is relative to the modelling or process noise Q. We will see in detail how the

prediction step estimates the state prior to measurement by propagating forward how the mean of the state should change with time [1], which is what we are attempting to track.

Over time, the mean of the state estimation error is zero, meaning our estimator makes the Kalman filter, on average, unbiased, neither consistently above nor below the true expected value of x. Of course, the same conditions that ensure unbiasedness in RLS also apply to the Kalman filter: the measurement noise vk must be zero-mean and uncorrelated with the signal.

The gain matrix Kk in the Kalman filter is updated based on the dynamic model of the system, the measurement noise, the modelling noise, and the previous covariance, thus allowing it to adaptively refine its gain matrix in a way that decreases the current covariance. For the state estimate after measurement, Kk balances the weight placed on the prediction and the measurement. We explore the details of the derivation and further insights in the upcoming sections.

Kalman Filter Equations

As was lightly touched on in the previous section, after convergence, the Kalman filter should in average have a zero-estimation error (given the conditions mentioned are met), as the KF estimation equals to the true mean of the target's states. The Kalman filter predicts future states by modelling how the states' mean evolves over time [1]. Using the system model, we express future states as a function of the state transition matrix, the mean of the previous state, the input signal (if any), and process noise [1]. For zero-mean noise, which is a requirement for an unbiased KF, the last term disappears, as shown below.

$$E(\boldsymbol{x_k}) = \boldsymbol{F_{k-1}} E(\boldsymbol{x_{k-1}}) + \boldsymbol{G_{k-1}} \boldsymbol{u_{k-1}} + E(\boldsymbol{w_{k-1}})$$

$$\overline{x_k} = \boldsymbol{F_{k-1}} \overline{x_{k-1}} + \boldsymbol{G_{k-1}} \boldsymbol{u_{k-1}}$$

The next property of interest is the covariance of the state estimation error. Substituting the above equation into the covariance expression gives the discrete-time Lyapunov (or Stein) equation [1]:

$$E[(\boldsymbol{x} - \overline{x_k})(\boldsymbol{x} - \overline{x_k})^T] = \boldsymbol{F_{k-1}} \boldsymbol{P_{k-1}} \boldsymbol{F_{k-1}}^{\boldsymbol{T}} + \boldsymbol{Q_{k-1}}$$

We cannot rely solely on measurements, as in RLS, as the tracked states are not constants. Instead, we use the model to predict the expected future state and correct this estimate using the measurements. The Kalman filter defines two estimates for x, the priori and posteriori estimates, each of whose equation are given below [1].

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-} = E[\boldsymbol{x_k} | \boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_{k-1}}] \textbf{ (priori)}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{+} = E[\boldsymbol{x_k} | \boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_k}] \textbf{ (posteriori)}$$

The posteriori estimates at $t = k$ refers to the mean of x after including the measurement at $t = k$. The more measurements available, the better the estimate, as noise averages out. The covariance of the estimation error, P_k^+, always decreases after a measurement, with the rate of decrease depending on measurement noise.

We use the same RLS equations, incorporating priori and posteriori estimates:

Prediction Step*:* The prediction is based on the model, providing the estimate of x at $t = k$ before the measurement y_k. We substitute the posteriori estimate from $k - 1$ and use the Stein equation to predict how the posteriori covariance propagates. The two prediction equations are given below.

$$\boldsymbol{x_k^-} = \boldsymbol{F_{k-1}x_{k-1}^+} + \boldsymbol{G_{k-1}u_{k-1}}$$

$$\boldsymbol{P_k^-} = \boldsymbol{F_{k-1}P_{k-1}^+{F_{k-1}}^T} + \boldsymbol{Q_{k-1}}$$

Time – Update Step*:* The Kalman gain K_k balances the contribution of the measurement and the model prediction. Using the priori estimate, the posteriori estimate is updated, and the covariance is recalculated for the next prediction step:

$$\boldsymbol{K_k} = \boldsymbol{P_k^- H_k^T} \cdot (\boldsymbol{H_k P_k^- H_k^T} + \boldsymbol{R_k})^{-1}$$

$$\boldsymbol{x_k^+} = \boldsymbol{x_k^-} + K_k(\boldsymbol{y_k} - \boldsymbol{H_k x_k^-})$$

$$\boldsymbol{P_k^+} = (\boldsymbol{I} - \boldsymbol{K_k H_k})\boldsymbol{P_k^-}(\boldsymbol{I} - \boldsymbol{K_k H_k})^{\boldsymbol{T}} + \boldsymbol{K_k R_k K_k^T}$$

Here we provide the summary of the Kalman filter taken from source [1], (pg. 128 – 129).

“1. The dynamic system is given by the following equations:

$$\boldsymbol{x_k} = \boldsymbol{F_{k-1}x_{k-1}} + \boldsymbol{G_{k-1}u_{k-1}} + \boldsymbol{w_{k-1}}$$

$$y_k = \boldsymbol{H_k x_k} + v_k$$

$$E\,(\boldsymbol{w_k w_i^T}) = Q_k \delta_{k-i}$$

$$E\,(\boldsymbol{v_k v_i^T}) = R_k \delta_{k-i}$$

$$E\,(\boldsymbol{v_k w_i^T}) = 0$$

2. The Kalman filter is initialized as follows:

$$\boldsymbol{\hat{x}_0^+} = E(\boldsymbol{x_0})$$

$$\boldsymbol{P_0^+} = E[(\boldsymbol{x_0} - \boldsymbol{\hat{x}_0^+})(\boldsymbol{x_0} - \boldsymbol{\hat{x}_0^+})^{\boldsymbol{T}}]$$

1. The Kalman filter is given by the following equations, which are computed for each time step k = 1, 2,..:

$$\boldsymbol{P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}{}^T + Q_{k-1}}$$

$$\boldsymbol{x_k^- = F_{k-1} x_{k-1}^+ + G_{k-1} u_{k-1}} = \textit{a priori state estimate}$$

$$\boldsymbol{K_k = P_k^- H_k^T \cdot (H_k P_k^- H_k^T + R_k)^{-1}}$$

$$\boldsymbol{x_k^+ = x_k^- + K_k(y_k - H_k x_k^-)} = \textit{a posteriori state estimate}$$

$$\boldsymbol{P_k^+ = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T}$$ ”

Insights on the Kalman Filter Equations

If we examine the priori covariance equation closely, we see that it is simply the result of a linear transformation applied to the posteriori estimate of x from step $k-1$. Consider a random variable x with a Gaussian distribution and parameters (μ, C_X), where C_X is the autocovariance matrix. If y is a linear transformation of x, then y also follows a Gaussian distribution:

$$\boldsymbol{y = Ax + b}$$

$$\boldsymbol{x \sim N(\mu, C_X)}$$

$$\boldsymbol{y \sim N(A\mu + b, AC_X A^T)}$$

The Q matrix represents the covariance of the added processing noise. Recall that the state is a multivariate stochastic sequence with a Gaussian distribution. Its contours form ellipsoids, representing state values with equal probability [8]. In the prediction step, the ellipsoid either contracts or expands with each iteration, depending on the system’s stability and the amount of processing noise present (the role of F in stability will be discussed in the next section) [2, 8].

Another important aspect is the inverse term in the gain matrix, known as the innovation covariance. When the distribution of y is noisier or more spread out, the gain matrix is smaller. Consequently, the innovation term contributes less to the state estimation [1, 8].

For the posteriori covariance, recall that the gain matrix was selected to minimize the estimation error covariance in RLS. In P_k^+, the term $(I - K_k H_k)$ multiplies P_k^-, reducing the covariance. How much the covariance ellipsoid shrinks depends on K_k [1]. The lower the measurement noise covariance, the smaller the gain matrix, and the closer $(I - K_k H_k)$ is to the identity. This is why the convergence rate depends on measurement noise: it determines how much P_k^- is reduced, while P_k^+ is always less than P_k^- [1].

For the KF, the relative norm of R versus Q ultimately decide how the estimator weights the model versus the measurements. If Q>> R, then the P_k^- increases, K_k increases, and the KF relies more on

the measurement, reflecting modelling error [1]. Conversely, if Q << R, K_k decreases, and the estimator relies more on the model [1].

