# Switching Trackers for Effective Sensor Fusion in Advanced Driver Assistance Systems

**Deo, A. & Palade, V.**



DOI 10.3390/electronics11213586
ESSN 2079-9292

Publisher: MDPI

*electronics*

MDPI

*Article*

# Switching Trackers for Effective Sensor Fusion in Advanced Driver Assistance Systems

**Ankur Deo [1,2] and Vasile Palade [2,*]**

[1] Department of Autonomous Driving, KPIT Technologies, Pune 411057, India
[2] Centre for Computer Science and Mathematical Modelling, Coventry University, Priory Road, Coventry CV1 5FB, UK
* Correspondence: ab5839@coventry.ac.uk

**Abstract:** Modern cars utilise Advanced Driver Assistance Systems (ADAS) in several ways. In ADAS, the use of multiple sensors to gauge the environment surrounding the ego-vehicle offers numerous advantages, as fusing information from more than one sensor helps to provide highly reliable and error-free data. The fused data is typically then fed to a tracker algorithm, which helps to reduce noise and compensate for situations when received sensor data is temporarily absent or spurious, or to counter the offhand false positives and negatives. The performances of these constituent algorithms vary vastly under different scenarios. In this paper, we focus on the variation in the performance of tracker algorithms in sensor fusion due to the alteration in external conditions in different scenarios, and on the methods for countering that variation. We introduce a sensor fusion architecture, where the tracking algorithm is spontaneously switched to achieve the utmost performance under all scenarios. By employing a Real-time Traffic Density Estimation (RTDE) technique, we may understand whether the ego-vehicle is currently in dense or sparse traffic conditions. A highly dense traffic (or congested traffic) condition would mean that external circumstances are non-linear; similarly, sparse traffic conditions would mean that the probability of linear external conditions would be higher. We also employ a Traffic Sign Recognition (TSR) algorithm, which is able to monitor for construction zones, junctions, schools, and pedestrian crossings, thereby identifying areas which have a high probability of spontaneous, on-road occurrences. Based on the results received from the RTDE and TSR algorithms, we construct a logic which switches the tracker of the fusion architecture between an Extended Kalman Filter (for linear external scenarios) and an Unscented Kalman Filter (for non-linear scenarios). This ensures that the fusion model always uses the tracker that is best suited for its current needs, thereby yielding consistent accuracy across multiple external scenarios, compared to the fusion models that employ a fixed single tracker.

**Keywords:** ADAS; sensor fusion; tracking; real time traffic density estimation; traffic sign recognition; Unscented Kalman filter; Extended Kalman filter



**Citation:** Deo, A.; Palade, V. Switching Trackers for Effective Sensor Fusion in Advanced Driver Assistance Systems. *Electronics* **2022**, *11*, 3586. https://doi.org/10.3390/electronics11213586

Academic Editor: Felipe Jiménez

Received: 29 September 2022
Accepted: 30 October 2022
Published: 3 November 2022



## 1. Introduction

An accurate and robust environment detection model is essential for autonomous vehicle driving. To construct such a model, usually a myriad of sensors—such as LiDAR (Light Detection and Ranging), ultrasonic sensors, RaDAR (Radio Detection and Ranging), and cameras—are used [1]. Different sensors have their own pros and cons. For example, a RaDAR sensor may detect the presence of an object as far as 100 metres away and determine its speed relative to the ego-vehicle. However, while RaDAR excels at object detection, object classification is a difficult task [2]. A camera sensor is well-suited where object classification is important, such as when reading traffic sign and lane information for the purpose of driving algorithms such as Traffic Sign Recognition and Lane Keep Assist. A camera may also be used to complement the RaDAR for features such as Front Collision Detection [1,2]. However, in the circumstances of low light and bad weather

(rainy, snowy, and foggy environments), a camera is typically limited in its performance. In such cases, a LiDAR sensor proves beneficial as it is immune to ambient light and captures a larger range [3]. However, a LiDAR cannot detect visual colours; therefore, while it may detect objects and their distance from the ego-vehicle seamlessly under most circumstances, object classification is better achieved with a camera [1,3]. Thus, all automotive sensors possess certain associated advantages and disadvantages. Sensor fusion techniques, on the other hand, leverage the output provided by multiple sensors; as a result, they are more reliable and error-free, compared to the techniques that employ a single sensor to gauge the environment [2,3].

