Sensor Fusion

Techniques & Algorithms

By : Yash Vardhan Raizada
Mentor : Prof. Pranamesh Chakraborty

Getting Started...

Sensor Fusion is combining of sensory 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.

In the following slides we will look at :

- What is Sensor Fusion ?
- Categorization of Sensor Fusion Methods
- Difference between Centralized & Decentralized methods of processing
- Different Levels of Sensor Fusion
- Different Algorithms being used nowadays
- Algorithms using Central Limit Theorem & Kalman Filters
- Applications & an example implementation in Self Driving Cars

What is Sensor Fusion ?

Sensor fusion is combining of sensory 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. The term uncertainty reduction in this case can mean more accurate, more complete, or more dependable, or refer to the result of an emerging view, such as stereoscopic vision (calculation of depth information by combining two-dimensional images from two cameras at slightly different viewpoints).

The data sources for a fusion process are not specified to originate from identical sensors. One can distinguish direct fusion, indirect fusion and fusion of the outputs of the former two. Direct fusion is the fusion of sensor data from a set of heterogeneous or homogeneous sensors, soft sensors, and history values of sensor data, while indirect fusion uses information sources like a priori knowledge about the environment and human input.

Sensor fusion is also known as (multi-sensor) data fusion and is a subset of information fusion.

Methods & Categorization

The three fundamental ways of combining sensor data are the following:

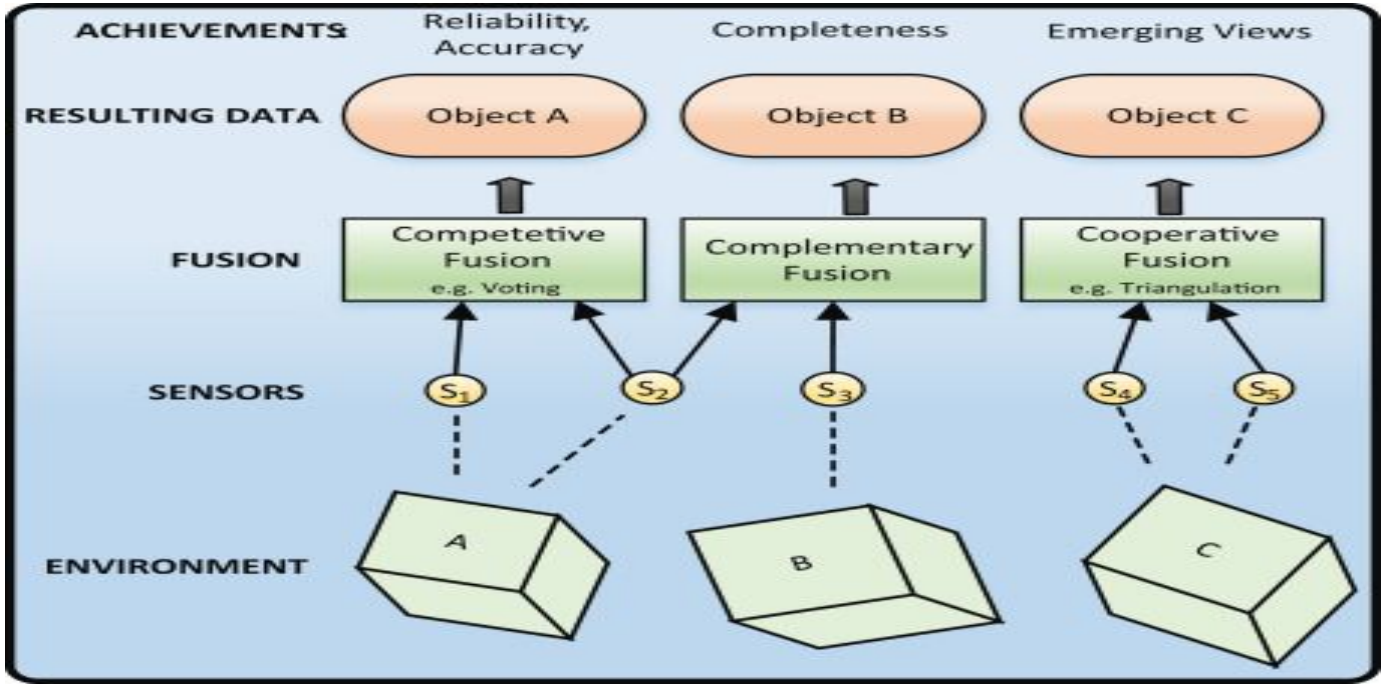
- Redundant sensors: All sensors give the same information for the world.
- Complementary sensors: The sensors provide independent (disjoint) types of information about the world.
- Coordinated sensors: The sensors collect information about the world sequentially.