Finally, consider the initial estimate. If it is close to the true state, P_k^- is small for $k = 1$, resulting in a smaller Kalman gain K_k. The filter may initially give less weight to measurements, which could mathematically slow convergence. In practice, however, a good initial estimate may allow the filter to stabilize more readily and, conversely, speed up convergence. On the other hand, if the initial estimate is far from the true state, P_k^- is large for $k = 1$, and the filter relies more heavily on measurements. Over time, the initial estimate has minimal effect on performance [1].

Summary of assumption and conditions on the optimality of the Kalman Filter

We summarize below the assumptions that make the Kalman filter an optimal linear estimator, based on [1]. These could also be considered limitations, since these conditions may not be met perfectly in practical applications. In the next section, we show how the Jacobian is used to linearize the state and measurement models around a nominal point, which enables the Extended Kalman Filter to handle nonlinear systems.

1. The system state or dynamics equation is linear,
2. The system measurement equation is linear.
3. The measurement noise and the processing noise vectors follow are RV and follow a Gaussian distribution,
4. The noise and processing noise have zero-mean,
5. The measurement noise at time k, and the estimation error at time $(k-1)$ is independent,
6. The measurement noise is white, meaning $E\,(\boldsymbol{v_k v_i}^T) = \boldsymbol{R_k}\delta_{k-i}$, where $\boldsymbol{R_k}$ measurement noise covariance matrix,
7. The processing noise is white, meaning $E\,(\boldsymbol{w_k w_i}^T) = \boldsymbol{Q_k}\delta_{k-i}$, where $\boldsymbol{Q_k}$ measurement noise covariance matrix, and
8. The measurement and processing noise are independent.

Non - Linear Models and the Discrete-time Extended Kalman Filter

Assuming discretized dynamics of continuous-time systems, the Extended Kalman filter is derived as follows. For a system modelled as shown below [1].

$$\boldsymbol{x_k} = f_{k-1}(\boldsymbol{x_{k-1}}, \boldsymbol{u_{k-1}}, \boldsymbol{w}_{k-1})$$

$$\boldsymbol{y_k} = h_{\boldsymbol{k}}(\boldsymbol{x_k}, \boldsymbol{v_k})$$

$$\boldsymbol{v_k} \sim N(0, \boldsymbol{R_k})$$

$$\boldsymbol{w_k} \sim N(0, \boldsymbol{Q_k})$$

The state function f(.) can be linearized by expanding the function using the Taylor series around the nominal point $\boldsymbol{x_{k-1}} = \boldsymbol{x_{k-1}^+}$ and with $\boldsymbol{v_k} = 0$. For the measurement function, h(.), we linearize the function around the nominal point $\boldsymbol{x_{k-1}} = \boldsymbol{x_k^-}$ and with $\boldsymbol{w_k} = 0$, our best estimate of x following the prediction step. Assuming the system is slowly varying, and the state estimate is close to the nominal point, higher order derivatives of the Taylor series may be reasonably considered to be negligible.

The resulting system state equation, system measurement equation, as well as the resulting Jacobians are given below.

$$y_k = f_{k-1}(\hat{x}_{k-1}^+, 0) + \left.\frac{\partial f_{k-1}}{\partial x}\right|_{\hat{x}_{k-1}^+} (x_{k-1} - \hat{x}_{k-1}^+)$$

$$y_k = h_k(\hat{x}_k^-, 0) + \left.\frac{\partial h_k}{\partial x}\right|_{\hat{x}_k^-} (x_k - \hat{x}_k^-)$$

$$F_{k-1} = \left.\frac{\partial f_{k-1}}{\partial x}\right| \hat{x}_{k-1}^+$$

$$H_k = \left.\frac{\partial h_k}{\partial x}\right| \hat{x}_k^-$$

The discrete- time Extended Kalman filter summary from source [1] is give below: **“**

1) Initialize the filter as follows:

$$\hat{x}_0^+ = E(x_0)$$

$$P_0^+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]$$

2) For k=1,2, …, perform the following:
   (a) compute the following partial derivative matrix:

$$F_{k-1} = \frac{\partial f_{k-1}}{\partial x}\bigg|_{\hat{x}_{k-1}^{+}}$$

(b) Perform the time update of the state estimate and estimation-error-covariance as follows:

$$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + Q_{k-1}$$

$$\hat{x}_k^- = f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, 0)$$

c) Compute the following partial derivative matrix:

$$H_k = \frac{\partial h_k}{\partial x}\bigg|_{\hat{x}_k^-}$$

d) Perform the measurement update of the estate estimate and estimation-error covariance as follows:

$$K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k[y_k - h_k(\hat{x}_k^-, 0)]$$

$$P_k^+ = (I - K_k H_k) P_k^- \text{ ”.}$$

The EKF in Radar – Camera Fusion and the Constant Acceleration (CA) Model

In our application, the radar provides x, y, and v_r measurements, while the camera outputs u, v pixels (which are coordinate transformed to x, y measurements in the radar coordinate frame). The tracked state vector is defined as shown below.

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

The radial velocity is not linearly related to the states x, y, v_x and v_y requiring the usage of the EKF. The equation below expresses the radial velocity as a function of the states [9]:

$$v_r = \frac{xv_x + yv_y}{\sqrt{x^2 + y^2}}$$

To linearize this measurement, we apply the Jacobian to v_r, evaluated at the priori state estimate:

$$\left[\frac{\partial v_r}{\partial s}\right]_{\hat{\boldsymbol{x}}_{k-1}^-} = \left[\frac{\partial v_r}{\partial x} \quad \frac{\partial v_r}{\partial y} \quad \frac{\partial v_r}{\partial v_x} \quad \frac{\partial v_r}{\partial v_y}\right]_{\hat{\boldsymbol{x}}_{k-1}^-}$$

The resulting expression is [9]:

$$\left[\frac{\partial v_r}{\partial s}\right]_{\hat{\boldsymbol{x}}_{k-1}^-} = \left[\frac{y(yv_x - xv_y)}{(x^2+y^2)^{3/2}} \quad \frac{x(xv_y - yv_x)}{(x^2+y^2)^{3/2}} \quad \frac{x}{\sqrt{x^2+y^2}} \quad \frac{y}{\sqrt{x^2+y^2}}\right]_{\hat{\boldsymbol{x}}_{k-1}^-}$$

The measurement matrix and measurement vector for the radar are:

$$\boldsymbol{H_{radar}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{y(yv_x - xv_y)}{(x^2+y^2)^{3/2}} & \frac{x(xv_y - yv_x)}{(x^2+y^2)^{3/2}} & \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \end{bmatrix}$$

$$\boldsymbol{y}_{radar} = \begin{bmatrix} x \\ y \\ v_r \end{bmatrix}$$

For the camera, the transformed (u, v) pixels directly provide us x, and y. Therefore, the $\boldsymbol{H}_{cam}$ and $\boldsymbol{y}_{cam}$ are given as shown below:

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

$$\boldsymbol{y}_{cam} = \begin{bmatrix} x \\ y \end{bmatrix}$$

This defines the output function h(x_k) and its Jacobian. In the next section, the selection of the tracked target's motion profile.

The Constant Acceleration Model

The Kalman filter uses the state transition matrix F to relate the system state at time $k-1$ to the state at time k. In our project, F models the evolution of the target's states. Based on kinematics, the discrete-time equations for one-dimensional motion are:

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \dot{\boldsymbol{x}}_{k-1}\Delta t + \frac{1}{2}\ddot{\boldsymbol{x}}_{k-1}\Delta t^2$$

$$\dot{\boldsymbol{x}}_k = \dot{\boldsymbol{x}}_{k-1} + \ddot{\boldsymbol{x}}_{k-1}\Delta t$$

$$\ddot{\boldsymbol{x}}_k = \ddot{\boldsymbol{x}}_{k-1}$$

One of the simplest target manoeuvre models based on these kinematic equations is the constant acceleration (CA) model, also referred to as the Wiener-process acceleration (WPA) or nearly constant acceleration (NCA) model [10]. A specific variant, the Wiener-sequence acceleration (WSA) model, has been used in radar-camera data fusion implementations with EKF [9]. As the name implies, the model assumes that the tracked target's acceleration varies gradually. Consequently, the model cannot accurately predict the target's states during rapid changes in acceleration. Compared to simpler motion models, such as constant velocity (CV), the CA model strikes a balance between accuracy and complexity.