Several studies on the topic of multi-object sensor fusion and tracking are available in the public domain [2,4,5]. In a typical target-level sensor fusion architecture, all target objects are independently tracked once the data from multiple sensors is associated or fused [6]. This step, as shown in Figure 1, is required to obtain continuity and consistency in the detected objects.

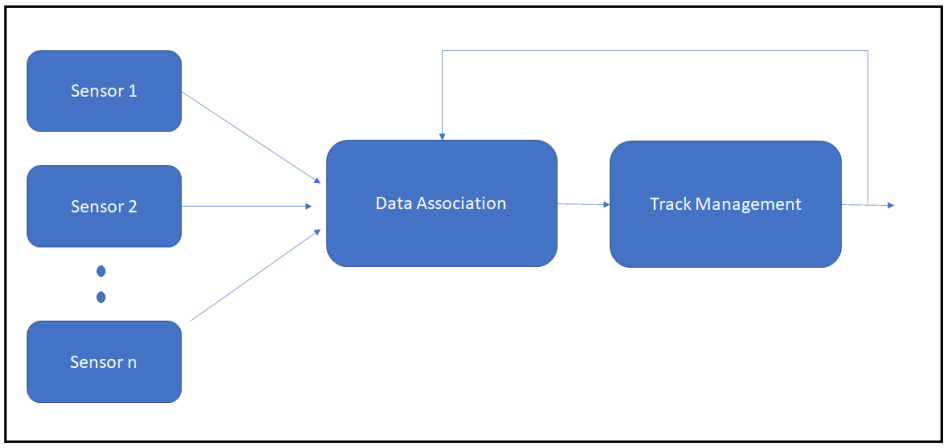


**Figure 1.** Target-level fusion architecture [5,7].

Overall, there are a variety of algorithms, present in today's technological world, that may be used for object detection, fusion, and tracking. While there are multiple fusion architectures, for our work, we follow a target-level sensor fusion architecture (also known as centralised fusion architecture) for ease of application and experimentation, as described in the work presented by Deo et al. [5]. There are multiple works which provide data regarding which tracking algorithms work best under specific external scenarios. For example, for a reliable object detection and fusion algorithm, while the Extended Kalman Filter works best for linear scenarios (where the change of the position of the detected object relative to the ego-vehicle is slow, such as in the sparse traffic on highways and long stretches of road), the Unscented Kalman Filter works best for highly dynamic scenarios (where there is a constant influx and outflux of detected road objects, and where these objects are moving randomly and quite quickly relative to the ego-vehicle, such as in urban junctions) [6]. In any vehicular journey, both scenarios are equally likely to occur—sometimes even in a short span of time. In fact, in standard vehicle-in-the-loop test cycles, both scenarios form critical constituents [7]. It may thus be seen that a fusion architecture cannot favour one kind of scenario over the other and must be designed with a variation of scenarios in mind. Hence, it is a necessity that a sensible sensor fusion architecture takes advantage of more than one tracking algorithm, by efficiently switching between multiple trackers on a real-time basis, to provide an accurate output under all scenarios. Considering this problem statement, in this paper, we build a logic which switches between the Extended Kalman Filter and the Unscented Kalman Filter, based on the real-world traffic density surrounding the ego-vehicle, and the ego-vehicle's type of environment (such as construction zone, pedestrian crossing, junctions, etc.) according to the data derived by the Real-time Traffic Density Estimation (RTDE) and Traffic Sign Recognition (TSR) algorithms. We use the KITTI dataset for all experimental validations of the proposed method.

In this paper, we first introduced the topic and the problem statement in Section 1. In Section 2, background works and research in key related areas are presented. Then, the sensor fusion architecture with the tracker switching technique using Real-time Traffic Density Estimation (RTDE) and Traffic Sign Recognition (TSR) for obtaining optimum performance is introduced and analysed in Section 3. The conclusion is presented in Section 4.

## 2. Material and Methods

In this paper, we focus on the technique to switch between different tracking algorithms in real-time to achieve optimum performance from the fusion architecture under all scenarios. The fusion architecture is considered to consist of three sensors—camera, RaDAR, and LiDAR. In order to switch between multiple tracking algorithms, we use a logic employing the TSR and the RTDE. Accordingly, first, we discuss the background research and details in the field of camera, RaDAR, and LiDAR object detection and data fusion; we then move our focus to the EKF and the UKF as trackers. We focus on their properties, advantages, and disadvantages. Later, we discuss the work performed in the fields of the TSR and RTDE techniques.