The three basic sensor communication schemes are:

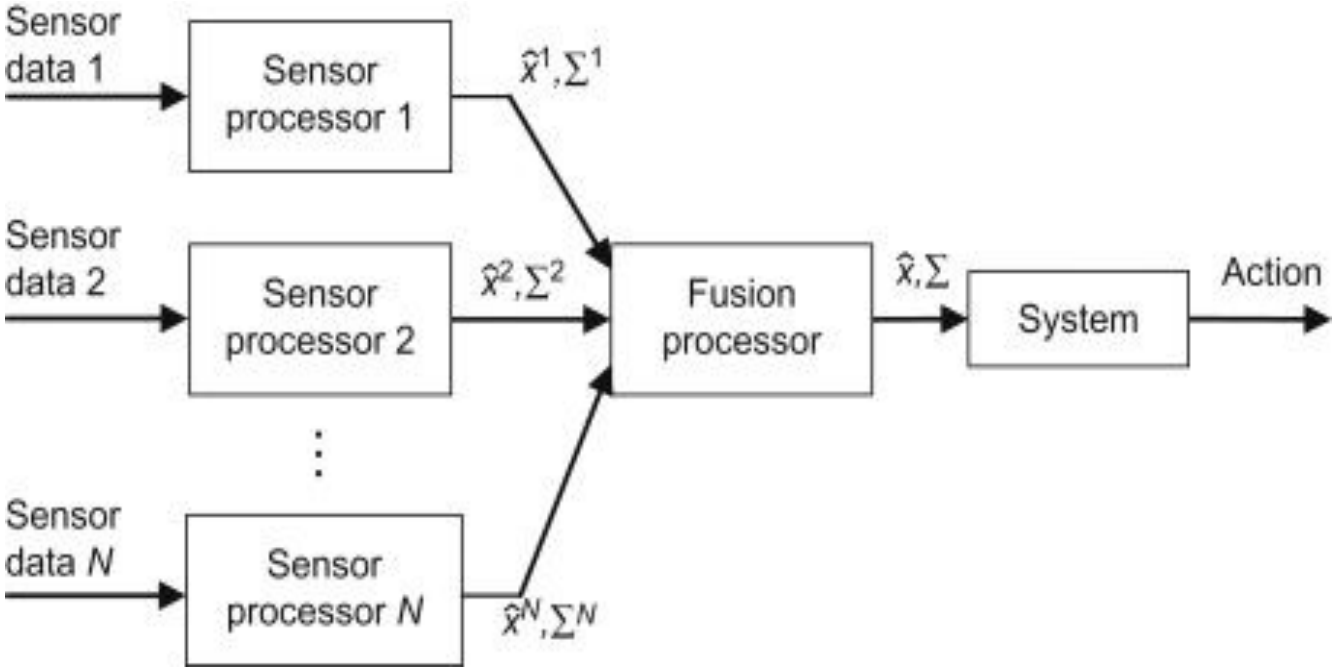
- Decentralized: No communication exists between the sensor nodes.
- Centralized: All sensors provide measurements to a central node.
- Distributed: The nodes interchange information at a given communication rate (e.g., every five scans, i.e., one-fifth communication rate).

The centralized scheme can be regarded as a special case of the distributed scheme where the sensors communicate to each other every scan.

Methods & Categorization



Methods & Categorization



Methods & Categorization

Another classification of sensor configuration refers to the coordination of information flow between sensors. These mechanisms provide a way to resolve conflicts or disagreements and to allow the development of dynamic sensing strategies.

- Sensors are in **redundant (or competitive) configuration** if each node delivers independent measures of the same properties. This configuration can be used in error correction when comparing information from multiple nodes. Redundant strategies are often used with high level fusions in voting procedures.
- **Complementary configuration** occurs when multiple information sources supply different information about the same features. This strategy is used for fusing information at raw data level within decision-making algorithms. Complementary features are typically applied in motion recognition tasks with Neural network, Hidden Markov model, Support-vector machine, clustering methods and other techniques.
- **Cooperative sensor fusion** uses the information extracted by multiple independent sensors to provide information that would not be available from single sensors. For example, sensors connected to body segments are used for the detection of the angle between them. Cooperative sensor strategy gives information impossible to obtain from single nodes. Cooperative information fusion can be used in motion recognition, gait analysis, motion analysis.

Centralized vs. Decentralized

In sensor fusion, centralized versus decentralized refers to where the fusion of the data occurs.

- In **centralized fusion**, the clients simply forward all of the data to a central location, and some entity at the central location is responsible for correlating and fusing the data.
- In **decentralized**, the clients take full responsibility for fusing the data. "In this case, every sensor or platform can be viewed as an intelligent asset having some degree of autonomy in decision-making."

Multiple combinations of centralized and decentralized systems exist.

Levels of Sensor Fusion

There are several categories or levels of sensor fusion that are commonly used.

Level 0 – Data alignment

Level 1 – Entity assessment (e.g. signal/feature/object).

Tracking and object detection/recognition/identification

Level 2 – Situation assessment

Level 3 – Impact assessment

Level 4 – Process refinement (i.e. sensor management)

Level 5 – User refinement

Levels of Sensor Fusion

Sensor fusion level can also be defined basing on the kind of information used to feed the fusion algorithm. More precisely, sensor fusion can be performed fusing raw data coming from different sources, extrapolated features or even decision made by single nodes.

- Data level - data level (or early) fusion aims to fuse raw data from multiple sources and represent the fusion technique at the lowest level of abstraction. It is the most common sensor fusion technique in many fields of application.
- Data level fusion algorithms usually aim to combine multiple homogeneous sources of sensory data to achieve more accurate and synthetic readings. When portable devices are employed data compression represent an important factor, since collecting raw information from multiple sources generates huge information spaces that could define an issue in terms of memory or communication bandwidth for portable systems.
- Data level information fusion tends to generate big input spaces, that slow down the decision-making procedure. Also, data level fusion often cannot handle incomplete measurements. If one sensor modality becomes useless due to malfunctions, breakdown or other reasons the whole systems could occur in ambiguous outcomes.