The WSA model assumes that acceleration variations behave as white noise [10], meaning the values at any instance are uncorrelated with other instances. Since the acceleration random variable is assumed Gaussian, this uncorrelation implies independence [1], in other words, the acceleration variation is treated as an independent stochastic process.

For 2-dimensional motion, the original model lists the states as: $\boldsymbol{s} = \left(x, y, v_x, v_y, a_x, a_y\right)^T$. However, in our radar-camera project, the measurements are uncoupled to the acceleration states $a_x$and a_y. These states are unobservable even after multiple dynamic steps and are therefore omitted from the model. The resulting F matrix is [9, 10]:

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The process noise covariance matrix represents the noise or the deviation of the target from the modelled motion profile, which is the constant acceleration assumption. The expression for the process noise covariance matrix is shown below [9, 10]; Moreover, the model assumes the process

noise for the pairs of states is independent. Its expression is given as [9, 10]. In the next section we discuss the estimations of the sensors noise variance matrices.

$$\boldsymbol{Q} = var(\boldsymbol{w}) \cdot E(\boldsymbol{w} \cdot \boldsymbol{w}^T)$$

$$\boldsymbol{Q} = \begin{bmatrix} \sigma_{ax}^2 & 0 & 0 & 0 \\ 0 & \sigma_{ay}^2 & 0 & 0 \\ 0 & 0 & \sigma_{ax}^2 & 0 \\ 0 & 0 & 0 & \sigma_{ay}^2 \end{bmatrix} \times \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} \frac{\sigma_{ax}^2 \, \Delta t^4}{4} & 0 & \frac{\sigma_{ax}^2 \, \Delta t^3}{2} & 0 \\ 0 & \frac{\sigma_{ay}^2 \, \Delta t^4}{4} & 0 & \frac{\sigma_{ay}^2 \, \Delta t^3}{2} \\ \frac{\sigma_{ax}^2 \, \Delta t^3}{2} & 0 & \sigma_{ax}^2 \, \Delta t^2 & 0 \\ 0 & \frac{\sigma_{ay}^2 \, \Delta t^3}{2} & 0 & \sigma_{ay}^2 \, \Delta t^2 \end{bmatrix}$$

The R_{cam}, R_{radar} Covariance Matrices Estimations

In this section we present our estimates for the $\boldsymbol{R}_{radar}$ and $\boldsymbol{R}_{cam}$, the autocovariance matrices which define the amount of noise present, and accordingly how confident we should we about the measurement received for a sensor with a given autocovariance matrix, as discussed previously, this decides how much weight is assigned to a given measurement in updating the states estimates.

The autocovariance matrices define the measurement noise present, and accordingly how confident the EKF should be in each sensor. Ideally, $\boldsymbol{R}_{radar}$ and $\boldsymbol{R}_{cam}$ are sensor-dependent obtained through collecting a dataset from the specific sensor for a given scenario, and based on known ground truth, comparing the two sets, the autocovariance matrices can be computed. A simpler yet less accurate approach is to estimate the autocovariances from the general sensor knowledge.

Radar measurement includes noise due to several sources. Source [9] lists the contributions from: (1) multipath effects, (2) processing noise including FFT, (3) inherent measurement noise represented by the received signal SNR, which is modelled by Gaussian distribution. (4) angle-dependent antenna gain. However, noise due to (1), (2), and (4) can be mitigated via the TI IWR 6843 advance processing capabilities; source [9] accounts for the contribution of the source (3) only in the estimation of the autocovariance matric. The variances of phase, range, and Doppler are estimated as one-fourth of the radar's resolutions.

The radar, TI IWR6843 outputs x, y, and vr, measurement instead of range, azimuth, and Doppler. The uncertainty or standard deviation in the measurements x and y (cartesian coordinates) can hence be determined from the uncertainty in the range and azimuth measurements (spherical coordinates) using the propagation of error equation given below [11].

$$\sigma_z^2 = \left(\frac{\partial z}{\partial w}\right)^2 \cdot \sigma_w^2 + \left(\frac{\partial z}{\partial x}\right)^2 \cdot \sigma_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 \cdot \sigma_y^2 + \cdots$$

The standard deviation in the x measurement can be computed as shown below.

$$x = \rho \cos(\theta)$$

$$\sigma_x^2 = \left(\frac{\partial \rho \sin(\theta)}{\partial \rho}\right)^2 \cdot \sigma_\rho^2 + \left(\frac{\partial \rho \cos(\theta)}{\partial \theta}\right)^2 \cdot \sigma_\theta^2$$

$$\sigma_x^2 = (\cos(\theta))^2 \cdot \sigma_\rho^2 - (\rho \sin(\theta))^2 \cdot \sigma_\theta^2$$

$$\sigma_x^2 = (\cos(\theta))^2 \cdot \sigma_\rho^2 - (\rho \sin(\theta))^2 \cdot \sigma_\theta^2$$

For the y measurement, the standard deviation is given as shown below:

$$y = \rho \sin(\theta)$$

$$\sigma_y^2 = (\sin(\theta))^2 \cdot \sigma_\rho^2 + (\rho\cos(\theta))^2 \cdot \sigma_\theta^2$$

The radar autocovariance can be given as shown below. The 0s elements in $\boldsymbol{R}_{radar}$ indicate that the noise present in each measurement (x, y, and vr) are uncorrelated.

$$\boldsymbol{R}_{radar} = \begin{bmatrix} (\cos(\theta))^2 \cdot \left(\frac{\Delta\rho^2}{4}\right) - (\rho\sin(\theta))^2 \cdot \left(\frac{\Delta\theta^2}{4}\right) & 0 & 0 \\ 0 & (\sin(\theta))^2 \cdot \left(\frac{\Delta\rho^2}{4}\right) + (\rho\cos(\theta))^2 \cdot \left(\frac{\Delta\theta^2}{4}\right) & 0 \\ 0 & 0 & \frac{\Delta v_d^2}{4} \end{bmatrix}$$

As for $\boldsymbol{R}_{cam}$, camera measurement noise is assumed uncorrelated and dependent on the reprojection errors from the extrinsic calibration. Let Δd be the diagonal reprojection error; then

$$\Delta u = \Delta v = \frac{\Delta d}{\sqrt{2}}$$

$$\begin{bmatrix} \Delta x \\ \Delta y \\ 1 \end{bmatrix} = \frac{1}{k} \times H_{3\times3}^{-1} \times \begin{bmatrix} \Delta u \\ \Delta v \\ 1 \end{bmatrix}$$

$$\boldsymbol{R}_{cam} = \begin{bmatrix} (\Delta x)^2 & 0 \\ 0 & (\Delta y)^2 \end{bmatrix}$$

In the next section, we present the considerations in the selection between the synchronous and asynchronous fusion architectures and present the rationale behind the selected implementation, based on our application.

Synchronous and Asynchronous Data Fusion

Thus far, we have discussed the mapping matrices H from the states' space to the sensor's measurement space, and the CA model, which informs us about the EKF's **F** and **Q** matrices, as well as discussed, the **R** matrices estimation of the two sensor measurements. By that we fully determined the matrices for our system, leaving us with the how question of how and when to apply the fusion step.

The selection between whether to apply synchronous or asynchronous fusion depends mainly on the specific application and relates to multiple performance factors. For instance, it is common to use asynchronous fusion in applications involving sensor data transmitted wirelessly, due to the system involving multi-rate sensors and usually an inconsistent update rate caused by competitive channel access and packet collisions introducing uncertainties. Applying asynchronous fusion means updating the Kalman filter upon the receipt of a new measurement from any of the sensors.