### *2.1. Dataset*

The KITTI dataset was used in this work for camera, LiDAR, and RaDAR sensor data experimentation and analysis. This dataset is used for evaluating the performances of the detection and tracking algorithms. The KITTI dataset, available at: http://www.cvlibs.net/datasets/kitti/ (as accessed on 22 August 2022), consists of data that is temporally synchronised by default. Hence, the only care we are required to take regards the 'spatial synchronisation' of sensor data. After the data is synchronised spatially by using transformation matrices, a comparison of the output of the fusion block with the KITTI's ground-truth data suggests the accuracy of the algorithm under test. The KITTI dataset also provides a variety of traffic data for testing under numerous circumstances. The dataset is open source and the granular data may be converted to 'rosbag', thereby simplifying the importation of the data in a Linux-supported ROS-based environment. The dataset also provides support for importing data in the form of MATLAB utility functions; this makes it easy for the simulation and validation of algorithms. Due to these advantages, the KITTI dataset was chosen for this work over other available datasets.

### *2.2. Camera, RaDAR and LiDAR Sensor Fusion*

Different object detection algorithms that are available possess their own strengths and weaknesses. When any algorithm is to be executed on an embedded platform, it is important to evaluate its time and space complexity to understand the performance [6,8]. While some algorithms excel in the speed of execution, others might lead to a more accurate result. It is, therefore, of utmost importance to achieve a fine balance between the speed of execution, the accuracy of detection, and the computational complexity of the algorithm [8]. In this paper, for the purpose of gauging the performance of the tracker algorithm under multiple scenarios, we have used some standard off-the-shelf LiDAR, camera, and RaDAR detection algorithms, namely, YOLOv4 for camera object detection as implemented by Kumar et al. [9], a RaDAR based object detection methodology as worked upon by Manjunath et al. [10], and DBSCAN (Density-Based Spatial Clustering of Applications with Noise) for LiDAR object detection, which is similar to that developed by Deng et al. [11]. As explained earlier in Section 1, we shall use the target-level sensor fusion model, as worked upon by Deo et al. [5], for fusing the sensor data.

#### 2.2.1. Camera Object Detection—YOLOv4

There are several popular techniques for object detection in images and videos, such as the Single Shot Detector (SSD), the techniques based on convolutional neural networks (Fast R-CNN, Faster R-CNN, etc.), Histogram of Gradients (HOG), You Only Look Once (YOLO),

and the region-based convolutional neural network-based methods. It must be noted, however, that compared to other object detection algorithms, YOLO is a faster and simpler algorithm, providing a high speed of execution and demanding less computational prowess. This leads to a good performance of the object detection algorithm whilst minimising the cost of the hardware upon which it is executed. As a result, we selected the YOLO based algorithms for this work.

For camera object detection, we used the YOLOv4 algorithm, which is an improved version of the previously well-optimised YOLOv3 [9]. The goal was to improve the execution speed alongside the optimised performance compared to YOLOv3, which ensures that a single GPU should be sufficient for training an object detector with real-time high accuracy and performance. The YOLOv4 algorithm for object detection provides the necessary freedom of configuration; at the same time, it is computationally lighter than other algorithms like YOLOv5, while still leading to acceptable results [12]. As a result, it continues to be the most widely used object detection algorithm [9]. In this paper, we use the YOLOv4 algorithm as worked upon by Kumar et al. [9], which is described in Figure 2.

