

INDEX

S.No	Titles	Page No
1	Introduction	
2	Definition	
3	Sensor Fusion Importance	
4	Higher accuracy through Design	
5	Key aspects of Design	
6	Use cases and Sensor Requirements	
7	Capabilities of Individual Sensors	
8	Understanding Sensor Fusion Capability with respect to ADAS	
9	Sensor Fusion Steps	
10	Classification of Sensor Fusion	
11	Classification based on Abstraction Level	
12	Classification based on structure	
13	Classification based on Relation between different input sources	
14	Classification based on Input/Output	
15	Sensor Fusion Techniques and Algorithms	
16	Classification of different algorithms	
17	State Estimation / Tracking Techniques	
18	Kalman Filter	
19	Difficulties in Sensor Fusion Implementation	
20	How to Overcome the Difficulties in Sensor Fusion Implementation	
21	Importance of Validation	
22	Validation of Sensor Fusion	
23	Future Trends	
24	Aptive's Approach	
25	Visteon's Approach	
26	References	

SENSOR FUSION

Chapter 1 – Introduction

Sensor data fusion acts as a human brain, providing the car with a complete and accurate picture of its surroundings ⁽⁸⁾.

Vehicles equipped with Advanced Driver Assistance Systems (ADAS) and Autonomous Vehicles (AV) use different sensors like **Cameras, Radars, Lidars, and Ultrasonic sensors** which act as the **eyes** of an autonomous vehicle (AV) and help the vehicle perceive its surroundings, i.e., people, objects, traffic, road geometry, weather, etc. This perception is critical, and it ensures that the AV can make **the right decisions, i.e., stop, accelerate, turn, etc⁽¹⁾**.

As we reach **higher levels of autonomy** in autonomous driving, the **complexity increases** substantially. Here multiple sensors are required to understand the environment correctly. But every sensor is **different** and has its limitations, e.g., a camera will work very well for lane detection or object classification. In contrast, a radar may provide good data for long-range detection or in different light conditions ⁽¹⁾.

To use computer technology to automatically analyze and synthesize information and data from multiple sensors or sources under certain criteria to complete the information processing process required for making decisions and estimations. The sensors that serve as data sources can be the same (isomorphic) or different (heterogeneous), but they are not simply stacked together. They must be deeply integrated from the data level⁽⁷⁾.

In short, sensor fusion technology is just like a "coach" because it is capable of kneading sensors with different abilities into a united team of players that can work together and complement each other to win the game⁽⁷⁾.

To power the sensor fusion, we need **inputs**. These inputs are typically sensors, such as **cameras, lidar, radar and ultrasonic sensors**, but can also include information from the car-to-car communication system⁽¹⁴⁾.

Following sensor fusion, the **output** is the **environment model**, which can be divided into different sub-products⁽¹⁴⁾:

1. The **static environment model** describes all static raised objects, such as walls or parked vehicles⁽¹⁴⁾.
2. The **dynamic environment model** describes all potentially moving objects, such as pedestrians and cars⁽¹⁴⁾.
3. The **drivable areas** include empty space and areas to overtake⁽¹⁴⁾.
4. The **regulatory environment model** describes all the elements that have legal relevance, such as lane markings and traffic signs⁽¹⁴⁾.

This **environment model** can then be used by the car to power automated driving functions.

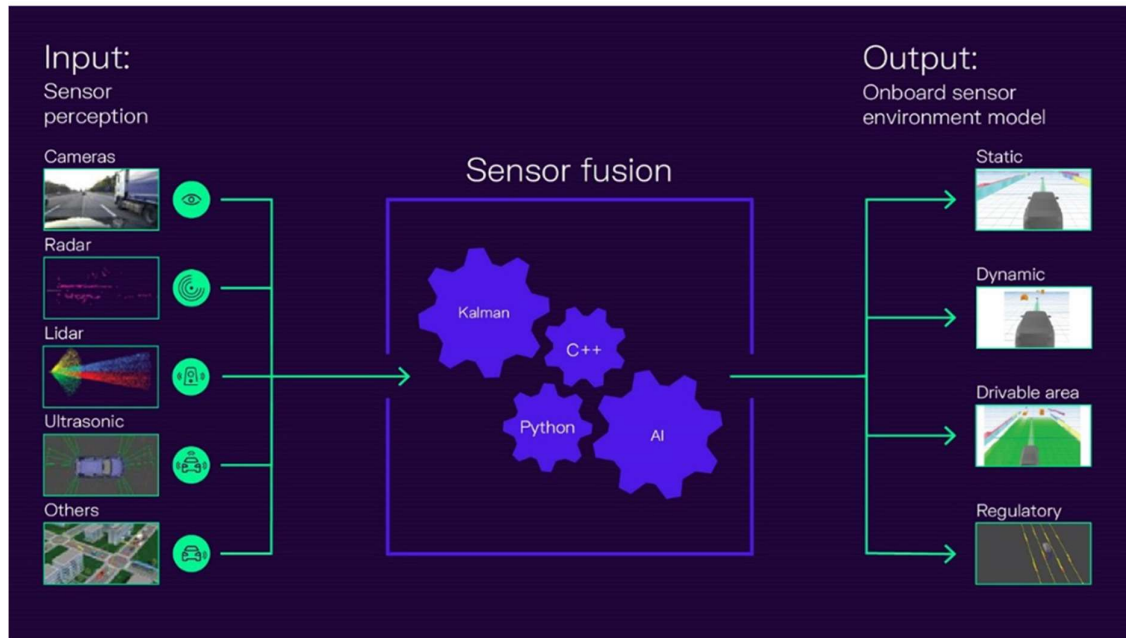


Figure 1 - Information from various kinds of sensors is fused together to produce an environment model⁽¹⁴⁾.

Sensor Fusion Importance:

Many advanced driver assistance systems (ADAS) are currently trying to utilise multisensory architectures, where the driver assistance algorithm receives data from a multitude of sensors. As mono-sensor systems cannot provide reliable and consistent readings under all circumstances because of errors and other limitations, fusing data from multiple sensors ensures that the environmental parameters are perceived correctly and reliably for most scenarios, thereby substantially improving the reliability of the multi-sensor-based automotive systems⁽³⁾.

Using Sensor Fusion techniques, data from multiple sensors is **fused** in Autonomous Vehicles to provide the best possible input so that the AV takes the **correct decisions (brake, accelerate turn, etc.)**

Sensor Fusion **improves the overall performance** capability of an Autonomous Vehicle, and there are multiple fusion techniques and which one to use depends on the feature's Operation Design Domain (ODD).⁽¹⁾

ODD:

SAE J3016 defines ODD as “Operating conditions under which a given driving automation system or feature thereof is specifically designed to function, including, but not limited to, environmental, geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics ”⁽⁶⁾.

An ODD definition describes specific operating conditions in which the automated driving system is designed to properly operate. It specifies what operating parameters the CAV must be able to manage; for example, weather conditions, infrastructure, location, time of day and everything else that can have an impact on the driving situation. The ODD is thus an important part of the safety concept of a vehicle and must be valid throughout its entire service life for a particular configuration of the CAV.

It integrates the acquired data from multiple sensing modalities to reduce the number of detection uncertainties and overcome the shortcomings of individual sensors operating independently. Moreover, sensor fusion helps to develop a consistent model that can perceive the surroundings accurately in various environmental conditions. For instance, camera and radar fusion may provide high-resolution images and the relative velocities of the detected obstacles in the perceived scene.⁽²⁾

The complexity of the environment and features specifications drive what kind of fusion strategy and **what type of fusion requirement needs to be worked out**. Executing Sensor fusion is a complex activity, and one should account for numerous challenges. ⁽¹⁾.

Enhanced Accuracy

A single sensors may be subject to inaccuracies or noise due to various factors, such as environmental conditions, manufacturing defects, or wear and tear. In this regard, sensor fusion plays a pivotal role in reducing errors and noise in the data collected from multiple sensors, leading to enhanced accuracy in decision-making and overall system performance.

This improvement in accuracy is particularly important in applications where precision and **safety** are of utmost importance, such as **autonomous vehicles**.

For example where enhanced accuracy is crucial is in the development of autonomous vehicles. These **vehicles rely heavily on sensor data to make real-time decisions about their surroundings**, such as detecting obstacles, determining the position of other vehicles, and navigating complex road networks. By fusing data from various sensors like cameras, radar, lidar, and GPS, autonomous vehicles can achieve a higher. Sensors are the eyes and ears of the system, constantly collecting data about the environment. These sensors are increasingly becoming part of the **Internet of Things (IoT)**, meaning they're connected to the internet and can transmit data in real-time

Robustness

Robustness is another significant advantage of sensor fusion. By combining data from multiple sensors, sensor fusion **can compensate for the limitations or failures** of individual

sensors, thereby ensuring that the system remains functional and reliable even in challenging conditions.

The concept of **redundancy** is closely related to robustness in sensor systems. Redundancy refers to the use of multiple sensors or sensor types to measure the same parameter or environmental characteristic. This redundancy can help mitigate the impact of sensor failure or degradation, as other sensors can continue to provide valuable information. For example, if one sensor fails to detect an obstacle due to a malfunction, other sensors in the system can still provide information about the obstacle, ensuring that the system remains aware of its environment.

In applications such as autonomous vehicles, robustness is of paramount importance. These vehicles must operate safely and reliably in a wide range of environmental conditions and scenarios, and sensor failure can have severe consequences for the vehicle's occupants and other road users. Through sensor fusion, these vehicles fuse data from multiple sensors to achieve a level of robustness that would be difficult to attain using individual sensors alone.