Moreover, in systems involving sensors with more reliable update rates, a simpler method is to apply synchronous fusion, which involves updating the Kalman filter using the measurements from all sensors simultaneously. While synchronous fusion is appreciably simpler to implement, this comes at a cost, as source [12] suggests that synchronous fusion tends to be less accurate and undoubtedly introduces considerably greater latency compared to asynchronous fusion. This is because synchronous fusion methods involve waiting for all sensor measurements before updating the Kalman filter. In interpolation, for the slower sensor (with fewer measurements per frame), empty slots may be filled by estimations from previous measurements. The R and H matrices and the y vector of each sensor are merged as shown below. Note the following dimensions: $H_k \in \mathbb{R}^{q\times n}$, $y_k \in \mathbb{R}^{q\times 1}$, and $K_k \in \mathbb{R}^{n\times q}$.

$$\boldsymbol{R}_{combined} = \begin{bmatrix} \boldsymbol{R}_{radar} & 0 \\ 0 & \boldsymbol{R}_{cam} \end{bmatrix}$$

$$\boldsymbol{y}_{combined} = \begin{bmatrix} \boldsymbol{y}_{radar} \\ \boldsymbol{y}_{cam} \end{bmatrix}$$

$$\boldsymbol{H}_{combined} = \begin{bmatrix} \boldsymbol{H}_{radar} \\ \boldsymbol{H}_{cam} \end{bmatrix}$$

In asynchronous fusion, the F and Q matrices are time-variant, since they depend on Δt (calculated as |t_current − t_measurement_frame| [13]), which is non-constant for every iteration of the Kalman filter. Therefore, asynchronous fusion is more computationally intensive. Furthermore, it involves careful debugging to keep track of the received measurements in order to avoid logical errors. Specifically, it requires handling out-of-sequence measurements, which occur when the microprocessor receives a measurement from a sensor processing a frame at t_l while the last Kalman filter update corresponded to a frame taken at t_k, with t_l > t_k, meaning that the received measurement is from the past. Handling

out-of-sequence measurements is explained in detail in a later section. In the next section, we discuss the rationale behind the selected fusion architecture, taking into account the YOLO implementation results. In the next section, we discuss the background and selection of the system motion model.

Selected Data Fusion Method

In our application, the implemented YOLO object detection algorithm on the Raspberry Pi achieved an update rate of 70 FPS for a simulation video. However, since the HQ IR-Cut camera used has a 30 FPS output rate, the YOLO output rate is limited by the camera and should achieve an update rate of 30 FPS.

For the radar, the data transmission rate via the UART port can be controlled and adjusted via the configuration file which is sent to the radar module during deployment. The configuration file can be customized using Texas Instruments' mmWave_Demo_Visualizer browser-based application; a screenshot of the application is shown in Figure 3.

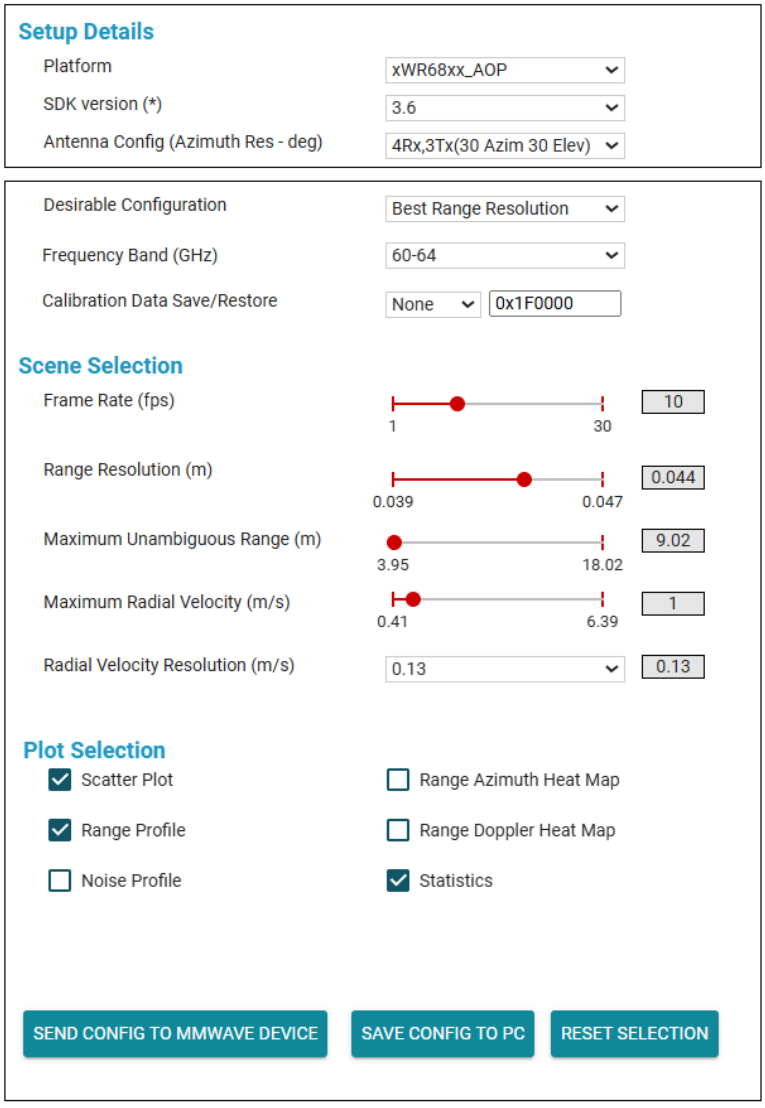


Figure 3. Screenshpt of the mmWave_Demo_Visualizer browser-based application

The figure shows the FPS rate along with other adjustable parameters.These parameters are interrelated, the table in the report in Appendix A summarize these interdependencies based on [14]. For example, in the “best range” scene (as opposed to best velocity or best range-resolution scenes), increasing the maximum unambiguous range coarsens the range resolution.

For the desirable configuration, we opted for the best_range scene. In the best-velocity scene, the maximum possible radial velocity was 3.84 m/2, which is unsatisfactory for pedestrian/car tracking. In the best-range-resolution scene, for a maximum radial velocity of 8 m/s (again, well below car velocities of ~30 m/s), the radial velocity resolution was 1.09 m/s, which is again unsatisfactory, since pedestrians average velocity is about 0.95–1 m/s [15], and under those configurations the radar could detect a single target that actually corresponds to two closely spaced pedestrians moving at similar speeds.

The best_range scene provided a reasonable trade-off for this application. The summary of the chosen configuration file is shown below. Note that the radar FPS was kept at 10 FPS, the higher end of the recommended range. According to [14], the frame period T_f can be expressed as $NT_c + T_{\text{IFT}}$, where T_c is the chirp time or sweep time and T_{IFT} is the inter-frame time.

During T_{IFT} the radar processes and transmits captured data. A higher frame rate reduces T_{IFT}, which may lead to data loss if T_{IFT} is insufficient to transmit the captured data over the relatively low baud rate of the UART transmission protocol; it is therefore suggested to select a relatively low frame rate (3 – 10 fps) [14]. The summary of the radar configuration parameters selected are given below.

% Frequency:60
% Platform:xWR68xx_AOP
% Scene Classifier:best_range
% Azimuth Resolution(deg):30 + 30
% Range Resolution(m):0.293
% Maximum unambiguous Range(m):15
% Maximum Radial Velocity(m/s):19.43
% Radial velocity resolution(m/s):0.31
% Frame Duration(msec):100
% RF calibration data:None
% Range Detection Threshold (dB):15
% Doppler Detection Threshold (dB):15
% Range Peak Grouping:enabled
% Doppler Peak Grouping:enabled
% Static clutter removal:disabled
% Angle of Arrival FoV: Full FoV

Now that the update rates of each sensor are determined, we can determine the best fusion method, synchronous or asynchronous. Since there is a considerable difference between the update rates of YOLO and the radar, we opt for asynchronous fusion. Applying interpolation would introduce considerable uncertainty and increase error, reducing the reliability of the EKF tracker. The second

option of dropping YOLO outputs until the radar is ready would introduce significant latency. The asynchronous fusion logic is shown in Figure 4.