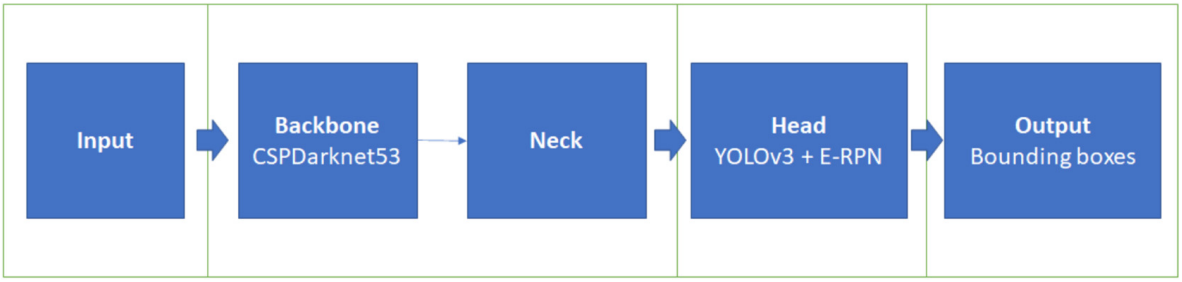


**Figure 2.** Architecture of YOLOv4 algorithm.

#### 2.2.2. LiDAR Object Detection—DBSCAN

In the case of LiDAR, a DBSCAN algorithm is used to detect objects surrounding the ego-vehicle. This algorithm iterates over all LiDAR-generated points in the frame. Typically, clustering LiDAR point-cloud data is extremely useful in several ADAS applications which require real-time object detection and classification. Density-based clustering algorithms work on a principle which groups the target objects based on their density—objects are within high-density regions and separated by low-density regions, which usually consist of noise or objects with which we are not particularly interested [13]. Such a characteristic makes them a suitable candidate for LiDAR point cloud clustering. The DBSCAN algorithm does not require a prior assignment for the number of clusters [13], which is a typical scenario in dynamic environments like self-driving environments. In this algorithm, the software constructs a set of points which are reachable by density from an arbitrary point. This process is repeated for all points in the point cloud. It computes the neighbourhood of a particular set point, and if this neighbourhood contains more than a certain number of points, it is included in the target region [13,14]. All the neighbouring points are made to pass through this process until the algorithm is no longer able to increase the expanse of the cluster. If the considered point is not seen to be an interior one, that is, if there is an absence of enough neighbouring points, it is classified as 'noise'. This mechanism makes the DBSCAN algorithm robust to outlier points [14,15]. In this paper, we use the DBSCAN algorithm for LiDAR object detection as worked upon by Deng et al. [11] and Yabroudi et al. [15]. The DBSCAN process for LiDAR object detection may be seen in Figure 3.

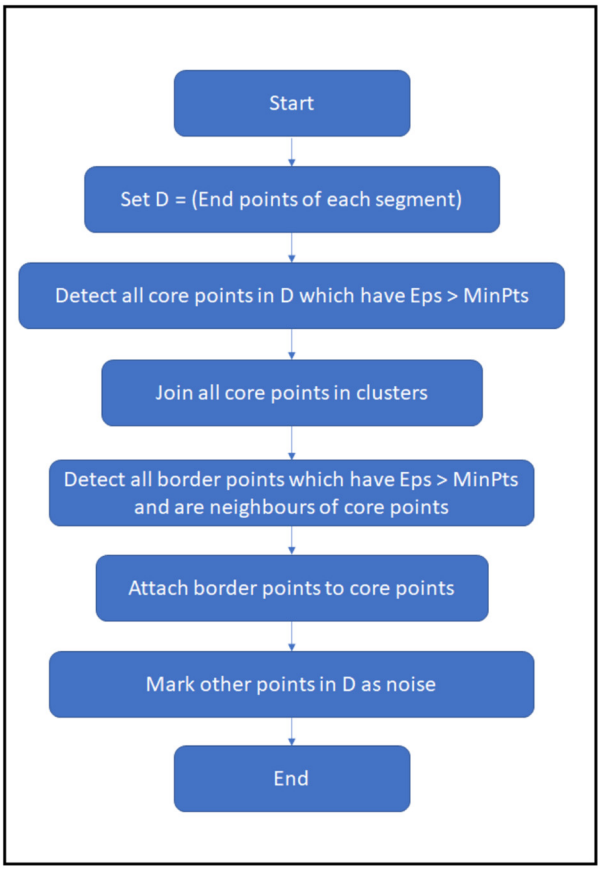


**Figure 3.** DBSCAN-based LiDAR object detection.

In Figure 3, 'Eps' signifies the maximum distance between the points in a cluster, and 'MinPts' signifies the minimum size of points necessary to form a cluster.

#### 2.2.3. RaDAR Object Detection

In typical automotive applications, multiple objects are present in the field of view of the front RaDAR sensor, and the ultimate goal is to detect and track each relevant target object. To address this task, Kellner et al. [16] developed a technique which states the estimation of the presence of target objects using a doppler RaDAR sensor. This technique, however, may only be used for a single sensor. Khalil et al. [17] and Manjunath et al. [10] shed light on a postprocessing architecture that is an evolved version of the algorithm presented by Kellner et al. [16]. This model may be used to detect and track more than one object in the field of view of RaDAR.