Levels of Sensor Fusion

- Feature level - features represent information computed onboard by each sensing node. These features are then sent to a fusion node to feed the fusion algorithm. This procedure generates smaller information spaces with respect to the data level fusion, and this is better in terms of computational load. Obviously, it is important to properly select features on which to define classification procedures: choosing the most efficient features set should be a main aspect in method design. Using features selection algorithms that properly detect correlated features and features subsets improves the recognition accuracy but large training sets are usually required to find the most significant feature subset.

Levels of Sensor Fusion

- Decision level - decision level (or late) fusion is the procedure of selecting an hypothesis from a set of hypotheses generated by individual (usually weaker) decisions of multiple nodes. It is the highest level of abstraction and uses the information that has been already elaborated through preliminary data- or feature level processing. The main goal in decision fusion is to use meta-level classifier while data from nodes are preprocessed by extracting features from them. Typically decision level sensor fusion is used in classification an recognition activities and the two most common approaches are majority voting and Naive-Bayes. Advantages coming from decision level fusion include communication bandwidth and improved decision accuracy. It also allows the combination of heterogeneous sensors.

Algorithms

Sensor fusion is a term that covers a number of methods and algorithms, including:

- Central limit theorem
- Kalman filter
- Bayesian networks
- Dempster-Shafer
- Convolutional neural network

Algorithms

Two example sensor fusion calculations are illustrated below.

Let x1 and x2 denote two sensor measurements with noise variances $\sigma 1^2$ and $\sigma 2^2$, respectively. One way of obtaining a combined measurement x3 is to apply the **Central Limit Theorem**, which is also employed within the Fraser-Potter fixed-interval smoother, namely :

$$\mathbf{x}_3 = \sigma_3^2(\sigma_1^{-2}\mathbf{x}_1 + \sigma_2^{-2}\mathbf{x}_2)$$

where $\sigma_3^2=(\sigma_1^{-2}+\sigma_2^{-2})^{-1}$ is the variance of the combined estimate. It can be seen that the fused result is simply a linear combination of the two measurements weighted by their respective noise variances.

Algorithms

Another method to fuse two measurements is to use the optimal **Kalman filter**. Suppose that the data is generated by a first-order system and let **P(k)** denote the solution of the filter's Riccati equation. By applying Cramer's rule within the gain calculation it can be found that the filter gain is given by:

$$\mathbf{L}_k = \begin{bmatrix} \frac{\sigma_2^2 \mathbf{P}_k}{\sigma_2^2 \mathbf{P}_k + \sigma_1^2 \mathbf{P}_k + \sigma_1^2 \sigma_2^2} & \frac{\sigma_1^2 \mathbf{P}_k}{\sigma_2^2 \mathbf{P}_k + \sigma_1^2 \mathbf{P}_k + \sigma_1^2 \sigma_2^2} \end{bmatrix}.$$

By inspection, when the first measurement is noise free, the filter ignores the second measurement and vice versa. That is, the combined estimate is weighted by the quality of the measurements.

For Detailed Calculations Visit : https://www.sciencedirect.com/topics/engineering/sensor-fusion