To ensure high levels of accuracy with Sensor Fusion, one must ensure the right KPIs and robust design is being used, it includes⁽¹⁾

Higher accuracy through design⁽⁹⁾:

Key aspects of design⁽⁹⁾:

To achieve high quality performance of Sensor Fusion, there are some **key aspects of design** that must be considered:

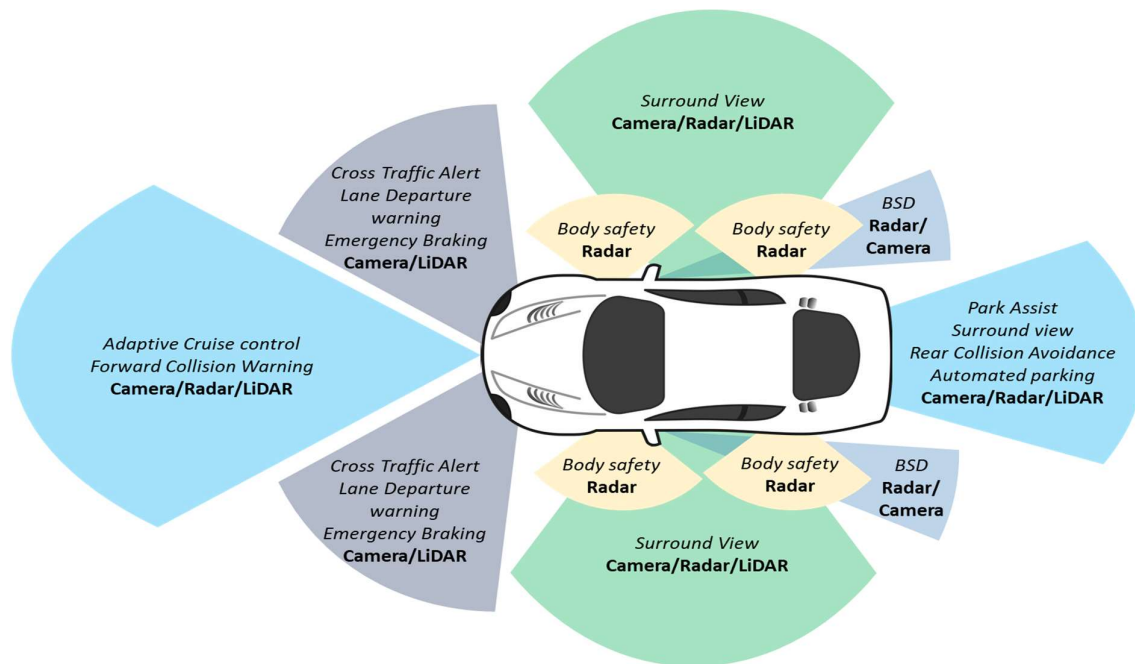
- **Selection of algorithms** for data association and estimation technique

- There are many algorithms available for data association the popular algorithms are nearest neighbour, probabilistic, and joint probabilistic data association technique.⁽⁹⁾
- The various popular estimation techniques like linear **Kalman filter, Extended Kalman filter, particle filter etc.**
- **The fusion strategy**
 - The data association algorithm and estimation technique shall be decided based on type of sensor used for fusion, state estimation requirement (dynamic/static object estimation) and sensor outputs⁽⁹⁾.
- **Track Management:**
 - To reduce false output, fusion track management needs to build confidence before initialize track however if it is taken more time then it will lead to latency which will delay in action by AD system which is not good for AD system performance. The track initialization strategy shall be decided considering operating environment, feature ODD, etc. The same is case for track deletion⁽⁹⁾
- **Filter tuning**
 - Filter tuning very critical for filter performance. The practical aspect needs to be considered for the filter tuning and this can be done using sensor characterization using real world data⁽⁹⁾

What Is a Kalman Filter?

The Kalman filter is an algorithm that estimates the state of a system from measured data. It was primarily developed by the Hungarian engineer Rudolf Kalman, for whom the filter is named. The filter's algorithm is a two-step process: the first step predicts the state of the system, and the second step uses noisy measurements to refine the estimate of system state.

There are now several variants of the original Kalman filter. These filters are widely used for applications that rely on estimation, including computer vision, guidance and navigation systems, econometrics, and signal processing⁽⁴⁾.

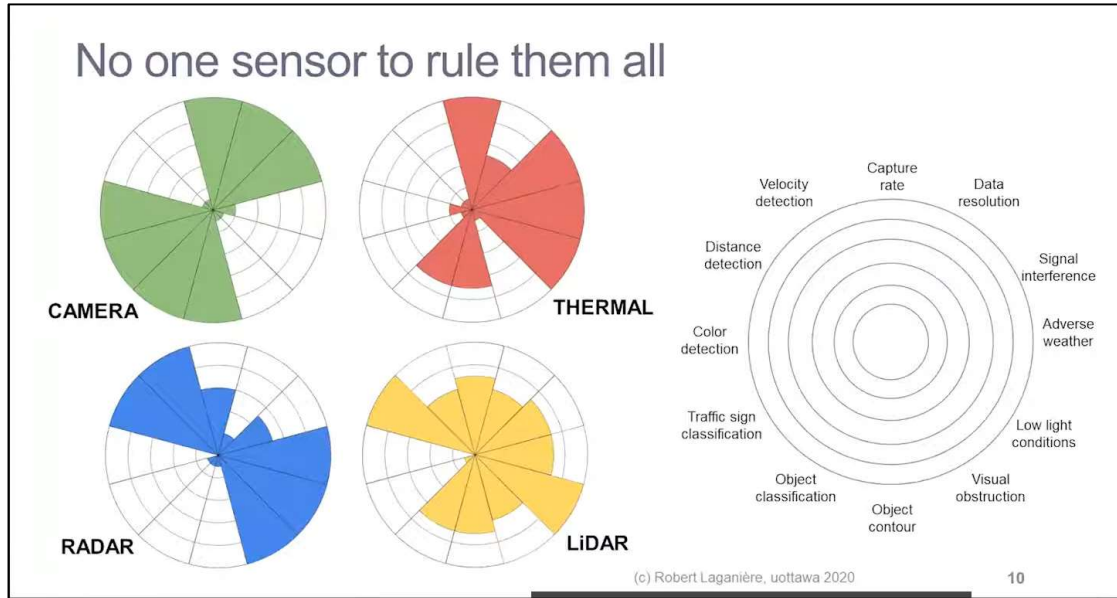
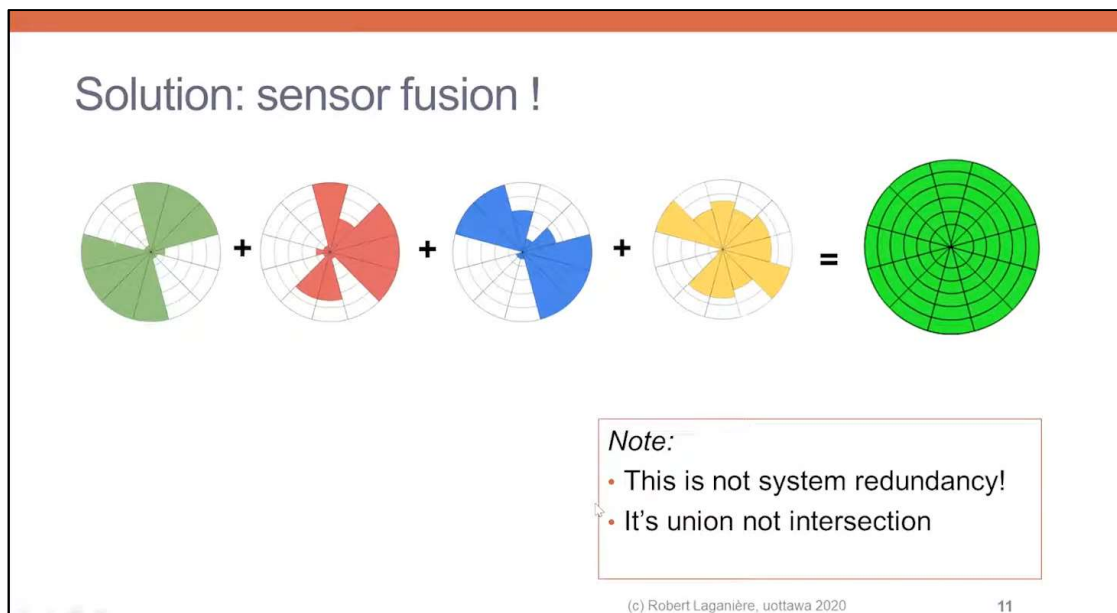
Use cases and Sensor requirements⁽¹⁵⁾:

Application	Description	Optimal Sensor Fusion Case
Adaptive Cruise Control (ACC)	This application requires a very long range with small FoV.	Camera + Lidar + Radar
Blind Spot Detection (BSD)	Medium range of 40m with velocity estimation	Camera + Radar
Lane Change Assist (LCA)	Medium range of 40m with velocity estimation	Camera + Radar
Automatic Emergency Braking (AEB)	3D positional accuracy is highly required	Camera + Radar
Automated Parking	Surround view with 3D precise object detection	Camera + Radar
Park Assist	-	Camera + Lidar + Radar
Body Safety	-	RADAR
Rear collision avoidance	-	Camera + Lidar + Radar
Emergency Braking		Camera + Lidar
Lane Departure Warning	-	Camera + Lidar
Forward Collision Warning	-	Camera + Lidar + Radar

Capabilities of Individual Sensors^(1,5,2):

S.No	Capabilities	Camera	Radar	LiDAR
1	Long range detection	<i>Average</i>	<i>Good</i>	<i>Average</i>
2	Different light conditions	<i>Average</i>	<i>Good</i>	<i>Good</i>
3	Different weather conditions (light rain, fog)	<i>Poor</i>	<i>Good</i>	<i>Poor</i>
4	Object classification	<i>Good</i>	<i>Poor</i>	<i>Good</i>
5	Stationary object detection	<i>Good</i>	<i>Poor</i>	<i>Good</i>
6	Object Detection	<i>Average</i>	<i>Good</i>	<i>Good</i>
7	Pedestrian Detection	<i>Good</i>	<i>Poor</i>	<i>Average</i>
8	Dirt	<i>Poor</i>	<i>Good</i>	<i>Average</i>
9	Velocity	<i>Average</i>	<i>Good</i>	<i>Average</i>
10	Long range accuracy	<i>Average</i>	<i>Good</i>	<i>Good</i>
11	Data Density	<i>Good</i>	<i>Poor</i>	<i>Average</i>
12	Packaging	<i>Average</i>	<i>Good</i>	<i>Poor</i>
13	Resolution	<i>Good</i>	<i>Poor</i>	<i>Average</i>
14	Lane Detection	<i>Good</i>	<i>Poor</i>	<i>Poor</i>
15	Obstacle edge detection	<i>Good</i>	<i>Poor</i>	<i>Good</i>

Table 1 – Capability of individual sensors

**Solution:**

Understanding Sensor Fusion capability with respect to ADAS⁽¹¹⁾

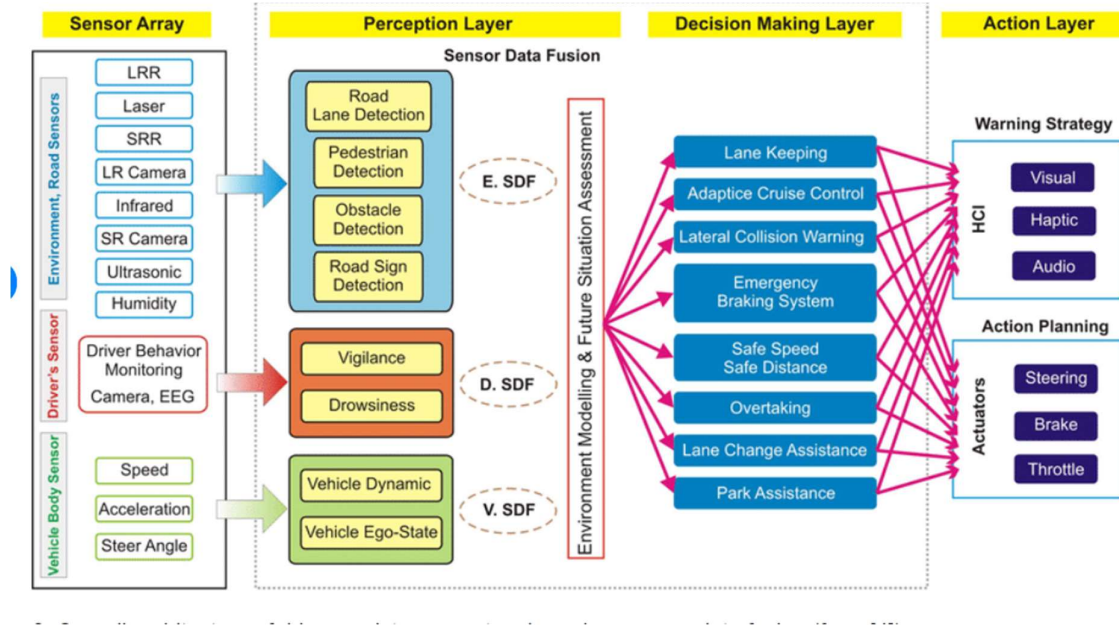


Fig. 3. Overall Architecture of Driver Assistance System based on sensor data fusion⁽¹¹⁾.