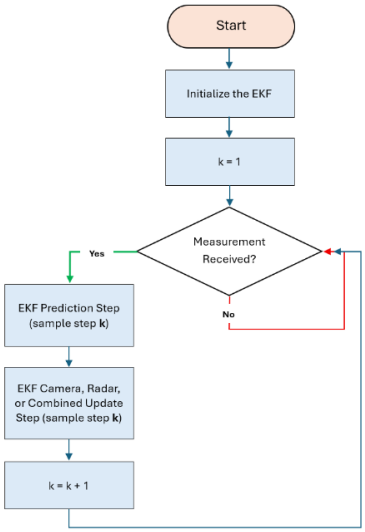


Figure 4. Extended Kalman Filter Asynchronous Logic Flowchart

The EKF is initialized ($k = 0$); the system then waits for a measurement. Once a measurement is received, $\boldsymbol{\Delta t}$ is determined and $F_{k-1}$and $Q_{k-1}$are computed. The EKF performs the prediction step for sample k, then the states are updated using the received measurement for sample k. After that, the sample index k is incremented, and the system waits for the next measurement.

As discussed before, asynchronous fusion faces the possibility of receiving a stale or out-of-sequence measurement; this is when a measurement received corresponds to a frame at time t_l, while the tracker has already updated its states using a frame corresponding to $t = t_k$, with $t_k > t_l$. The handling of Out of Sequence Measurements (OOSM) is discussed in the next section.

Handling Out of Sequence Measurements

According to sources [16], four approaches exist for handling stale measurements that arrive within 0.5 seconds of the last EKF update. Measurements older than this threshold are discarded. Some approaches require computationally heavy corrections unsuitable for real-time applications.

The simplest approach is to discard stale measurements [16], but this risks losing valuable information. Therefore, we adopt the second-simplest method: which is to retrodict back to the time of the frame that measurement received corresponds to [16].

The retrodicted a priori state is:

$$\boldsymbol{x_l^- = F_{k-1}^{-1} \cdot x_k^-}$$

The priori estimation error covariance matrix remains referenced to time t_k, as given below, yet the process noise covariance matrix, $\boldsymbol{Q}_{k-1}$, is assumed to be zero [16].

$$\boldsymbol{P_k^-} = \boldsymbol{F_{k-1}^{-1}} \boldsymbol{P_{k-1}^+} (\boldsymbol{F_{k-1}^{-1}})^T$$

In the update step, the posteriori state is determined as given below. The Kalman gain and the posteriori estimation error autocovariance matrix remain unchanged and referenced to time t_k [16].

$$\boldsymbol{x_k^+} = \boldsymbol{x_k^-} + \boldsymbol{K_k}(\boldsymbol{y_l} - \boldsymbol{H_k x_l^-})$$

$$\boldsymbol{K_k} = \boldsymbol{P_k^- H_k^T} \cdot (\boldsymbol{H_k P_k^- H_k^T} + \boldsymbol{R_k})^{-1}$$

$$\boldsymbol{P_k^+} = (\boldsymbol{I} - \boldsymbol{K_k H_k}) \boldsymbol{P_k^-} (\boldsymbol{I} - \boldsymbol{K_k H_k})^T + \boldsymbol{K_k R_k K_k^T}$$

While this is relatively simple, it falls short in its low accuracy, as it reduces the mathematical complexity of the equation when $\boldsymbol{Q}_{k-1}$ is not zero, which is not an accurate assumption. Nonetheless, it avoids completely discarding a stale measurement which can provide new valuable information. In the next section, we discuss the application of the EKF in radar-camera fusion system.

Special Case

In this section, we discuss how a special case of received measurements is handled. Whenever a camera and the radar capture a measurement of nearly the same frame and arrive within a 0.05 sec delay window (representing capture and processing latency), the EKF update is performed jointly, updating a track's state estimates using the combined camera and radar measurements. This case is flagged as case 3; radar-alone update is flagged as case 2, while camera-alone update is flagged as case 1.

In the case 3 flag, the measurement prediction step is performed assuming both camera and radar measurements (using $H_{\text{cam}}$and H_{radar}). In the GNN step of constructing the cost function, both the radar and camera cost functions are constructed independently. After that, the Hungarian algorithm is applied to both cost functions. For tracks which get assigned both a camera measurement and a radar measurement, their assignment count increments, and the assigned measurement is saved as shown below.

$$y_{assigned} = \begin{bmatrix} y_{assigned,radar} \\ y_{assigned,cam} \end{bmatrix}$$

For tracks assigned only either a camera or a radar measurement (meaning a sensor sees a measurement that the other sensor does not detect), their deletion count increments, and the EKF update step is skipped for these tracks. Note that the H, R, and $y_{\text{predicted}}$ matrices and vectors are formulated in the same way as for synchronous fusion, this was discussed in the Synchronous and Asynchronous Data Fusion section. In the next section, we discuss the data association algorithm and target tracking.

Data Association and Target Tracking

Data association and target tracking is used to make sense of the measurements received from sensors, as to associate new measurements with the series of past detections of a given target or targets. In radar sensor target detection, it is quite common for radar data more so than the camera to frequently produce false alarms or clutter. This nudges us to go for using data association methods, target tracking, and track management to accurately differentiate true targets measurements from false alarms and clutter.

Overview of common data association methods

Data association methods aim to assign n observations to k tracks in a way that optimizes a given criterion. Some approaches are deterministic, optimizing a probabilistic likelihood, such as the Suboptimal Nearest Neighbour (SNN) or Global Nearest Neighbour (GNN). The SNN independently assigns each measurement to a track, which can sometimes result in conflicts where two tracks are assigned the same measurement. GNN resolves this by considering all assignments globally, eliminating such conflicts.

Other methods, such as Probabilistic Data Association (PDA), consider all possible associations of a measurement to a single target [16]. For n measurements, PDA generates $n + 1$ hypotheses: H_0 assumes none of the measurements belongs to target k, H_1 assumes measurement 1 belongs to target k while the rest are clutter, and so on [16].

While PDA is more robust than SNN or GNN because it handles assignment probabilistically, it is limited to single-target tracking. In multi-target scenarios, conflicts can still occur. This limitation is addressed by Joint Probabilistic Data Association (JPDA), which extends PDA by considering all combinations of track-to-measurement assignments jointly. The number of hypotheses increases combinatorially as n choose k [16, 17].

In general, the accuracy of methods increases from SNN → GNN, and PDA → JPDA. However, there is a trade-off between accuracy and computational demand. Considering the computational limitations of the Raspberry Pi in our project, we selected GNN, which provides acceptable performance even in fairly complex scenarios.

The GNN and the Target Tracking Block Diagram

The Kalman filter is a primary component in target tracking. Figure 5 shows a block diagram of the logic and steps used in implementing target tracking. The flow chart is constructed based on [9, 16, 18].

Since asynchronous fusion is used, the pipeline is triggered upon the arrival of a new measurement from a sensor. The measurement is processed, and the prediction for sample step k is performed (sample step **does not** refer to the clock timestamp of the instant at which the prediction or update was performed).

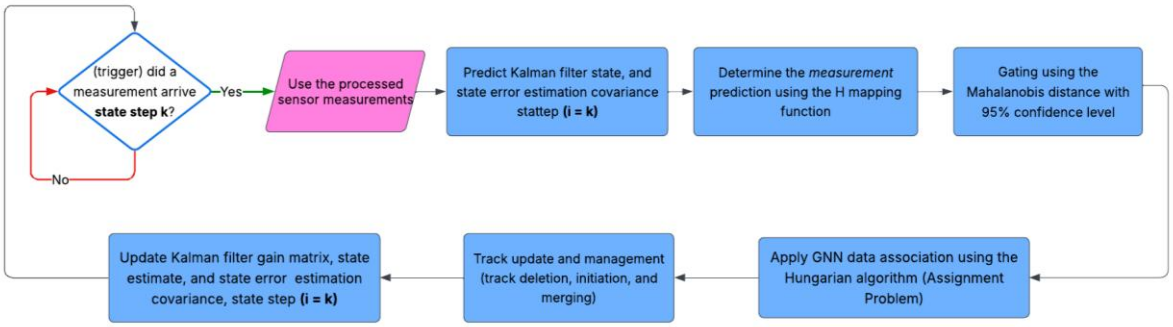


Figure 5. Target Tracking and Data Association Block Diagram.

The state prediction for each target is then used to produce the measurement prediction for each target/track. The Jacobian of the function $h(x_k)$, H, is used to map the state predictions to the measurement prediction space. Gating is then performed to eliminate outliers for a given track, thereby reducing computations required in the association steps [16, 17]. The data association method is then applied, followed by the track management logic, including initiation and deletion of tracks. After that, the Kalman filter update step is performed for sample step k.