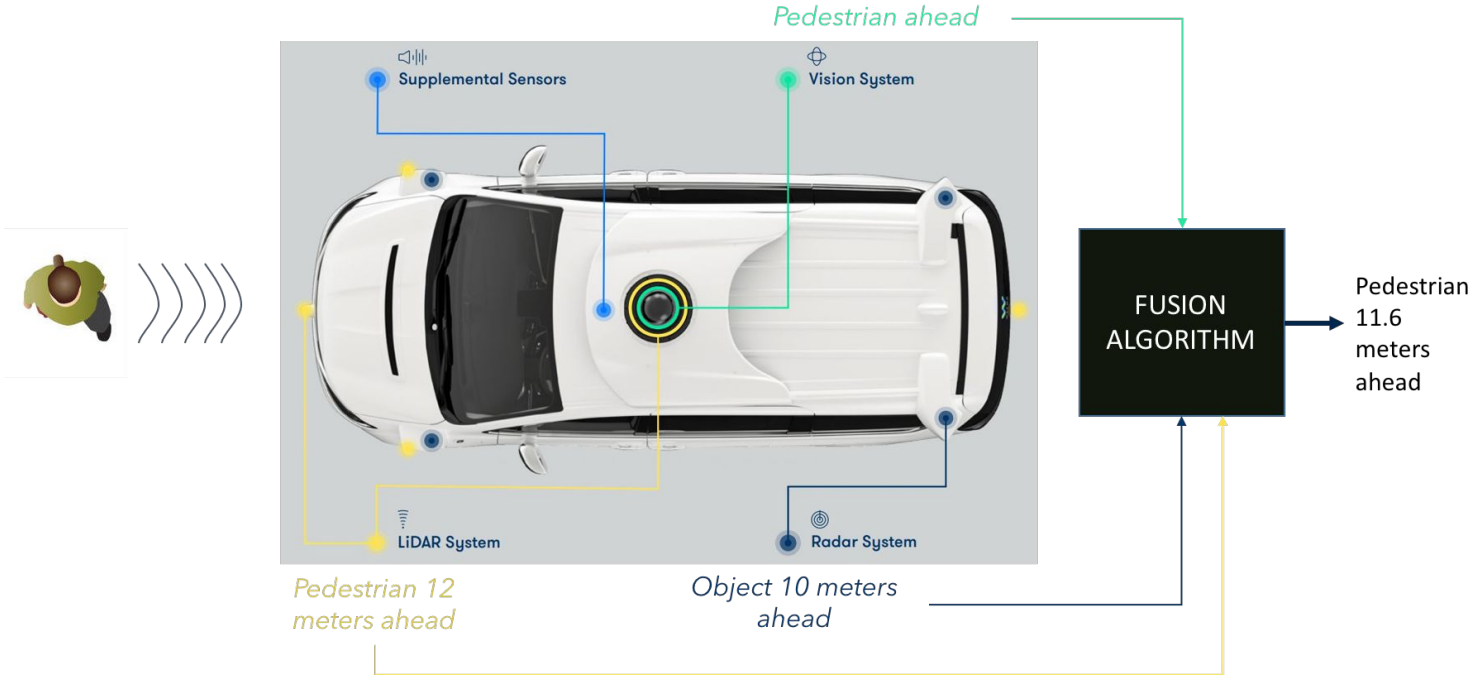
Applications

One application of sensor fusion is GPS/INS, where Global Positioning System and inertial navigation system data is fused using various different methods, e.g. the extended Kalman filter. This is useful, for example, in determining the altitude of an aircraft using low-cost sensors.

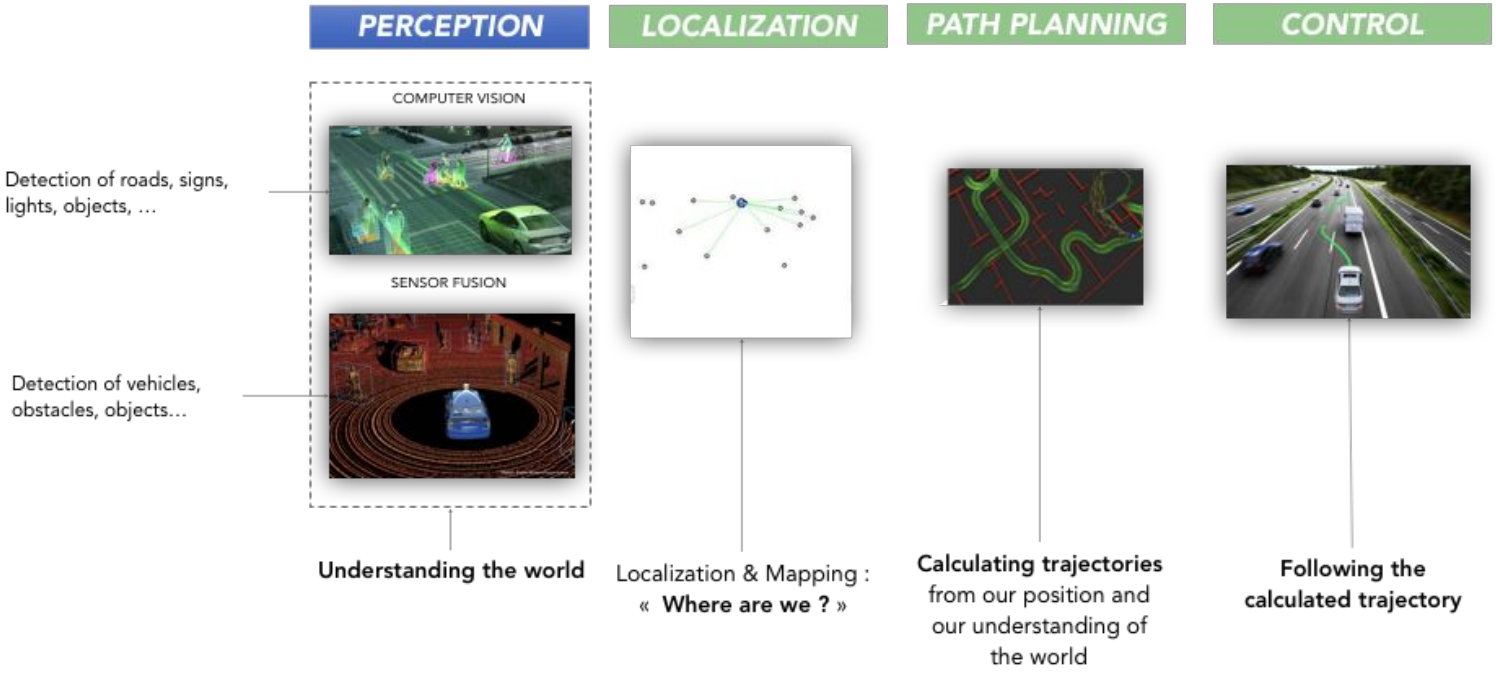
Another example is using the **data fusion approach to determine the traffic state (low traffic, traffic jam, medium flow) using roadside collected acoustic, image and sensor data**. In the field of autonomous driving sensor fusion is used to combine the redundant information from complementary sensors in order to obtain a more accurate and reliable representation of the environment.

Although technically not a dedicated sensor fusion method, modern Convolutional neural network based methods can simultaneously process very many channels of sensor data (such as Hyperspectral imaging with hundreds of bands) and fuse relevant information to produce classification results.