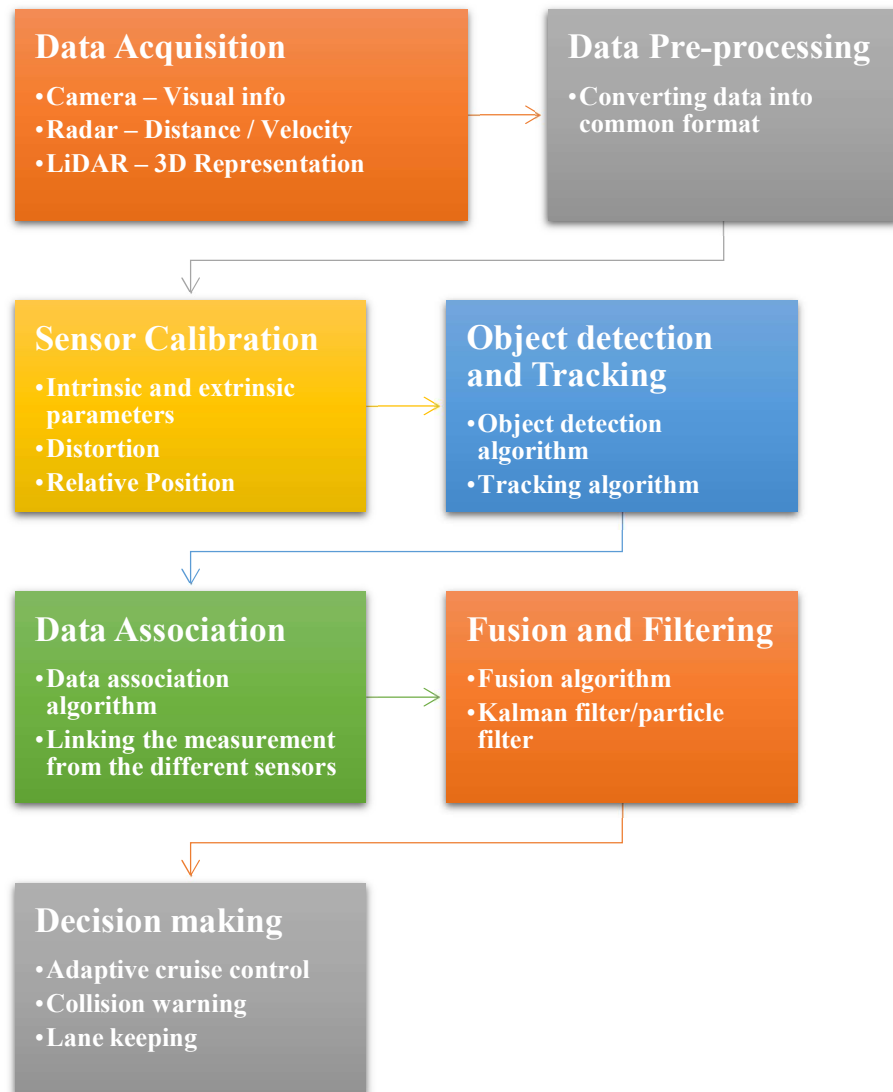
After implementing sensor assembly (refer chapter XXX), a practical framework needs to be designed. We assume a driver assistance system as a vehicle–driver–environment interactive closed-loop system; moreover, we focus on not only the current situation of vehicle and driver, but also the future situation by predicting the potential collision probability distribution. In this framework, onboard sensors provide real-time information about drivers, traffic environment, and vehicles. Configure these sensors is closely related to the application domain.⁽¹¹⁾

Now we take Perception layer to focus more on Sensor Fusion.

Perception Layer: this layer aims to give a realistic representation of the environment to the applications; it can be seen as an intermediate layer between the sensorial sub-system and the system applications. It includes sensor fusion of previous layer in order to environment

modeling, overall sensor fusion, and future situation assessment. The role of the perception layer is to:

- Carry out a perception enhancement of the external scenario, even independently on any given application.
- Describe the environment and the traffic scenario (obstacles and host vehicle dynamics, road geometry, etc.) in a formal way.
- Support the specific needs of the ADAS system in the reconstruction of the environment.
- Act as an intermediate layer between sensors and functions, defined by the I/O protocols and interfaces.

Sensor Fusion Steps:

The process of sensor fusion involves several steps⁽¹⁰⁾:

Data Acquisition:

Each sensor collects data about the vehicle's surroundings. For example, cameras capture visual information, radar measures the distance and velocity of objects, lidar creates a 3D representation of the environment, and ultrasonic sensors detect nearby objects⁽¹⁰⁾.

Data Pre-processing:

The data collected by different sensors may have different formats, resolutions, or coordinate systems. Pre-processing involves converting the data into a common format and aligning it with a unified coordinate system. This step ensures that the data can be effectively combined and compared⁽¹⁰⁾.

Sensor Calibration:

Calibration refers to the process of determining the relation between the output of measuring instrument (here multi sensors) and the value of the input quantity or attribute according to a standard measurement. Sensors need to be accurately calibrated to account for variations in their measurements and position. Calibration involves determining the intrinsic and extrinsic parameters of each sensor, such as its field of view, distortion characteristics, and relative position to the vehicle. Precise calibration is crucial for accurate sensor fusion^{(10),(11)}.

Object Detection and Tracking:

The next step is to identify and track objects in the environment using the sensor data. Each sensor provides information about objects, such as their position, size, velocity, and classification. Object detection algorithms and tracking algorithms are applied to the sensor data to identify and monitor objects over time⁽¹⁰⁾.

Data Association:

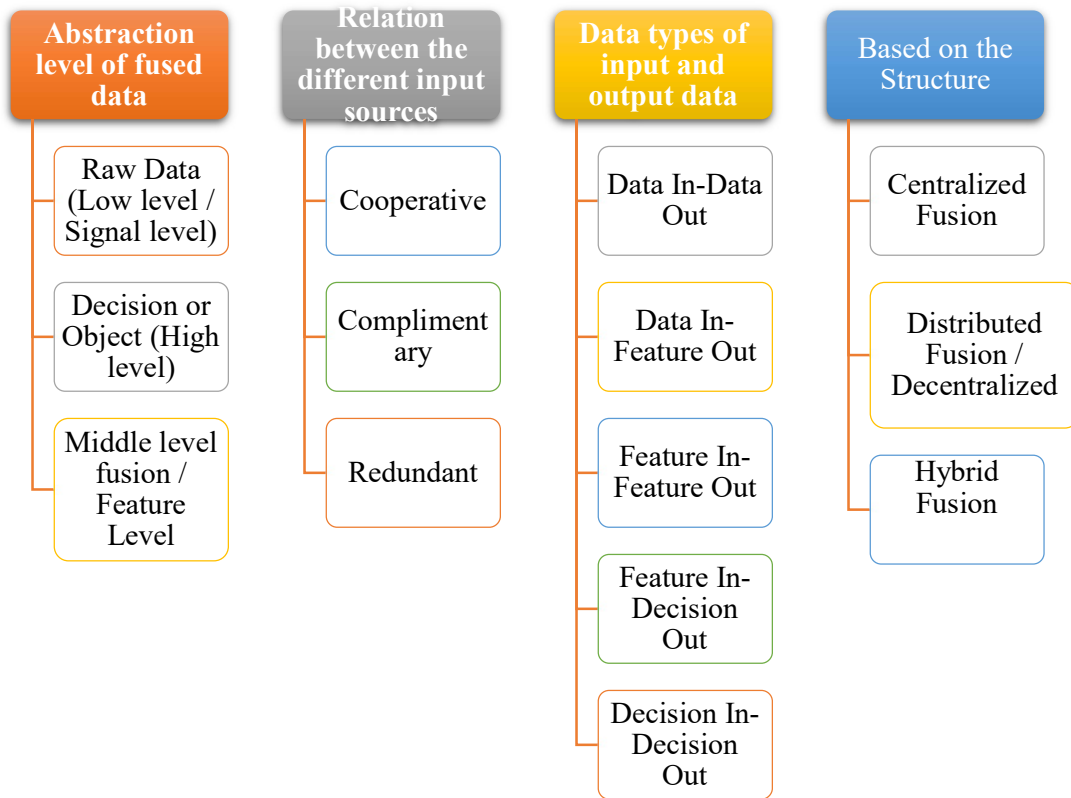
Data association refers to the process of linking the measurements from different sensors to the same objects. This step involves determining which measurements correspond to the same object in the environment. For example, associating a radar measurement with a corresponding object detected by a camera. Data association algorithms consider factors like proximity, object motion, and sensor characteristics to establish reliable associations⁽¹⁰⁾.

Fusion and Filtering:

Once the sensor data is associated with objects, fusion algorithms are used to combine the information from multiple sensors. These algorithms can range from simple techniques like averaging or voting to more sophisticated methods like **Kalman filters or particle filters**. The fused data provides a more accurate representation of the environment and the objects within it⁽¹⁰⁾.

Decision Making:

The final step involves using the fused sensor data to make decisions or trigger actions. This could include collision warnings, adaptive cruise control, lane keeping assistance, or autonomous driving manoeuvres. The decisions are based on the fused information, which provides a more comprehensive view of the environment and improves the reliability and accuracy of the system⁽¹⁰⁾.