In the following sections, we explain the gating step, the data association method selected (GNN), and the track management logic layer.

Gating

The initial step of DA and target tracking is the gating step. As mentioned, it is used to reduce how many measurements-to-track associations are considered [16, 17]. The gating is based on the Mahalanobis distance equation given below; the equation is dependent on the innovation term. The greater the innovation term, the greater the difference between a given received measurement and the predicted measurement for a given track, indicating a smaller likelihood that that measurement should be associated with that track.

$$d_{ij}^2 = (y_j - H\hat{x}_i^-)^T \cdot S^{-1} \cdot (y_j - H\hat{x}_i^-)$$

The equation is also dependent on the inverse of the autocovariance matrix of the innovation term; the smaller the variance, the lower noise is present and hence the more confident we are about the received measurement, and hence the more weight it is given in the sum of the squares of the distance for a given n – dimensional multivariant RV y [16].

A measurement is said to be an outlier with 95% confidence based on the equation given below [16, 19]. We obtain the value of G from the chi-squared distribution with n degrees of freedom [16, 19]. In our case, we are receiving 3 measurements from the radar (x, y, and v_r), and hence $n = 3$, G is approximately 7.815. For the camera with $n = 2$, G is approximately 5.991.

$$(y_j - H\hat{x}_i^-)^T \cdot S^{-1} \cdot \left(y_j - H\hat{x}_i^-\right) < G \approx 7.815 \; for\; Radar$$

$$(y_j - H\hat{x}_i^-)^T \cdot S^{-1} \cdot \left(y_j - H\hat{x}_i^-\right) < G \approx 5.991 \; for\; camera$$

Global Nearest Neighbour (GNN)

In GNN, we choose the data association hypothesis Ω^k which has the highest probability [16]. We construct the probability of a data association algorithm as follows. The probability distribution of an innovation term is given by g below. The probability that a given measurement y_j should be associated with a prediction $H\hat{x}_i^-$ decreases as the difference increases, and the more spread the innovation probability distribution is.

$$g = \frac{1}{\sqrt{(2\pi)^n |C|}} e^{-\frac{1}{2}(y_j - H\hat{x}_i^-)^T \cdot S^{-1} \cdot \left(y_j - H\hat{x}_i^-\right)}$$

The probability of the occurrence of independent events is equal to the multiplication of the probabilities of the individual events; hence, a data association hypothesis probability is given as shown below, $f\left(\Omega^k\right)$. This reads as: the probability that measurement y_1 is associated with prediction $H\hat{x}_1^-$ ***and*** measurement y_2 is associated with prediction $H\hat{x}_2^-$, and so on [16].

$$f\left(\Omega^k\right) = \prod_{(i,j) \in \varphi_h} \frac{1}{\sqrt{(2\pi)^n |C|}} e^{-\frac{1}{2} d_{ij}^2}$$

A question arises: how can we find the data association hypothesis that optimizes the probability expression given above? The expression given above is the maximum likelihood estimation (MLE) expression. If we take the natural log on both sides, we obtain the following expression:

$$ln\left[f\left(\Omega^k\right)\right] = -\frac{nj}{2}\ln[2\pi] - \frac{j}{2}\ln[C] - \frac{1}{2}\sum_{(i,j) \in \varphi_h} d_{ij}^2$$

To maximize the above expression, is to minimize $\mathrm{j}\ln[C] + \sum_{(i,j) \in \varphi_h} d_{ij}^2$. According to [19], we obtain the same results if we only minimize $\sum_{(i,j) \in \varphi_h} d_{ij}^2$; which is to select the data association hypothesis resulting in the smallest possible sum of squared distances of the n-dimensional measurement to a given track's prediction position.

But how to practically determine the associations that result in the maximum likelihood probability? To answer this, let us consider two scenarios, in the first assume we have three targets, each produces their own ellipsoidal gates of uncertainty. The targets are relatively far apart and there is no overlap in their respective gates. In this case, solving the optimal association problem is relatively simple. Track j is matched to closest observation i in its ellipsoid, the rest of the observations are considered to be

potential detected targets or clutter. A more complex scenario is given in Figure 6, adapted based on [16, 17, and 19]

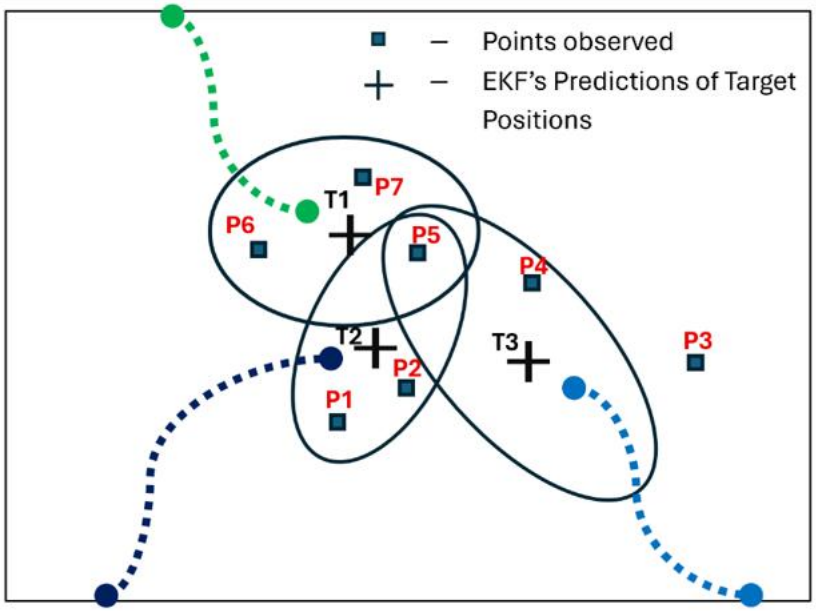


Figure 6. Target Tracking and Data Association Problem Scenario

One way is to list all possible associations, which would be equal to n (measurements) – choose – k (tracks), prior to gating. However, after gating, we roughly only need to consider the sum of n measurement per gate (excluding cases of conflicts, multiple measurements assigned to the same track); and measurements outside the gates are considered potential new targets or false alarms.

Practically this corresponds to constructing a matrix with the tracks as rows and the measurements are the columns. The matrix is filled using the logic given below (based on [19]), and the table we obtain for the scenario given in Figure 6 is given in Table 1.

For each track T_k *of the set of tracks*:
    If P_i *is within the gate of* T_k:
        Compute d_{ij}^2 *as the squared distance from predicted position of* T_k *to* P_i
    Else:
        set d_{ij}^2 *to* 100

Table 1. Cost Matrix for Figure 6 Scenario

	P1	P2	P3	P4	P5	P6	P7
T1	100	100	100	100	d_{51}^2	d_{61}^2	d_{71}^2
T2	d_{12}^2	d_{22}^2	100	100	d_{52}^2	100	100
T3	100	100	100	d_{43}^2	d_{53}^2	100	100

We obtain what is referred to as the cost function, and the problem is to determine the global optimal assignment such that the sum of the corresponding cost function elements is minimal. This is solved using the Hungarian algorithm, discussed in the following section.

The Assignment Problem

The assignment is formulated as given below [16]. The condition of the M matrix makes it a permutation matrix, resulting in only a non-zero element (one) per column and row, hence resulting in a single assignment for each track, and avoiding the occurrence of conflicts. The Hungarian algorithm allows us to obtain the assignment resulting in the smallest possible total cost without needing to consider all possible combinations. The Hungarian algorithm solves the assignment problem in a polynomial time $O(n^3)$ [20].

$$Total\ Cost = \sum_{i=1}^{N}\sum_{j=1}^{N} M_{ij}\, C_{ij}$$

$$\forall i \sum_{j=1}^{N} M_{ij} = 1 \quad and\ \forall j \sum_{i=1}^{N} M_{ij} = 1$$

$$M_{ij} \in \{0, 1\}$$

Based on [16], the Hungarian algorithm steps are as given below.