Example : Sensor Fusion in Self-Driving-Cars

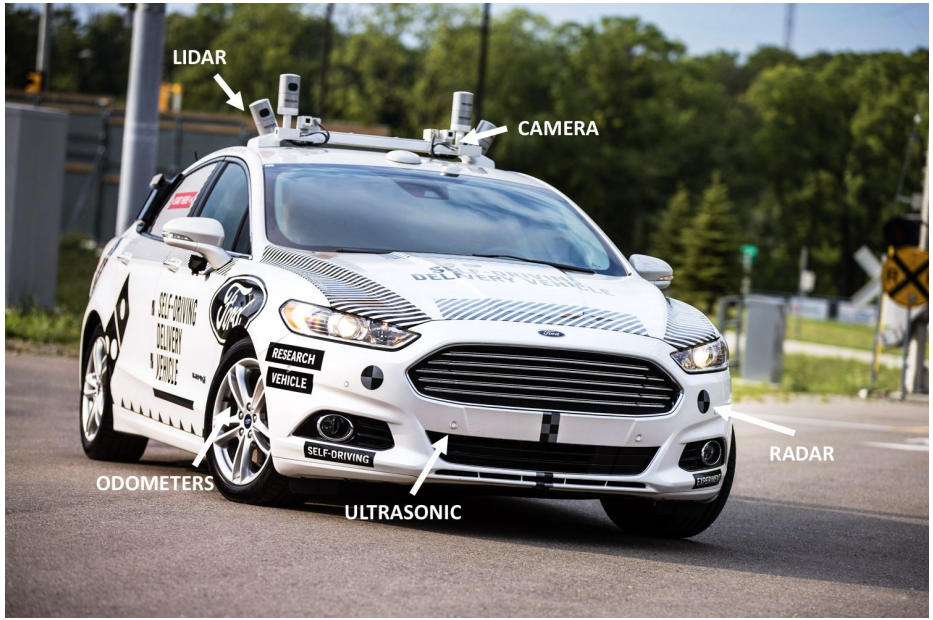


Example : Sensor Fusion in Self-Driving-Cars



Example：Sensor Fusion in Self-Driving-Cars

The autonomous vehicle uses a large number of sensors to understand its environment, to locate and move.



Example : Sensor Fusion in Self-Driving-Cars

Each of these sensors has advantages and disadvantages. The aim of sensor fusion is to use the advantages of each to precisely understand its environment.

- The camera is a very good tool for detecting roads, reading signs or recognizing a vehicle.
- The Lidar is better at accurately estimating the position of this vehicle.
- The Radar is better at accurately estimating the speed.

In the cycle of operation of an autonomous vehicle, there is Perception. This step is crucial for the vehicle to detect obstacles, roads, pedestrians ... A vehicle is full of sensors, so it is very important to understand how these sensors are used to understand how a car can drive without a driver.

Example : Sensor Fusion in Self-Driving-Cars

Kalman filters : The algorithm used to merge the data is called a Kalman filter.

The Kalman filter is one of the most popular algorithms in data fusion. Invented in 1960 by Rudolph Kalman, it is now used in our phones or satellites for navigation and tracking. The most famous use of the filter was during the Apollo 11 mission to send and bring the crew back to the moon.

When to use a Kalman Filter ?

A Kalman filter can be used for data fusion to estimate the state of a dynamic system (evolving with time) in the present (filtering), the past (smoothing) or the future (prediction). Sensors embedded in autonomous vehicles emit measures that are sometimes incomplete and noisy. The inaccuracy of the sensors (noise) is a very important problem and can be handled by the Kalman filters.

A Kalman filter is used to estimate the state of a system, denoted x. This vector is composed of a position p and a velocity v.

Example : Sensor Fusion in Self-Driving-Cars

$$x = \begin{pmatrix} p \\ v \end{pmatrix}$$

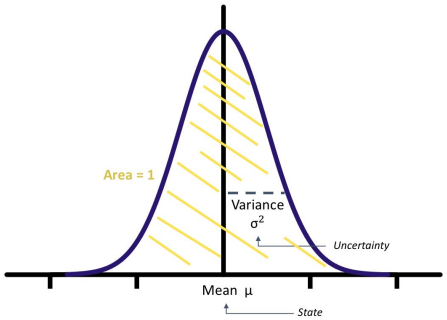
At each estimate, we associate a measure of uncertainty P.

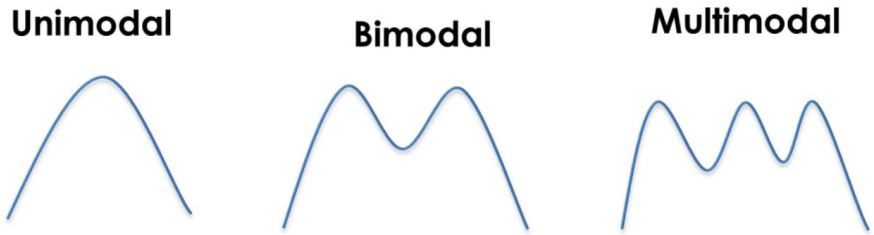
By performing a fusion of sensors, we take into account different data for the same object. A radar can estimate that a pedestrian is 10 meters away while the Lidar estimates it to be 12 meters. The use of Kalman filters allows you to have a precise idea to decide how many meters really is the pedestrian by eliminating the noise of the two sensors.