Classification of Sensor fusion⁽³⁾⁽⁷⁾:

There are four primary approaches to combine sensory data from various sensing modalities in the MSDF frameworks⁽²⁾⁽⁷⁾:

1. **High-level fusion (HLF) / Object Level,**
2. **Low-level fusion (LLF) / Data Level, and**
3. **Mid-level fusion (MLF) / Feature Level⁽¹⁶⁾**

Sensor Fusion Levels Comparison:				
Level	Concept	Methods	Advantages	Disadvantages
Data Level	Combines raw sensor data before object detection/feature extraction	Kalman filters, probabilistic methods, PointNet	Handles new objects/sensors, flexible for object detection algorithms	Computationally expensive, data alignment & interpretation challenges
Feature Level	Combines extracted features from individual sensors	R-CNN variants, YOLO	Balances complexity & performance, sensor-specific feature extraction, handles new objects	Requires effective feature extraction, resource-intensive
Object Level	Combines information about already identified objects	Voting, Kalman filters, Bayesian inference	High accuracy for identified objects, efficient	Relies on individual sensor detection/tracking, limited to known objects

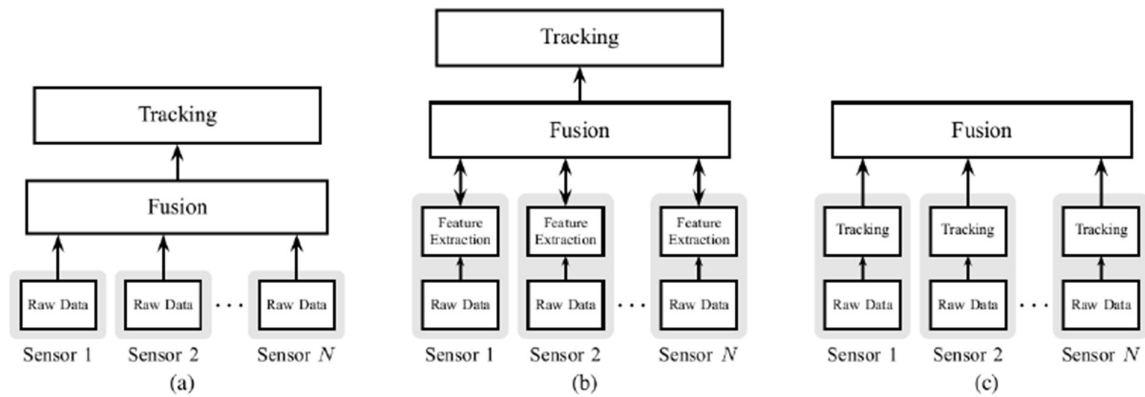


Fig xx – Basic structure of common sensor data fusion architecture for ADAS. (a) low-level fusion, (b) feature-level fusion and (c) high-level track fusion.

In the **HLF approach**, each sensor carries out object detection or a tracking algorithm independently and subsequently performs fusion. For instance, the HLF approach to fuse the processed data, i.e., radar signals and LiDAR point clouds independently and subsequently

used a non-linear Kalman Filter method to detect obstacles and state tracking. The HLF approaches are often adopted due to a lower relative complexity than the LLF and MLF approach. However, HLF provides inadequate information as classifications with a lower confidence value are discarded if, for example, there are several overlapping obstacles⁽²⁾.

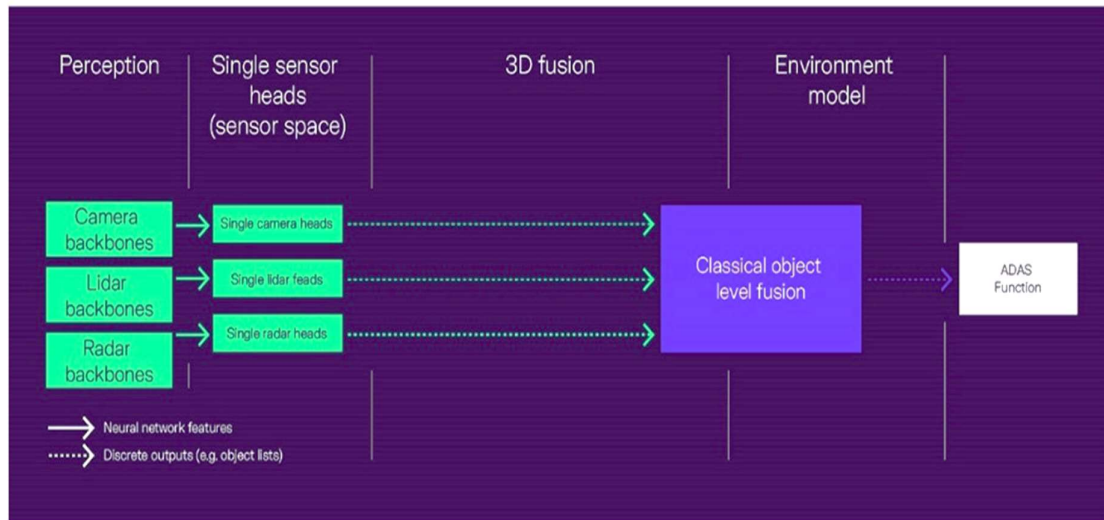


Fig xx - Object-level sensor fusion predicts objects for each sensor individually and fuses them to create one unified list.

In Object – level Sensor Fusion level, it uses independent network backbones and sensor heads to predict objects for each sensor individually. Sensor fusion then receives these object lists and fuses the individual objects to create one unified list containing the fused objects of all sensors. One drawback of this fusion strategy is the late combination of information. If two or more sensors only have partial information on a certain object, that object might not be detected by any sensor. However, if the partial information is already combined on feature level and the combined features are provided to an object detection head, it's likely that the object will be detected. This is exactly what AI-based sensor fusion does⁽¹⁴⁾.

Contrarily, with the LLF approach, data from each sensor are integrated (or fused) at the lowest level of abstraction (raw data). Therefore, all information is retained and can potentially improve the obstacle detection accuracy. proposed a two-stage 3D obstacle detection architecture, named 3D-cross view fusion (3D-CVF). In the second stage, they utilized the LLF approach to fuse the joint camera-LiDAR feature map obtained from the first stage with the low-level camera and LiDAR features using a 3D region of interest (RoI)-based pooling method. They evaluated the proposed method on KITTI and nuScene datasets and reported that the object detection results outperformed the state-of-the-art 3D object detectors in the KITTI leaderboard. In practice, the LLF approach comes with a multitude of challenges, not least in its implementation. It requires precise extrinsic calibration of sensors to accurately fuse their perceptions of the environment. The sensors must also counterbalance ego-motion (3D motion of a system within an environment) and be temporally calibrated⁽²⁾.

The MLF / Feature Level, is an abstraction level between LLF and HLF. It fuses multi-target features extracted from the corresponding sensor data (raw measurements), such as color information from images or location features of radar and LiDAR, and subsequently perform recognition and classification on the fused multisensory features. proposed a feature-level sensor fusion framework to detect targets in a dynamic background environment with limited communication capability. They utilized the Symbolic Dynamic Filtering (SDF) algorithm to extract the low-dimensional features from multiple infrared sensors in different orientations and in the presence of changing ambient light intensities and subsequently fusing the extracted features as clusters with the agglomerative hierarchical clustering algorithm for moving target detection. The MLF, however, appears to be insufficient to achieve a SAE Level 4 or Level 5 AD system due to its limited sense of the environment and loss of contextual information⁽²⁾.

Feature Level Fusion (AI Based) - Instead of fusing object lists from different sensors, AI-based sensor fusion combines information at the feature level, and predictions are made only once and on the basis of all available information – not for each sensor individually⁽¹⁴⁾.

The foundation of AI-based low level sensor fusion are feature maps of the individual sensors, which are extracted by independent network backbones. Spatial fusion transforms these feature maps into a unified space and fuses them together. The result is a unified representation of the vehicle's environment, containing the information of all sensors. A commonly used space is the bird's eye view⁽¹⁴⁾.

On top of this, temporal fusion incorporates time as an additional source of information. It exploits the output of the spatial fusion from previous time steps, fusing current and past information on feature level to improve the feature quality and compensate for missing features. The key component is a temporal memory, which aggregates and transports information over and through time⁽¹⁴⁾.

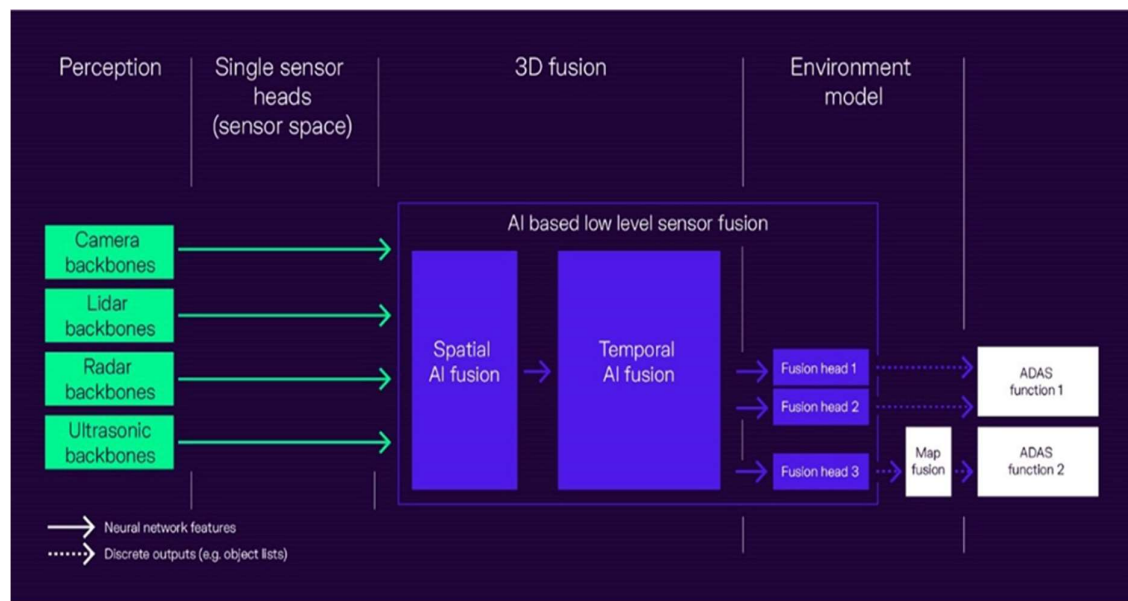


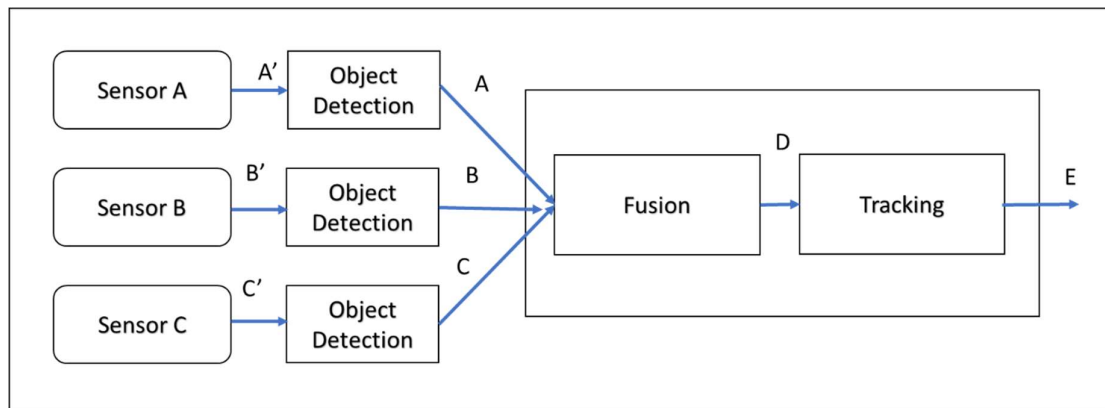
Fig - AI-based sensor fusion (feature level) is a more advanced approach that makes predictions only once and on the basis of all information.