1. Pad extra rows or columns with rows to construct an N x N square matrix. In the example given in Table 3, we would add four more rows with zeros
2. Subtract the minimal element from each column
3. Subtract the minimal element from each row
4. Draw the minimalist possible number of column and rows to cover the zeros, if they are N lines, then we created our permutation matrix, otherwise, we proceed to step 5.
5. Determine the minimum number from the elements that are not crossed by the lines. Subtract this number from the group of elements that are not crossed by the lines and add this element to the elements at the intersection of the drawn lines.
6. Check again if we created a permutation matrix, otherwise, go back to step 4.

According to [19], GNN may assign a track with an empty gate a measurement that is outside of its gate. We handle this limitation using track management logic, where tracks with empty gates are flagged as partially inactive and their rows is deleted from the constructed cost function before the Hungarian is applied. Track management is discussed in the upcoming section.

Track Management

Track management involves a logic layer that handles track deletion and initiation [18]. After the GNN data association step, measurements left unassigned are initiated as potential new tracks or false alarms, and whenever a track is not assigned any measurement (empty gate) for 30 consecutive frames, the track is deleted [18].

We use a structure of Tracks; the detection field is incremented every time a track is not assigned any measurement and is reset to zero every time the track is assigned a measurement. The assignment field is incremented every time a given track is assigned a measurement. When the assignment exceeds 30, the track is said to be established and is displayed as a confirmed track.

The block diagram given in Figure 36 provides a comprehensive description of how tracks are managed for all three cases: camera-alone, radar-alone, and combined EKF update steps. In the next section we summarize the full fusion and target tracking pipeline.

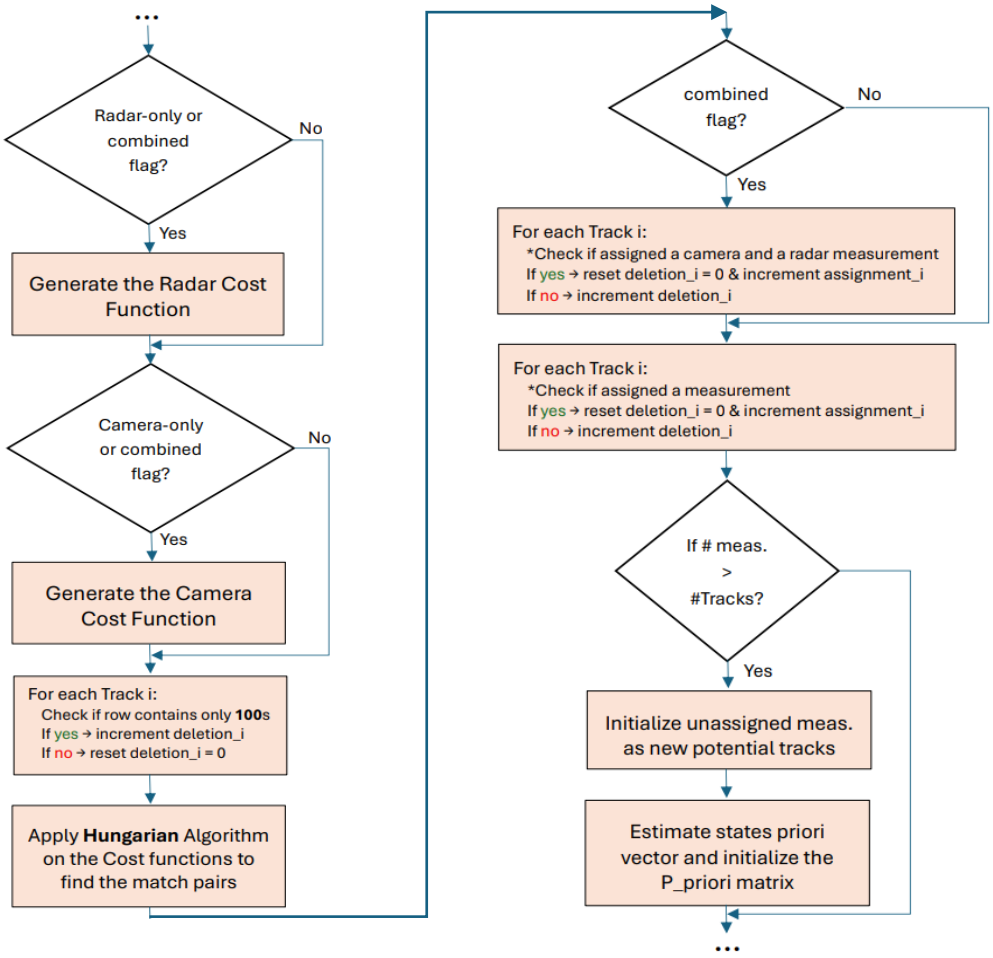


Figure 36. Target tracking logic flowchart.

Full Fusion and Tracking Pipeline Summary

This project implements an offline Radar–Camera data fusion system using the EKF for target tracking. Fusion is executed asynchronously, with the pipeline triggered by the arrival of new measurements. This required handling out-of-sequence measurements, as described previously.

The EKF update combines camera and radar measurements (similar to synchronous fusion) in cases where both sensors capture a measurement corresponding to the same frame and arrive within a small time window of 0.05 s. The EKF is integrated within the GNN data association framework used to optimize probabilistic measurement-to-track assignments. Moreover, for camera measurements, the integrated Random Forest is intended to classify frames as dense fog or no fog based on extracted features and to scale the camera noise covariance matrix accordingly.

Track management includes deletion of tracks not updated for 30 consecutive frames and initialization of new tracks for unassigned measurements. Tracks with more than 30 assigned measurements are displayed as established trac

References

[1]. D. Simon, *Optimal state estimation*. Hoboken, N.J.: Wiley-Interscience, 2006.

[2]. Al-khwarizmi (الخوارزمی), "Sensor Fusion: Extended Kalman Filter - Autonomous Car Motion Estimation," *YouTube*, Mar. 19, 2023.

[3]. Steve Brunton, "The Kalman Filter [Control Bootcamp]," *YouTube*, Feb. 06, 2017. https://www.youtube.com/watch?v=s_9InuQAx-g&list=PLMrJAkhIeNNR20Mz-VpzgfQs5zrYi085m&index=18 (accessed Aug. 26, 2025).

[4]. "Lecture 24: Observability and Constructibility 7 Observability and Constructibility." Accessed: Aug. 26, 2025. [Online]. Available: https://cim.mcgill.ca/~boulet/304-501A/L24.pdf

[5]. Steve Brunton, "Control Bootcamp: Observability," *YouTube*, Feb. 06, 2017. https://www.youtube.com/watch?v=iRZmJBcg1ZA&list=PLMrJAkhIeNNR20Mz-VpzgfQs5zrYi085m&index=16 (accessed Aug. 26, 2025).

[6]. T. Michael and Heath, "Chapter 3 -Linear Least Squares." Accessed: Aug. 26, 2025. [Online]. Available: https://heath.cs.illinois.edu/scicomp/notes/chap03_8up.pdf

[7]. B. P. Lathi and Z. Ding, *Modern digital and analog communication systems*. New York: Oxford University Press, 2019.

[8]. B. Douglas, "#2: The Kalman Filter," *Engineering Media*, Feb. 25, 2021. https://engineeringmedia.com/controlblog/the-kalman-filter

[9]. R. Zhang and S. Cao, "Extending Reliability of mmWave Radar Tracking and Detection via Fusion With Camera," *IEEE Access*, vol. 7, pp. 137065–137079, 2019, doi: https://doi.org/10.1109/access.2019.2942382.

[10]. X. Li and V. P. Jilkov, "Survey of maneuvering target tracking: dynamic models," Jul. 2000, doi: https://doi.org/10.1117/12.391979.

[11]. V. Lindberg, "Uncertainties and Error Propagation," *www.geol.lsu.edu*, Jul. 01, 2000. http://www.geol.lsu.edu/jlorenzo/geophysics/uncertainties/Uncertaintiespart2.html

[12]. K. Zhang, Z. Wang, L. Guo, Y. Peng, and Z. Zheng, "An Asynchronous Data Fusion Algorithm for Target Detection Based on Multi-Sensor Networks," *IEEE Access*, vol. 8, pp. 59511–59523, 2020, doi: https://doi.org/10.1109/access.2020.2982682.

[13]. Kalman, “Signal Processing Stack Exchange,” *Signal Processing Stack Exchange*, Sep. 04, 2019. https://dsp.stackexchange.com/questions/60511/kalman-filter-how-to-combine-data-from-sensors-with-different-measurement-rate/60513#60513 (accessed Aug. 26, 2025).