A Kalman filter can generate estimates of the state of objects around it. To make an estimate, it only needs the current observations and the previous prediction. The measurement history is not necessary. This tool is therefore light and improves with time.

How it looks like ? State and uncertainty are represented by Gaussians.

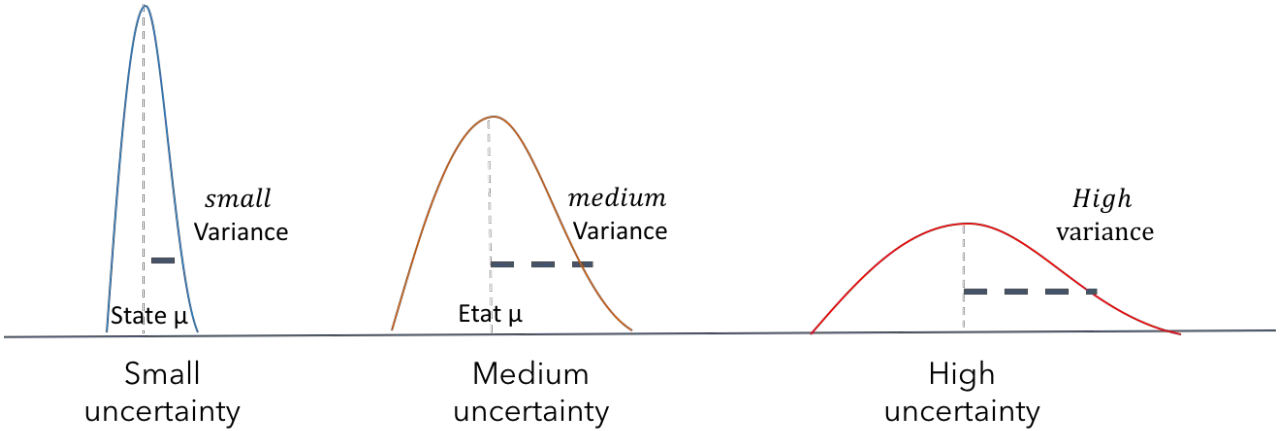
Example : Sensor Fusion in Self-Driving-Cars





A Gaussian is a continuous function under which the area is 1. This allows us to represent probabilities. We are on a probability to normal distribution. The uni-modality of the Kalman filters means that we have a single peak each time to estimate the state of the system.

Example : Sensor Fusion in Self-Driving-Cars



We have a mean μ representing a state and a variance σ^2 representing an uncertainty. The larger the variance, the greater the uncertainty.

Gaussians make it possible to estimate probabilities around the state and the uncertainty of a system. A Kalman filter is a continuous and uni-modal function.

Example : Sensor Fusion in Self-Driving-Cars

Bayesian Filtering

In general, a Kalman filter is an implementation of a Bayesian filter, ie a sequence of alternations between prediction and update or correction.

Bayes filter

Prediction: We use the estimated state to predict the current state and uncertainty.

Update: We use the observations of our sensors to correct the predicted state and obtain a more accurate estimate.

To make an estimate, a Kalman filter only needs current observations and the previous prediction. The measurement history is not necessary.

Example : Sensor Fusion in Self-Driving-Cars

Maths

The mathematics behind the Kalman filters are made of additions and multiplications of matrices. We have two stages: Prediction and Update.

Prediction

Our prediction consists of estimating a state x’ and an uncertainty P’ at time t from the previous states x and P at time t-1.

Prediction formulas

F: Transition matrix from t-1 to t

ν: Noise added

Q: Covariance matrix including noise

Example : Sensor Fusion in Self-Driving-Cars

$$x' = Fx + u$$

$$P' = FPF^T + Q$$

Example : Sensor Fusion in Self-Driving-Cars

Update : The update phase consists of using a z measurement from a sensor to correct our prediction and thus predict x and P.

Update formulas

y: Difference between actual measurement and prediction, ie the error.

S: Estimated system error

H: Matrix of transition between the marker of the sensor and ours.

R: Covariance matrix related to sensor noise (given by the sensor manufacturer).

K: Kalman gain. Coefficient between 0 and 1 reflecting the need to correct our prediction.

The update phase makes it possible to estimate an x and a P closer to reality than what the measurements provide.