The result of the spatial and temporal fusion steps is a high-quality, multimodal feature map in a unified 3D space. Based on the feature map, the heads, which define the environment model, compute the final predictions. Since an environment model for automated driving must provide a variety of information about the environment, so we use a multi-head architecture. Each individual head is trained for one specific task, like detecting objects, classifying road signs, or segmenting roads to detect drivable space⁽¹⁴⁾.

(a)			
Sensor Fusion Approaches	Descriptions	Strengths	Weaknesses
High-Level Fusion (HLF)	Each sensor carries out detection or tracking algorithm separately and subsequently combines the result into one global decision.	Lower complexity and requires less computational load and communication resources. Further, HLF enables standardizing the interface towards the fusion algorithm and does not necessitate an in-depth understanding of the signal processing algorithms involved.	Provides inadequate information as classifications with a lower confidence value are discarded. Furthermore, fine-tuning the fusion algorithms has a negligible impact on the data accuracy or latency.
Low-Level Fusion (LLF)	Sensor data are integrated at the lowest level of abstraction (raw data) to be of better quality and more informative.	Sensor information is retained and provides more accurate data (a lower signal-to-noise ratio) than the individual sensors operating independently. As a result, it has the potential to improve the detection accuracy. In addition, LLF reduces latency where the domain controller does not have to wait for the sensor to process the data before acting upon it. This can help to speed up the performance—of particular importance in time-critical systems.	Generates large amount of data that could be an issue in terms of memory or communication bandwidth. Further, LLF requires precise calibration of sensors to accurately fuse their perceptions and it may pose a challenge to handle incomplete measurements. Although multi-source data can be fused to the maximum extent, there is data redundancy, which results in low fusion efficiency.
Mid-Level Fusion (MLF)	Extracts contextual descriptions or features from each sensor data (raw measurements) and subsequently fuses the features from each sensor to produce a fused signal for further processing.	Generates small information spaces and requires less computation load than LLF approaches. Further, MLF approach provides a powerful feature vector and the features selection algorithms that detect corresponding features and features subsets can improve the recognition accuracy.	Requires large training sets to find the most significant feature subset. It requires precise sensor calibration before extracting and fusing the features from each sensor.

Classification based on the Structure:

Once you have selected the sensors that need to be fused, the next step is to learn how to integrate them. **The architecture of sensor fusion is divided into three types**, according to the **method of fusion⁽⁷⁾**:

Centralised Fusion:

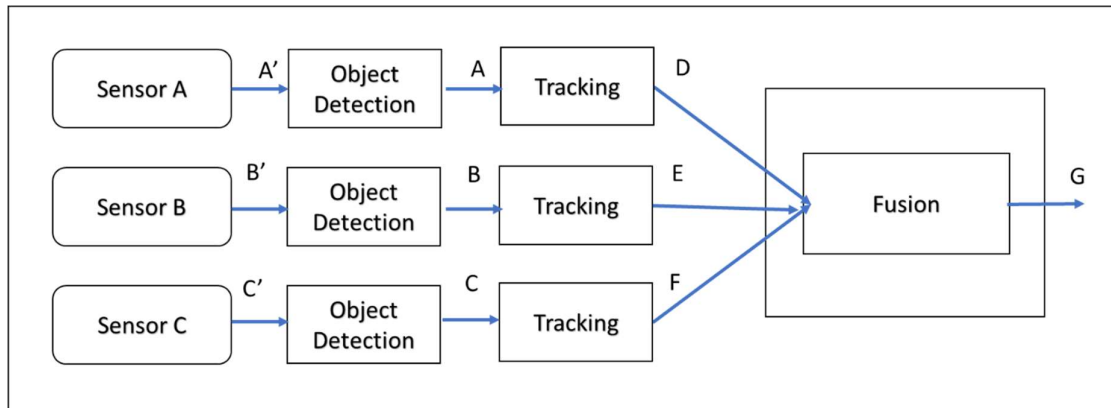
The terminology used in Figure XX is explained below⁽³⁾:

- **A', B', C'**—Raw data from sensor (pixel-level data for camera, point cloud data for LiDAR and ADC Data of RADAR).
- **A, B, C**—Processed Data from sensor object detection blocks. Pre-processing blocks indicate object the detection algorithm implemented for the respective sensors.
- **D**— Temporally and spatially synchronised data from the two sensors.
- **E**—Fused Data. Output of Sensor Fusion. These data are the output of tracking algorithm, and are immune to false negative, false positives and other noise present in sensor data⁽³⁾.

In Centralised Sensor fusion, the fusion and tracking node is built inside the central processor. The fusion block receives synchronised data from various input blocks, which in this case are sensors A, B and C (Camera, Radar, LiDAR respectively). The output of the fusion block is given as the input to the tracker block. The tracker helps in suppressing noise, false positives and false negatives, thereby providing fusion output with least errors⁽³⁾.

It involves sending **the raw data** obtained by each sensor directly to the **central processing unit** for fusion processing. The advantages of this method are **high accuracy and flexible algorithms**. However, due to the large amount of data that needs to be processed, the computing power of the central processing unit is reduced, the requirements for the central unit are higher, and the delay of data transmission needs to be taken into consideration.⁽⁷⁾

Distributed fusion / Decentralised Fusion:



- A', B', C'— Raw data from sensor (pixel-level data for camera, point cloud data for LiDAR and ADC Data of RADAR).
- A, B, C—Data from the sensor object detection blocks. Pre-processing blocks indicate object.
- the detection algorithm implemented for the respective sensors.

- D—Tracking data of Sensor A. This block ensures that data are consistent despite inconsistencies at the output of the pre-processing block.
- E—Tracking data of Sensor B. This block ensures that data are consistent despite inconsistencies at the output of the pre-processing block.
- F—Tracking data of Sensor C. This block ensures that data are consistent despite inconsistencies at the output of the pre-processing block.
- G—Output of the fusion block. Data from all three sensors are spatially and temporally aligned.

In the so-called distributed method, the original data obtained by each sensor is **initially processed in a location closer to the sensor**, and then the results are sent to the central processing unit for information fusion calculation, after which the final results are obtained. This method requires **low communication bandwidth, fast calculation speed, and good reliability**. However, because the original data is filtered and processed, some information will be lost, **so in principle the accuracy of the final results will not be as high as with the centralized method⁽⁷⁾**.

In this method, the fusion node is built inside the central processor; however, the tracking nodes for respective sensors are outside the central processor. The tracker, which is applied to both sensors, independently helps in suppressing false positives, noise, and false negatives for each sensor, thereby providing the central processor with data that are pure and devoid of errors and inconsistencies. As the tracker is applied independently to both sensors, it can be understood that this architecture involves higher processing and is computationally heavier than Centralized fusion. However, as highly consistent data from both sensors are fed to the fusion algorithm, the output of the architecture is highly precise.

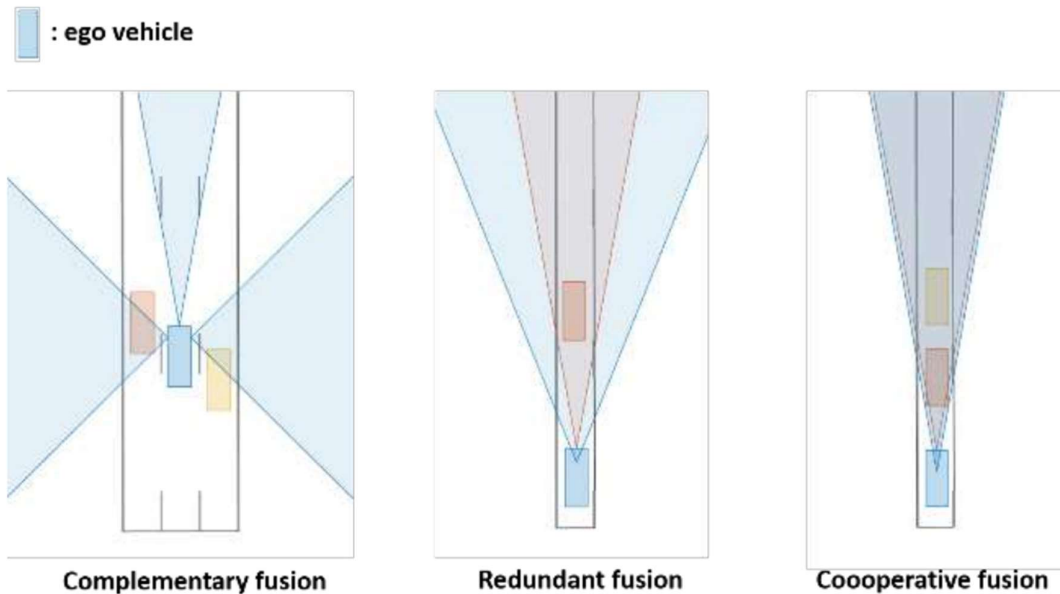
In both these methods, the pre-processing block comprises the respective object detection algorithms. The tracking block is the unscented **Kalman filter** used in both architectures (in a different manner, however). The alignment block takes care of the spatial and temporal alignment of data from the two sensors. The fusion block ultimately associates the data from the two sensors to a single fixed target.