[14]. “mmWave Demo Visualizer User’s Guide mmWave Demo Visualizer,” 2017. Accessed: Mar. 20, 2025. [Online]. Available: https://www.ti.com/lit/ug/swru529c/swru529c.pdf?ts=1742455933180&ref_url=https%253A%252F%252Fwww.google.com%252F

[15]. M. F. Mohamad Ali, M. S. Abustan, S. H. Abu Talib, I. Abustan, N. Abd Rahman, and H. Gotoh, “A Case Study on the Walking Speed of Pedestrian at the Bus Terminal Area,” *E3S Web of Conferences*, vol. 34, p. 01023, 2018, doi: https://doi.org/10.1051/e3sconf/20183401023.

[16]. D. Waard, “UvA-DARE (Digital Academic Repository) A new approach to distributed data fusion,” 2008. Accessed: Aug. 26, 2025. [Online]. Available: https://pure.uva.nl/ws/files/4318786/59162_07.pdf

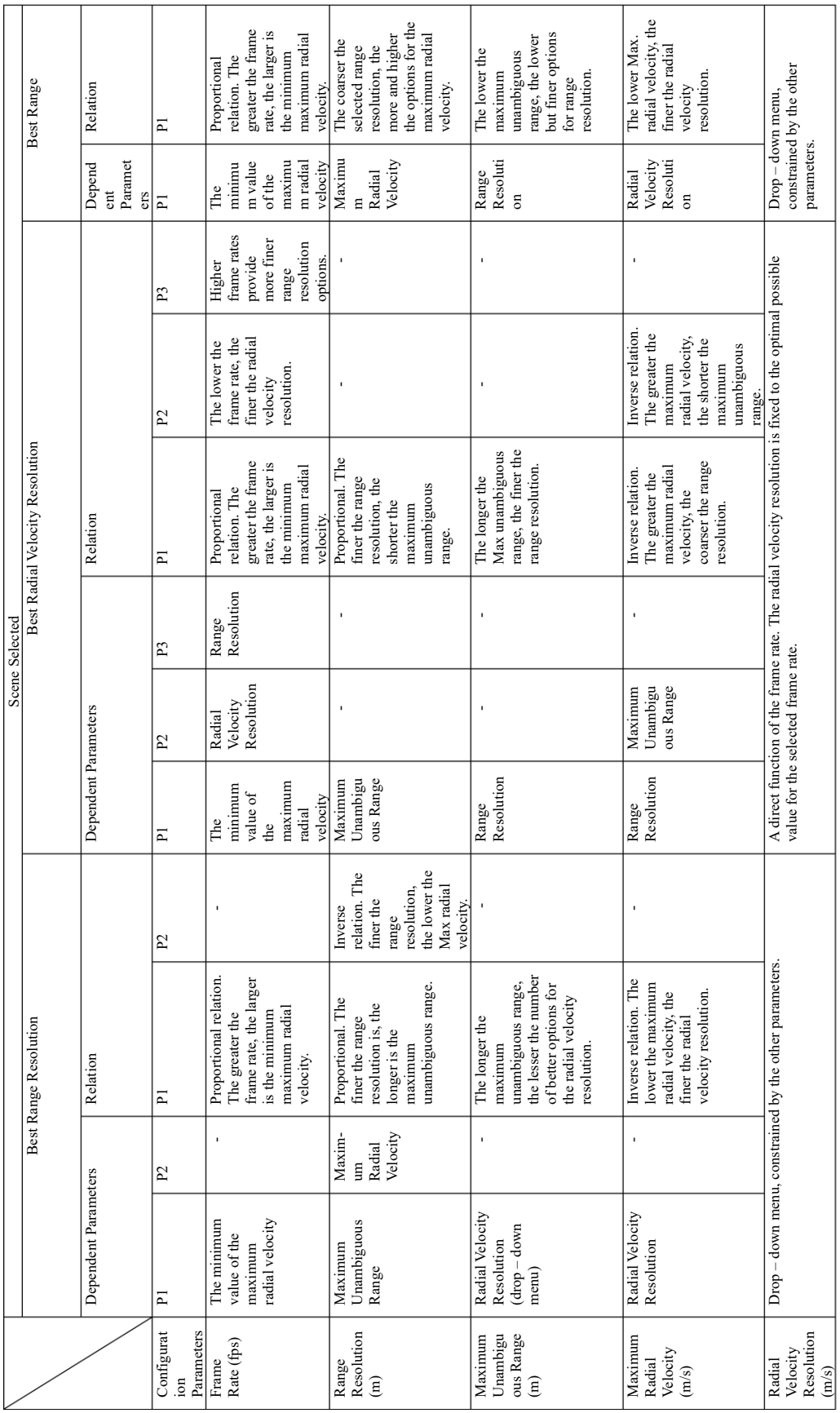
[17]. B. Collins, “Introduction to Data Association,” 2012. https://www.cse.psu.edu/~rtc12/CSE598C/datassocPart1.pdf (accessed Aug. 26, 2025).

[18]. K. Aziz, Eddy De Greef, Maxim Rykunov, A. Bourdoux, and H. Sahli, “Radar-camera Fusion for Road Target Classification,” Sep. 2020, doi: https://doi.org/10.1109/radarconf2043947.2020.9266510.

[19]. Pavlina Konstantinova, A. Udvarev, and Tzvetan Semerdjiev, “A study of a target tracking algorithm using global nearest neighbor approach,” Jan. 2003, doi: https://doi.org/10.1145/973620.973668.

[20]. Wikipedia Contributors, “Hungarian algorithm,” *Wikipedia*, Aug. 27, 2019. https://en.wikipedia.org/wiki/Hungarian_algorithm

Appendix A


Scene Selected
Best Range Resolution
Best Radial Velocity Resolution
Best Range
Dependent Parameters
Relation
Dependent Parameters
Relation
Dependent Parameters
Relation
Configuration Parameters
P1
P2
P1
P2
P1
P2
P3
P1
P2
P3
P1
P1
Frame Rate (fps)
The minimum value of the maximum radial velocity
-
Proportional relation. The greater the frame rate, the larger is the minimum maximum radial velocity.
-
The minimum value of the maximum radial velocity
Radial Velocity Resolution
Range Resolution
Proportional relation. The greater the frame rate, the larger is the minimum maximum radial velocity.
The lower the frame rate, the finer the radial velocity resolution.
Higher frame rates provide more finer range resolution options.
The minimum value of the maximum radial velocity
Proportional relation. The greater the frame rate, the larger is the minimum maximum radial velocity.
Range Resolution (m)
Maximum Unambiguous Range
Maxim-um Radial Velocity
Proportional. The finer the range resolution is, the longer is the maximum unambiguous range.
Inverse relation. The finer the range resolution, the lower the Max radial velocity.
Maximum Unambiguous Range
-
-
Proportional. The finer the range resolution, the shorter the maximum unambiguous range.
-
-
Maximum Radial Velocity
The coarser the selected range resolution, the more and higher the options for the maximum radial velocity.
Maximum Unambiguous Range (m)
Radial Velocity Resolution (drop – down menu)
-
The longer the maximum unambiguous range, the lesser the number of better options for the radial velocity resolution.
-
Range Resolution
-
-
The longer the Max unambiguous range, the finer the range resolution.
-
-
Range Resolution
The lower the maximum unambiguous range, the lower but finer options for range resolution.
Maximum Radial Velocity (m/s)
Radial Velocity Resolution
-
Inverse relation. The lower the maximum radial velocity, the finer the radial velocity resolution.
-
Range Resolution
Maximum Unambiguous Range
-
Inverse relation. The greater the maximum radial velocity, the coarser the range resolution.
Inverse relation. The greater the maximum radial velocity, the shorter the maximum unambiguous range.
-
Radial Velocity Resolution
The lower Max. radial velocity, the finer the radial velocity resolution.
Radial Velocity Resolution (m/s)
Drop – down menu, constrained by the other parameters.
A direct function of the frame rate. The radial velocity resolution is fixed to the optimal possible value for the selected frame rate.
Drop – down menu, constrained by the other parameters.