Example : Sensor Fusion in Self-Driving-Cars

$$y = z - Hx'$$

$$S = HP'H^T + R$$

$$K = P'H^T S^{-1}$$

$$x = x' + Ky$$

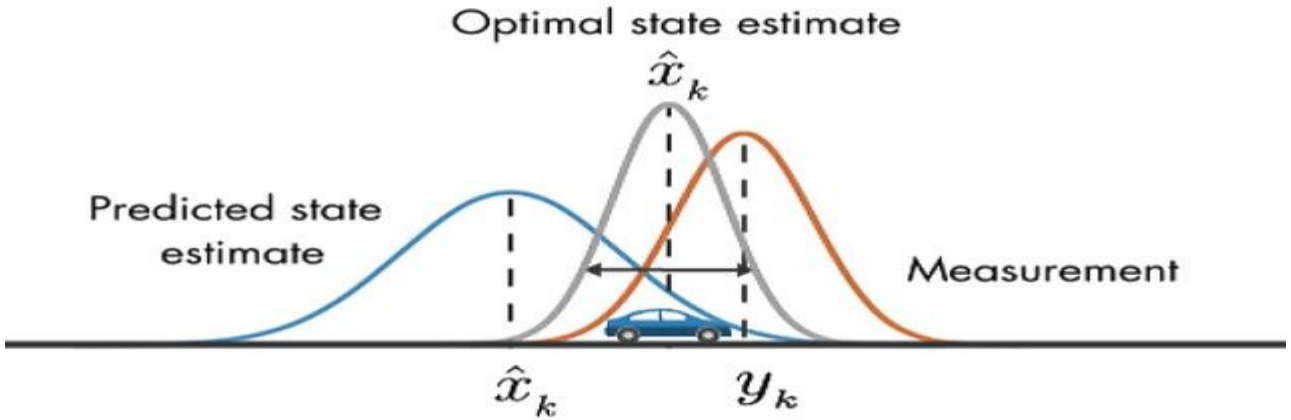
$$P = (I - KH)P'$$

Example : Sensor Fusion in Self-Driving-Cars

A Kalman filter allows predictions in real time, without data beforehand. We use a mathematical model based on the multiplication of matrices for each time defining a state x (position, speed) and uncertainty P.

Représentation Prior/Posterior

This diagram shows what happens in a Kalman filter.



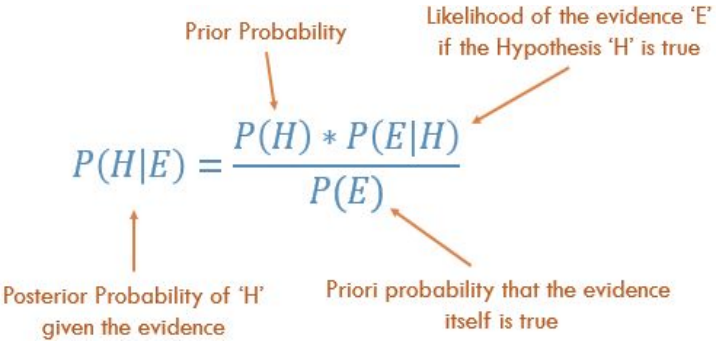
Example : Sensor Fusion in Self-Driving-Cars

Predicted state estimate represents our first estimate, our prediction phase. We are talking about prior.

Measurement is the measurement from one of our sensors. We have better uncertainty but the noise of the sensors makes it a measurement that is always difficult to estimate. We talk about likelihood.

Optimal State Estimate is our update phase. The uncertainty is this time the weakest, we accumulated information and allowed to generate a value more sure than with our sensor alone. This value is our best guess. We speak of posterior.

What a Kalman filter implements is actually a Bayes rule.



Example : Sensor Fusion in Self-Driving-Cars

In a Kalman filter, we loop predictions from measurements. Our predictions are always more precise since we keep a measure of uncertainty and regularly calculate the error between our prediction and reality. We are able from matrix multiplications and probability formulas to estimate velocities and positions of vehicles around us.

“Extended/Unscented” filters and non-linearity

An essential problem arises. Our mathematical formulas are all implemented with linear functions of type $y = ax + b$.

A Kalman filter always works with linear functions. On the other hand, when we use a Radar, the data is not linear.

Example : Sensor Fusion in Self-Driving-Cars

Functionment of a Radar

The Radar sees the world with three measures:

ρ (rho): The distance to the object that is tracked.

φ (phi): The angle between the x-axis and our object.

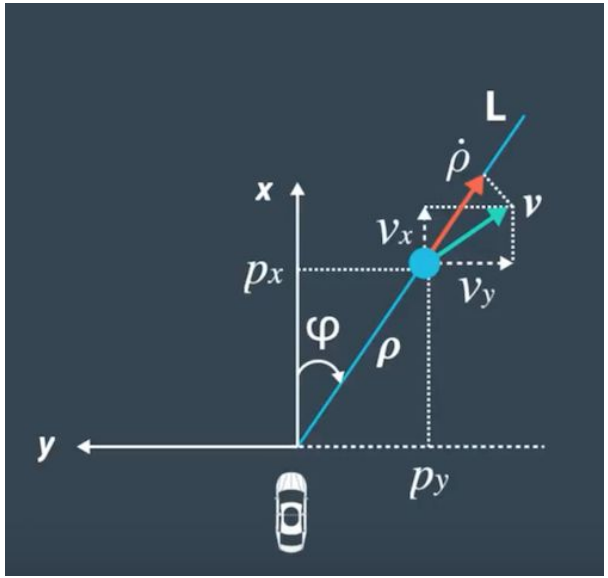
$\dot{\rho}$ (rhodot): The change of ρ, resulting in a radial velocity.

These three values make our measurement a nonlinear given the inclusion of the angle φ.

Our goal here is to convert the data ρ, φ, $\dot{\rho}$ to Cartesian data (px, py, vx, vy).

If we enter non-linear data in a Kalman filter, our result is no longer in uni-modal Gaussian form and we can no longer estimate position and velocity.