The only difference between the two proposed methods is the way the ‘**tracker**’ block is used. As we shall later see in the experiments and results section, the position of the tracker block significantly affects the algorithm performance. In Decentralized, the tracker is applied on individual sensor data before the data are fused, while in Centralized fusion, the tracker is applied only once on the final fused output

Hybrid fusion: As its name suggests, this method combines the above two methods. In the hybrid method, some sensors use a centralized fusion method, while others use a distributed fusion method. Because it combines the advantages of both centralized and distributed fusion methods, the hybrid fusion framework has strong levels of adaptability and stability, but the overall system structure tends to be more complicated, and it will incur additional data communication and computing processing costs⁽⁷⁾.

There are no set guidelines when it comes to choosing the best strategy and architecture for sensor fusion, as it should always be determined according to specific practical applications. Of course, other factors such as computing power, communications, security and costs must also be considered in choosing the best type of sensor fusion architecture for an application⁽⁷⁾.

Sensor Fusion classification based on relation between the different input sources / Sensor Configuration⁽¹⁷⁾:



Depending on the relative position and orientation of the fixed sensor configuration in a perception architecture, sensor fusion can be classified into **complementary fusion**, **redundant fusion** and **cooperative fusion**.

As shown below in figure xx, the ego vehicle depicted in blue refers to the vehicle on which perception is enabled by placement of sensors in different configurations⁽¹⁷⁾.

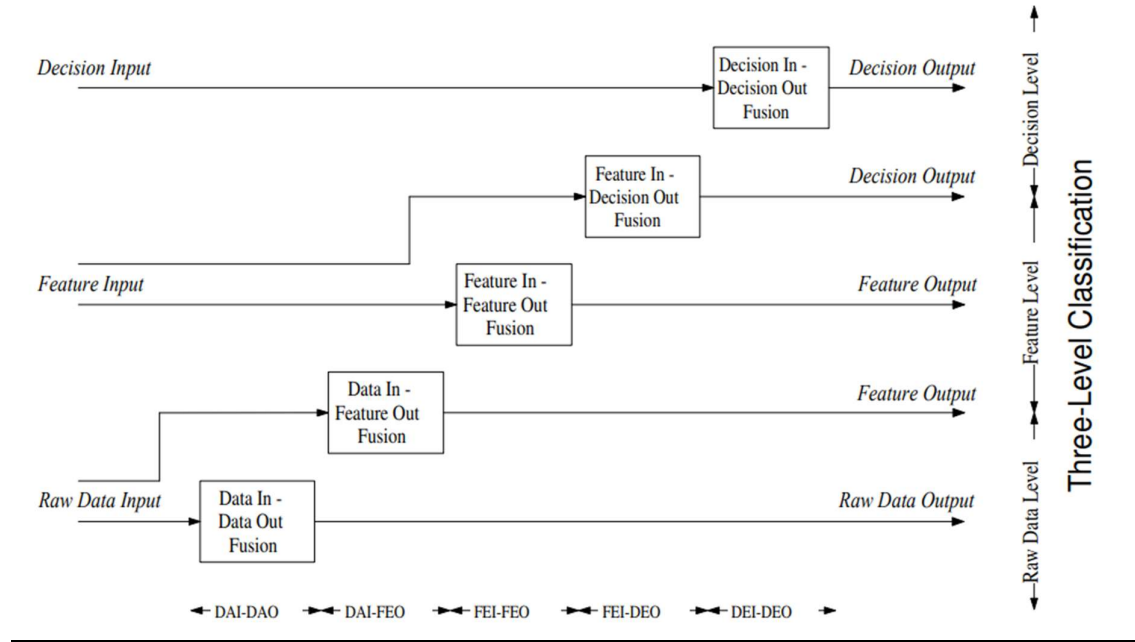
Complementary fusion involves positioning of sensors such that there is no overlap in their field of view resulting in a maximization of coverage of the environment. A sensor configuration is called complementary if the sensors do not directly depend on each other, but can be combined in order to give a more complete image of the phenomenon under observation. This resolves the incompleteness of sensor data. An example for a complementary configuration is the employment of multiple cameras each observing disjunct parts of a room. Generally, fusing complementary data is easy, since the data from independent sensors can be appended to each other⁽¹⁸⁾.

Redundant fusion, also referred to as competitive fusion, position sensors such that they focus on the same field of view ensuring higher reliability and detection accuracy⁽¹⁷⁾. Sensors in a competitive configuration have each sensor delivering independent measurements of the same property. Competitive configurations are used for fault-tolerant and robust systems. An example would be the reduction of noise by combining two overlaying camera images⁽¹⁸⁾.

. **Cooperative fusion** involves sharing a common field of view between 2 sensor modalities such that a new representation of the environment can be created to perform detection or depth estimation more efficiently⁽¹⁷⁾. A **cooperative** sensor network uses the information provided by two independent sensors to derive information that would not be available from the single sensors. An example for a cooperative sensor configuration is stereoscopic vision – by combining two-dimensional images from two cameras at slightly different viewpoints a three-dimensional image of the observed scene is derived.⁽¹⁸⁾

Categorization Based on Input/Output:⁽¹⁸⁾

Dasarathy proposed a refined categorization based on the three-level model. It categorizes fusion processes derived from the abstraction level of the processes' input and output data⁽¹⁸⁾.



The reason for the Dasarathy model was the existence of fusion paradigms where the input and output of the fusion process belong to different levels. Examples are feature selection and extraction, since the processed data comes from the raw data level and the results belong to the feature level. For example, pattern recognition and pattern processing operates between feature and decision level⁽¹⁸⁾.

These ambiguous fusion paradigms sometimes have been assigned according to the level of their input data and sometimes according to the level of their output data. To avoid these categorization problems, Dasarathy extended the three-level view to five fusion categories defined by their input/output characteristics. Figure 2 depicts the relation between the three-level categorization and the Dasarathy model⁽¹⁸⁾.

Sensor Fusion Techniques and Algorithms:

Sensor fusion techniques and algorithms have been extensively studied over the last number of years and now, are well-established in the literature. However, a recent study

revealed that obtaining the **current state-of-the-art fusion techniques and algorithms** is an arduous and **challenging** task due to **multidisciplinary** and variants of proposed fusion algorithms in the literature. The study classified these **techniques and algorithms** into⁽²⁾

- **classical sensor fusion algorithms and**
- **deep learning sensor fusion algorithms.**

On the one hand, the **classical sensor fusion algorithms**, such as knowledge based methods, statistical methods, probabilistic methods, et cetera, utilize the theories of uncertainty from data imperfections, including inaccuracy and uncertainty to fuse sensor data. It proposes a real-time roundabout detection and navigation system in a road environment utilizing a combination of the proposed “**Laser Simulator**” algorithm to detect objects and the knowledge-based fuzzy logic (FL) algorithm for decision making⁽²⁾.

On the other hand, **the deep learning sensor fusion algorithms** involve generating various multi-layer networks that enable them to process raw data and extract features to perform challenging and intelligent tasks, e.g., object detection in an urban environment for AV. In the AV context, algorithms, such as **Convolutional Neural Network (CNN)** and **Recurrent Neural Network (RNN)** are among the most employed algorithms in perception systems. It proposed an advanced weighted-mean⁽²⁾.

Some of the Popular three types of Object detection Algorithm (Deep learning based method):

1. R-CNN
2. YOLO
3. SSD
4. Voxelnet
5. PointNet

6. ResNet

7. CentreNet

Classification of different algorithms

Algorithm	Approach	Strengths	Weaknesses	Suitability
R-CNN	Two-stage (region proposals & refinement)	High accuracy, large objects	Slow execution, not real-time	High-precision tasks, autonomous driving, robotics
YOLO	Single-stage (direct bounding box & class prediction)	Fast execution, real-time	Lower accuracy, small objects	Video processing, resource-constrained environments
PointNet	Point-based feature extraction	Efficient for large point clouds, robust to noise	Limited expressiveness, not primarily for object detection	Feature extraction, pre-processing, direct detection with variants
CentreNet	Keypoint regression & grouping for bounding boxes	High accuracy for specific objects, mobile devices	Requires good keypoint detection, cluttered scenes	Specific object types (pedestrians, faces), resource-constrained environments
VoxelNet	Voxel-based object detection	Efficient for large point clouds, good accuracy for large objects	Limited resolution for small objects, parameter tuning	Autonomous driving, robotics, 3D scene understanding
ResNet	Deep convolutional neural network with skip connections	High accuracy and performance across tasks, building block for other algorithms	Computationally expensive for deep architectures	Backbone for various object detection algorithms
SSD	Single-stage with different feature maps for object sizes	Faster than R-CNN, balance between speed & accuracy	Might struggle with small objects and clutter	Real-time applications, tasks requiring speed-accuracy balance

R-CNN⁽¹³⁾:

R-CNN is one of the popular two stage detectors in which image is proposed into different regions then CNN is applied for the detection. R-CNN model gives high accuracy but it is a slow process which is not useful for real time application of autonomous vehicle. Then various advancement is seen in R-CNN. Fast R-CNN is proposed to increase the speed in this,

Input image is directly processed using CNN to produce a convolutional feature map. Now a days single stage detector like YOLO, SSD are used which gives faster result. The loss of accuracy is compensated using the sensor fusion technology. It is used widely in autonomous vehicle⁽¹³⁾.

You Only Look Once (YOLO) CNN algorithms to fuse **RGB camera and LiDAR point cloud** data to improve the real-time performance of object detection. YOLO detector was first created in 2016 by and has achieved a significant milestone over the last number of years. It is a single-stage detector that predicts **bounding boxes** and produces **class probabilities** with confidence scores on an image in a single neural network (one evaluation only). The YOLO based model provides fast detection speed of **45 FPS with 59.2% average precision** (AP, an evaluation metric that measures object detection or information retrieval model performances) on the VOC 2007 dataset. Besides, the latest **YOLOv4** released in April 2020, achieves state-of-the-art results at a real-time speed on the MS COCO dataset of approximately **65 FPS with 43.5% AP** (and 65.7% AP50—IoU above 50%) on an NVIDIA® Tesla® V100 Graphical Processing Unit (GPU). It is proposed a CNN-based method to detect aggressive driving behaviors through emotions using near-infrared light and thermal cameras. They conducted **score-level fusion** using the CNN output scores from near-infrared light images and thermal images to improve the detection accuracy. Their proposed method achieved a **high classification accuracy of emotions and demonstrated that their proposed technique achieved better performance than the conventional methods** for emotion detection⁽²⁾.

In addition, with the advent of **3D sensors** and diverse applications for understanding the 3D environment of the surrounding AV, there is an increased research focus on 3D object detection. It leverages their previously proposed VoxelNet framework and presented two feature-level fusion approaches called PointFusion and VoxelFusion to combine the RGB and

point cloud data for 3D object detection. VoxelNet is a generic 3D object detection network that unifies feature extraction and bounding box prediction processes into a single stage, end-to-end trainable deep network⁽²⁾.

The Point Fusion method uses the known calibration matrix to project 3D points onto the image and, subsequently extracts image features from a pre-trained 2D CNN and concatenate them at the point level. Subsequently, they leveraged the VoxelNet architecture to process the concatenated features and the corresponding points jointly. In contrast, the VoxelFusion method projects the non-empty 3D voxels created by the VoxelNet onto the image and extract features within the 2D ROIs and consequently concatenates the pooled image features at the voxel level. Point Fusion framework that leverages the image data and raw point cloud data for 3D object detection. They utilized the CNN and PointNet architectures to process the image and point cloud independently and subsequently combine the resulting outputs to predict multiple 3D box hypothesis and their corresponding confidences⁽²⁾.

The **PointNet** architecture is a novel neural network that provides a unified architecture for applications ranging from 3D classification to scene semantic parsing for processing raw point cloud data⁽²⁾.

Other deep learning-based sensor fusion algorithms, to name a few, include: _

- **ResNet**, or Residual Networks, is a residual learning framework that facilitates deep networks training⁽²⁾.
- **SSD, or Single-Shot Multibox Detector**, is a method that discretizes bounding boxes into a set of boxes with different sizes and aspect ratios per feature map location to detect objects with variant sizes—it overcomes the limitation of YOLO small and variant-scale object detection accuracy⁽²⁾.

- **CenterNet** represents the state-of-the-art monocular camera 3D object detection algorithm, which leverages key-point estimation to find center points of bounding boxes and regresses the center points to all other object properties, including size, 3D location, orientation, and pose⁽²⁾.

Table 8a below summarizes the strengths and weaknesses of the sensor fusion approaches:

- HLF, LLF, and MLF, and presents an overview of the sensor fusion techniques and algorithms for obstacle detection, namely YOLO, SSD, VoxelNet, and PointNet, in Table 8b.

(b)		
Algorithms	Descriptions	Advantages and Drawbacks
YOLO	You Only Look Once (YOLO) is a single-stage detector, which predicts bounding boxes and produces class probabilities with confidence scores on an image in a single CNN ¹ .	<ul style="list-style-type: none"> - Provides real-time detections. - Less accurate than SSD. - Poor detection of dense obstacles, e.g., flocks of birds, because each grid can propose only 2 bounding boxes. - Poor detection of small obstacles. - High localization error.
SSD	Single-Shot Multibox Detector (SSD) is a single-stage CNN detector that discretizes bounding boxes into a set of boxes with different sizes and aspect ratios to detect obstacles with variant sizes.	<ul style="list-style-type: none"> - Provides real-time and accurate obstacle detections. - Pose a challenge to detect smaller obstacles but performs better than YOLO. - Poor extractions of features in shallow layers. - Loss of features in deep layers.
VoxelNet	A generic 3D obstacle detection network that unifies feature extraction and bounding boxes prediction into a single-stage, end-to-end trainable deep network. In other words, VoxelNet is a voxelized method for obstacle detection using point cloud data.	<ul style="list-style-type: none"> - Does not require to extract features manually. - Requires large volume of data and memory for training.
PointNet	Presents a permutation-invariant deep neural network which learns global features from unordered point clouds (two-stage detection).	<ul style="list-style-type: none"> - Able to handle point clouds in any order, e.g., permutation independence on the order of point clouds. - Difficult to generalize to unseen point configurations.

State Estimation / Tracking Technique:

Kalman Filter⁽¹⁷⁾:

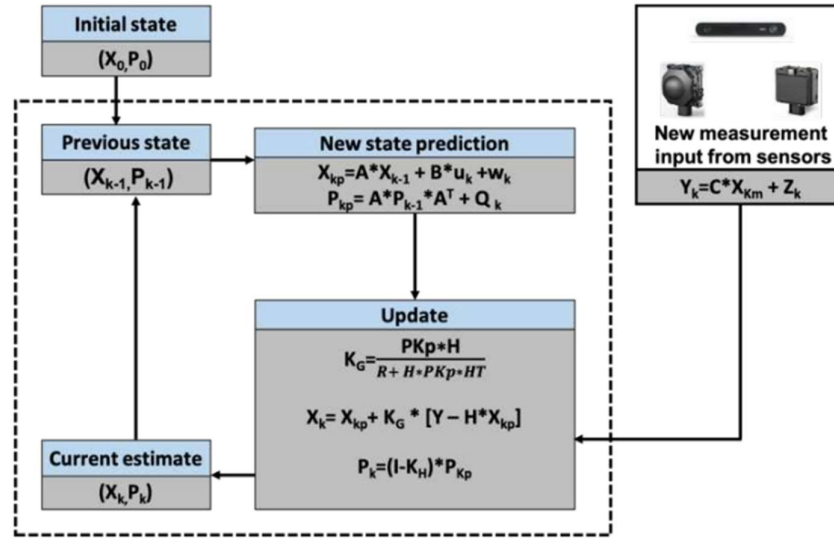


Figure xx – working of Kalman filter

Sensor fusion can also be classified according to the algorithm used for state estimation/tracking. The Kalman filter proposed by Rudolf E. Kalman in 1960 is the most widely used state estimation algorithm in autonomous driving and robotics applications for tracking targets. The Kalman filter family are a set of recursive mathematical equations that provide an efficient computational solution of the least-squares method for estimation. They support estimations of past, present and future states, even when the precise nature of the modeled system is unknown ⁽¹⁷⁾.

The figure xx above shows the general working of filters in the Kalman family and explains the important parameters and computations involved in the new state prediction and update steps. The parameters critical to functioning of the Kalman filter listed in the above

figure 3 are: the state covariance matrix (P) which is a measure of the error in estimation, the measurement covariance matrix (R) which is a measure of error in measurement, the process noise covariance matrix (Q) inherent to the model of the system and the KG (Kalman gain) which is a weight factor that compares which error to trust more between: error in the estimate versus error in measurement⁽¹⁷⁾.

Due to the fast convergence property of Kalman filters, the initial state (X_0, P_0) can be selected at random and does not affect the final estimate. A , B and u used in the prediction step consists of velocity and accelerations constants, which are derived from Newton's equation of motion. These are used to compute the new state (X_{kp}) as well as the new error in estimation (P_{kp}) of the state covariance matrix. Next, in the update step the Kalman gain KG is computed as the fraction of error in estimate over the total error (total error = sum of error in measurement and error in estimate). The value of KG lies in the range 0 to 1 and is the deciding factor for which of the errors to trust more in computation of updated state X_k . If the computed value of KG is close to 0 then the estimates are stable and measurements are inaccurate, where as if KG is closer to 1 then the measurements are accurate and the estimations are unstable. Since, the term $[Y - H * X_{kp}]$ represents the deviation of prediction at a previous time step from the new sensor reading at the current time step, KG determines what fraction of this difference is added as a correctional factor to update X_{kp} to X_k in the update step⁽¹⁷⁾.

The Kalman filter has the ability to obtain optimal statistical estimations when the system state is described as a linear model and the error can be modeled as Gaussian noise. If the system state is represented as a nonlinear dynamic model as opposed to a linear model, a modified version of the Kalman filter known as the Extended Kalman Filter (EKF) can be used, which provides an optimal approach for implementing nonlinear recursive filters. However, the EKF has disadvantages due to the computation of the Jacobian (matrix describing the state

of the system) being computationally expensive. Further, any attempts to reduce the cost through techniques like linearization make the performance unstable. The Unscented Kalman filter (UKF) has gained popularity, due to its ability to be implemented in parallel, as well as the absence of linearization step and associated errors of the EKF. In contrast to the linearization strategy used by the EKF, the UKF uses a sampling strategy to establish the minimum set of points around the mean referred to as sigma points. These points are propagated through nonlinear functions and the covariance of the estimations can be regained ⁽¹⁷⁾.

Difficulties in Sensor Fusion Implementation⁽¹⁰⁾:

Sensor Heterogeneity: Different sensors have distinct characteristics, such as measurement noise, field of view, and sampling rates. Integrating data from heterogeneous sensors requires careful calibration, synchronization, and alignment of the sensor outputs. Handling the discrepancies between sensors can be challenging and often requires sophisticated algorithms⁽¹⁰⁾.

Data Association and Fusion Algorithms: Merging data from multiple sensors involves associating sensor measurements with the same objects in the environment. This process, known as data association, becomes complex in situations with occlusions, cluttered scenes, or when multiple objects are close to each other. Developing robust data association algorithms that handle various scenarios is a significant challenge⁽¹⁰⁾.

Computational Complexity: Sensor fusion algorithms often require significant computational resources to process and fuse large amounts of data in real-time. Embedded systems in vehicles have limited computational capabilities and power constraints. Designing efficient algorithms that can run on resource-constrained hardware platforms without compromising safety is a difficult task⁽¹⁰⁾.

Sensor Failures and Fault Tolerance: Sensor failures or malfunctions can occur in automotive systems due to various reasons, such as environmental conditions, physical damage, or electrical faults. Ensuring fault tolerance and graceful degradation in sensor fusion systems is crucial. Implementing strategies to detect sensor failures, switch to redundant sensors, or rely on fallback mechanisms becomes essential for maintaining system reliability⁽¹⁰⁾.

How to Overcome the Difficulties in Sensor Fusion Implementation:

Overcoming the difficulties in sensor fusion implementation requires a combination of technical approaches, algorithmic advancements, and careful system design. Here are some strategies to address the challenges⁽¹⁰⁾:

Sensor Selection and Calibration: Carefully choose sensors with complementary capabilities that align with the specific requirements of the application. Ensure accurate calibration of sensors, considering factors such as intrinsic/extrinsic parameters, environmental conditions, and degradation over time. Calibration algorithms and techniques should be developed to minimize errors and align the sensor data accurately⁽¹⁰⁾.

Data Association Algorithms: Develop robust data association algorithms that can handle complex scenarios such as occlusions, cluttered scenes, and multiple objects in close proximity. Utilize techniques such as track matching, feature matching, or probabilistic methods like the Joint Probabilistic Data Association Filter (JPDAF) or Multiple Hypothesis Tracking (MHT) to associate sensor measurements with the correct objects⁽¹⁰⁾.

Fusion Algorithms and Models: Employ advanced fusion algorithms and models that can effectively combine data from multiple sensors. Techniques such as Kalman filters, particle

filters, or Bayesian networks can be used to fuse the sensor measurements and provide accurate and reliable estimates. Consider the strengths and limitations of each algorithm and select the most suitable one based on the specific requirements of the system⁽¹⁰⁾.

Computational Efficiency: Optimize sensor fusion algorithms to ensure computational efficiency, especially considering the resource constraints of embedded systems in vehicles. Utilize techniques like parallel processing, algorithmic simplification, or hardware accelerators to enhance computational performance and reduce processing time while maintaining accuracy⁽¹⁰⁾.