Example : Sensor Fusion in Self-Driving-Cars



So we use approximations, which is why we are working on two methods:

*- **Extended Kalman filters** use Jacobian and Taylor series to linearize the model.*

*- **Unscented Kalman filters** use a more precise approximation to linearize the model.*

Example : Sensor Fusion in Self-Driving-Cars

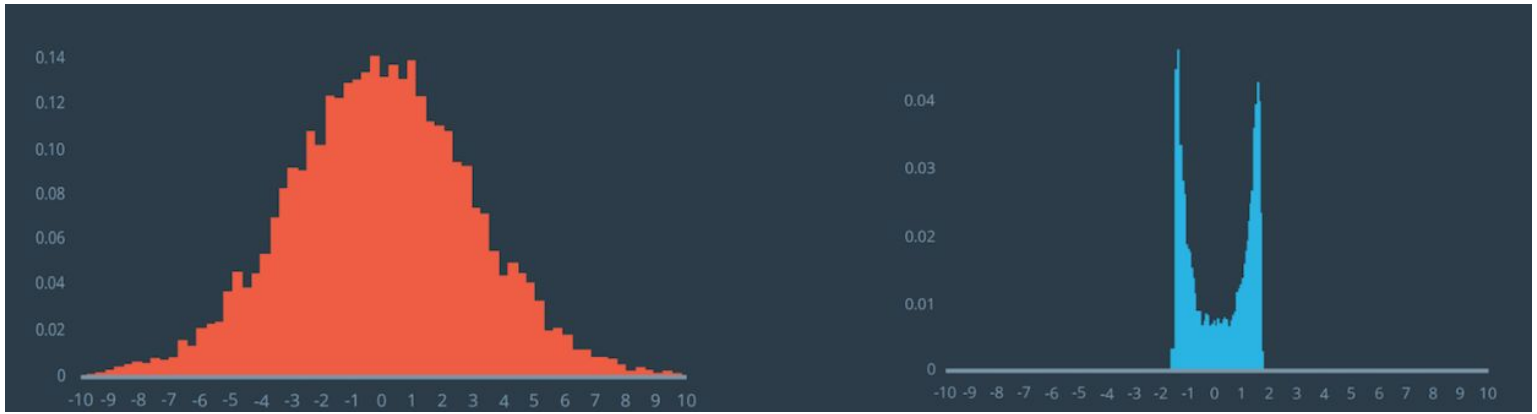
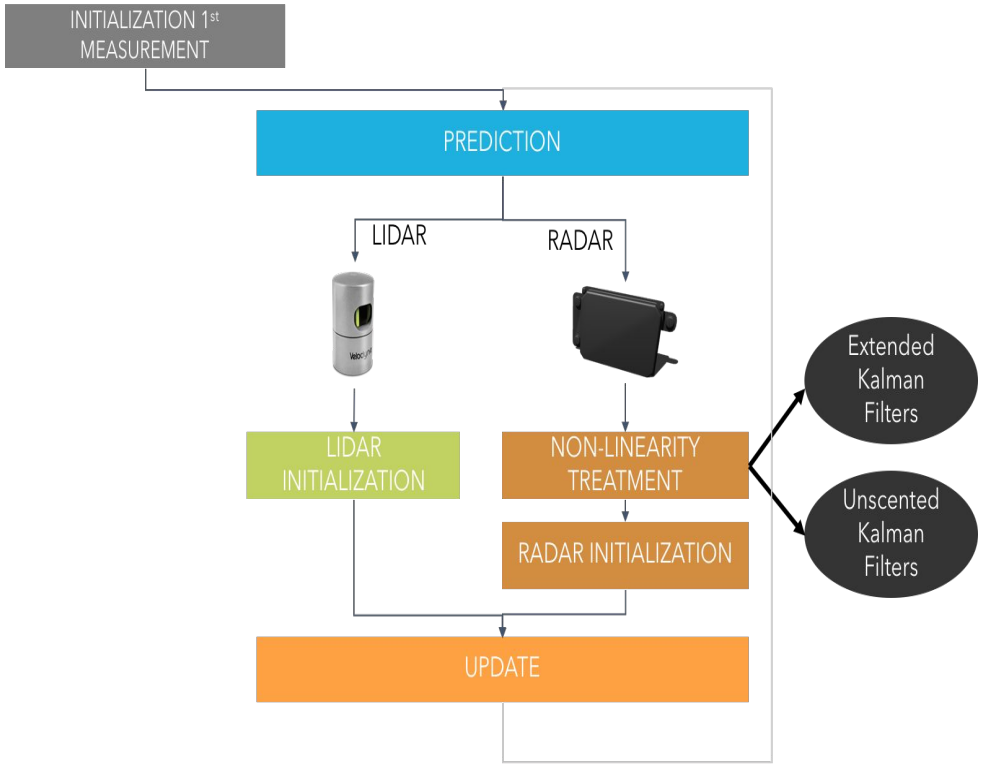


Image : Gaussian vs Non Linearity

To deal with the inclusion of non linearity by the Radar, techniques exist and allow our filters to estimate position and velocity of the objects that we wish to track.

Example : Sensor Fusion in Self-Driving-Cars



Sensor fusion is one of the most important topics in the field of autonomous vehicles. Fusion algorithms allow a vehicle to understand exactly how many obstacles there are, to estimate where they are and how fast they are moving.

Depending on the sensor used, we can have different implementations of the Kalman Filter.

Thank You

References & Sources of Information :

- google.com
- wikipedia.org
- sciencedirect.com
- towardsdatascience.com
- avnet.com