Redundancy and Fault Tolerance: Design the sensor fusion system to handle sensor failures and faults effectively. Implement redundancy by incorporating multiple sensors that can provide overlapping information. Develop fault detection and isolation mechanisms to identify sensor failures and switch to alternate sensors or activate fallback strategies. Redundancy and fault tolerance enhance system reliability and maintain safety even in the presence of sensor failures⁽¹⁰⁾.

Validation and Testing: Rigorously test the sensor fusion system using real-world scenarios and simulated environments. Use comprehensive datasets that cover a wide range of scenarios to evaluate the performance and robustness of the system. Conduct thorough validation and verification processes to ensure that the sensor fusion system meets the desired requirements for safety, accuracy, and reliability⁽¹⁰⁾.

Continuous Improvement: Sensor fusion algorithms and implementations should be continuously improved and updated as new technologies and techniques emerge. Stay updated with the latest research and advancements in sensor fusion, computer vision, machine learning,

and other relevant fields. Regularly monitor the performance of the system and apply updates or refinements as needed to enhance performance and address any shortcomings⁽¹⁰⁾.

By adopting these strategies, sensor fusion implementation can be improved, leading to more accurate and reliable perception of the environment, enhanced vehicle safety, and improved autonomy in automotive embedded systems⁽¹⁰⁾.

Importance of Validation:

It is also essential to validate fusion to ensure software quality and scenario covers. However, validation can lead to millions of scenarios, so it is always important to use the right validation strategy to test fusion with Simulated data, real-world data, or vehicle testing⁽¹⁾.

So fusion is a critical perception component for AD performance. One must consider key practical aspects and define the right strategy for design and validation to achieve the highest AD software maturity level⁽¹⁾.

At KPIT, we have been working on fusion for the past seven years. We have invested and developed a very robust **design and validation strategy** to ensure the highest level of **software quality**. The design takes care of variant handling for multi-sensor fusion (radar+radar, Camera+Radar, Camera+Radar+Lidar), different sensor topology and layout, sensor characteristics, and sensor degradation. We have filed multiple patents for various fusion techniques. We have been supporting multiple **OEMs and Tier-1 customers** for various fusion projects for different model year programs⁽¹⁾.

Validation of Sensor Fusion⁽⁹⁾:

It is also important to validate fusion to **ensure software quality and scenario covers**. However, Validation can lead **to millions of scenarios**, so it is always important to **use right validation strategy** to test fusion with Simulated data, real world data or in vehicle testing.

Following are the key aspect for the **Fusion validation**

- **Edge case Scenario Selection:** As fusion improve the perception improve performance by fusing various sensor noisy data, so scenarios shall be carefully selected which covers all edge cases. The scenario shall also mention which level of validation is required whether at simulation, real world or in vehicle. The objective shall be maximum coverage at simulation level to reduce cost and time at the same time there **shall not be any compromise in quality of test**⁽⁹⁾.
- **Sensor modelling:** To validate fusion in simulation with environment **noise effect high fidelity sensor model** is required which mimic environment effect on sensor performance. The high-fidelity sensor modelling is research topic and there are various techniques are proposing like data driven model or physics-based modelling. The right sensor model is very critical for simulation⁽⁹⁾.
- **Validation with real world data:** The **high-fidelity sensor model** can model environment conditions up to **90 to 95% accuracy**, so all scenarios are not possible to cover in simulation and fusion need to be validated with real world data.
- **Vehicle testing:** The final validation of fusion shall be on vehicle to ensure end to end testing at feature level considering actual conditions, **sensor latency and actuation delay**⁽⁹⁾

So fusion is very critical perception component for AD performance, and one must consider key practical aspect and define right strategy for design as well as for validation to achieve highest level of maturity of AD software. ⁽⁹⁾

Future Trends

Regardless of which sensor fusion architecture is used, sensor fusion is largely dependent on software and the main challenges arise from the algorithm. Hence the development of efficient algorithms based on actual applications has become the top priority of sensor fusion development.⁽⁷⁾

In terms of algorithm optimization, the introduction of artificial intelligence has made an ongoing impact on sensor fusion development. By using artificial neural networks (ANNs), software can imitate the judgment and decision-making processes of the human brain. ANNs can continuously learn and evolve, allowing the software to keep developing, thus accelerating the development of sensor fusion⁽⁷⁾.

Although software is critical for the sensor fusion process, hardware also plays a crucial role. For example, if all sensor fusion algorithm processing is done by the main processor, the load on the processor will be extremely heavy. Therefore, a popular approach in recent years has been to introduce a sensor hub. A sensor hub can independently process a sensor's data outside the main processor. This can reduce the load on the main processor, and also reduce the system power consumption by lowering the working time of the main processor. This is necessary in power-sensitive applications such as wearables and the Internet of Things (IoT)⁽⁷⁾.

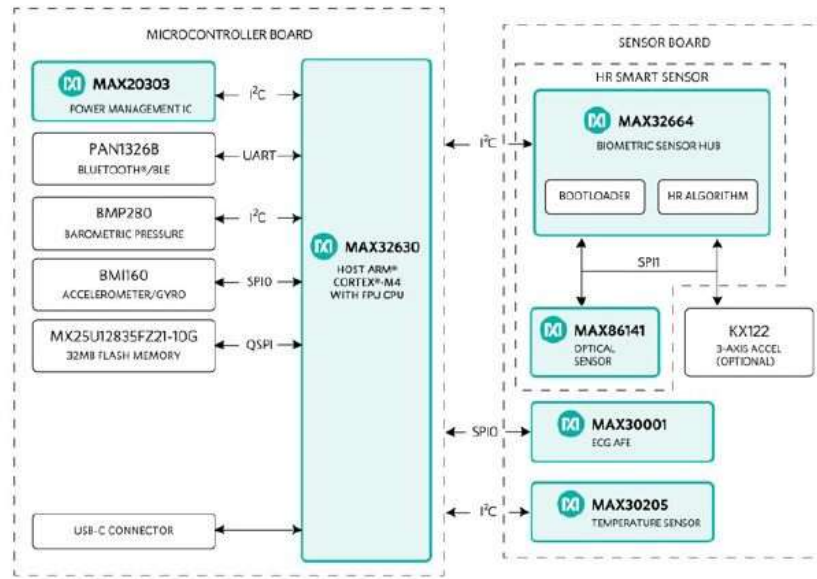


Figure 2. Example of a sensor hub: In this health wearable sensor system, MAX32664 is used as the sensor hub to perform fusion processing on the data information of the optical and motion sensors. (Image source: Maxim Integrated) ⁽⁷⁾

Aptiv's approach:

Low-level sensor fusion:

Of course, the more sensors on a vehicle, the more challenging fusion becomes, but also the more opportunity exists to improve performance. To tap into these benefits, Aptiv uses a technique called low-level sensor fusion⁽⁵⁾.

In the past, the processing power to analyze sensor data to determine and track objects has been packaged with the cameras or radars. With Aptiv's Satellite Architecture approach, the processing power is centralized into a more powerful active safety domain controller, allowing for low-level sensor data to be collected from each sensor and fused in the domain controller⁽⁵⁾.

Moving the processing to a domain controller results in sensors that take up less volume and less mass — up to 30 percent less. For comparison, the footprint of a camera is reduced from the size of a deck of playing cards to the size of a pack of chewing gum. By keeping sensors as small as possible, OEMs have more options in vehicle packaging⁽⁵⁾.

Another benefit is increased data sharing. With traditional systems, smart sensors process environmental inputs independently, which means any decisions made when using the information are only as good as what that individual sensor can see. However, with Satellite Architecture, where all the data coming from the sensors is shared centrally, there is more opportunity for active safety applications in the domain controller to make use of it. Aptiv can even apply artificial intelligence (AI) tools to extract useful information that would otherwise be discarded. The right AI can learn from it, and that helps us solve challenging corner cases our customers face⁽⁵⁾.

A third benefit of low-level sensor fusion is reduced latency. The domain controller doesn't have to wait for the sensor to process data before acting upon it. This can help speed performance in situations where even fractions of a second count⁽⁵⁾.

More data leads to better decisions. By embracing a vehicle architecture that allows for a high number of sensors and then synthesizes the data through sensor fusion, vehicles can become smarter, faster⁽⁵⁾.

Visteon Approach – Raw data Fusion⁽⁸⁾

Visteon is currently working on a signal-based multi-sensor data fusion approach. In this centralized architecture, sensor data from different sources are fused using AI techniques at a raw signal level. The major advantage of this approach is the availability of *complete* environmental information to the system which, with the help of machine learning techniques, will make informed decisions right from the start of a journey and, subsequently, be able to safely direct the car⁽⁸⁾.

The safe and successful introduction of autonomous driving depends on a robust solution to sensor data fusion – creating a “brain” for the car’s automated body that is able to process huge data sets, apply AI and rapidly transmit information to and from its environment. Visteon’s approach is based on integrated and centralized multi-sensor fusion technology that is built on fault-tolerant hardware and incorporates AI for the most accurate levels of object detection and classification⁽⁸⁾.

References:

1. <https://www.kpit.com/insights/achieve-sensor-fusion-for-autonomous-driving-adas-vehicles/>
2. <file:///C:/Creamcollar/Creamcollar/ADAS/ACC/Sensor%20fusion/sensor%20fusion.pdf>
3. [Centralised and Decentralised Sensor Fusion.pdf](#)
4. <https://in.mathworks.com/discovery/kalman-filter.html>
5. <https://www.apiv.com/en/insights/article/what-is-sensor-fusion>
6. file:///C:/Creamcollar/Creamcollar/ADAS/ACC/Sensor%20fusion/ODD_description_methods_for_automated_driving_vehi.pdf
7. <https://www.avnet.com/wps/portal/apac/resources/article/what-is-sensor-fusion/>
8. <https://www.visteon.com/current-sensor-data-fusion-architectures-visteons-approach/>
9. <https://www.motorindiaonline.in/how-to-achieve-the-best-sensor-fusion-for-adas-av-vehicles/>
10. <https://vlsifirst.com/blog/sensor-fusion-in-automotive-embedded-systems>
11. [Multisensor Data Fusion Strategies for.pdf](#)
12. <https://www.wevolver.com/article/what-is-sensor-fusion-everything-you-need-to-know>
13. <https://www.telematicswire.net/sensor-fusion-for-autonomous-vehicle/>
14. <https://cariad.technology/de/en/news/stories/sensor-fusion-introduction.html>
15. <https://www.pathpartnertech.com/sensor-fusion-for-l3-and-beyond/>
16. [Adaptive Cruise Control and Multi sensor Fusion.pdf](#)
17. [IMPORTANT - PERCEPTION ARCHITECTURE EXPLORATION FOR.pdf](#)
18. [An Introduction to Sensor Fusion.pdf](#)