In this work, we follow a postprocessing architecture worked upon by Manjunath et al. [10]. The general architecture used to address the task of RaDAR object detection is as stated in Figure 4.

### *2.3. Trackers—EKF and UKF*

Tracking is essential for obtaining the trajectory data of any detected object/objects and for further estimating its/their state in the presence of noise or temporary loss of data. In general, tracking algorithms consist of three main steps: data association, object position prediction, and object track management [18]. Different approaches exist for updating the state of detected object(s); however, the most preferred and easy-to-implement method is the 'Bayesian state estimation' (for example: the Kalman filter and various versions of it, such as the Unscented Kalman filter, the Extended Kalman filter, etc.) [18,19]. The functionality of a Bayesian state estimator may be seen in Figure 5. The object management step takes care of the appearance (object birth) and disappearance (object deletion from

the object list) of the detected object/objects from the target list. The object prediction step estimates the position of an object based on its current and past positions. Once the latest position of an object is available, the data association block calculates the error between the estimated position and the real position of the object. This error value is further used by the tracker to better predict the state of object in the next step.

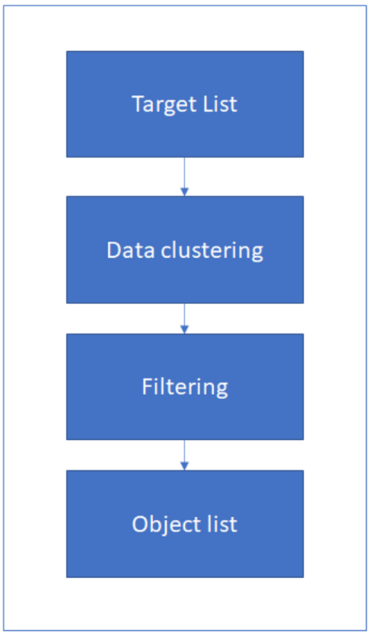


**Figure 4.** Post-processing-based RaDAR object detection.

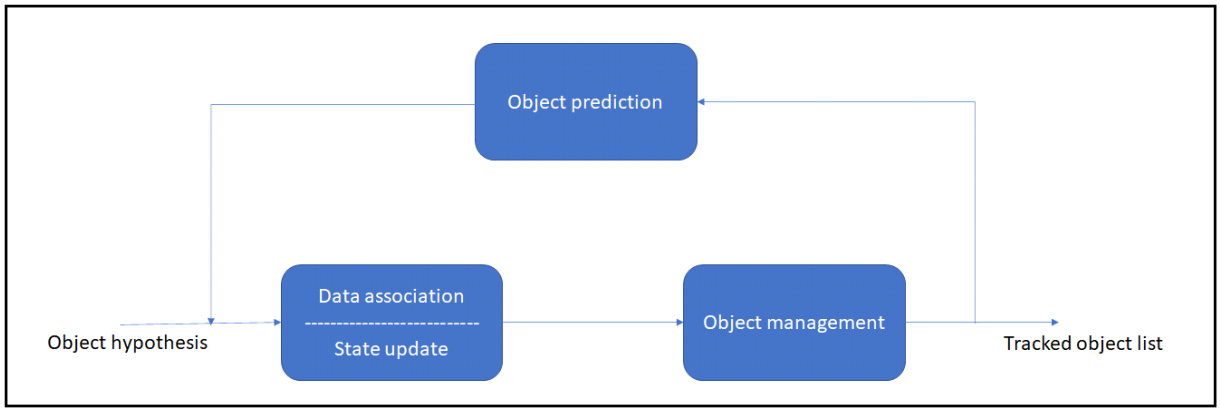


**Figure 5.** Architecture of multi-object tracking.

The basic premise of the Extended Kalman Filter (hereby referred to as the EKF) is as follows: calculate the first-order nonlinear Taylor expansion around the estimated position of the detected target object, and then transform the nonlinear system into a linear equation [19,20]. The EKF is very commonly used in filter systems and the calculation is easy to be implemented. However, Taylor expansion belongs to linear processes; therefore, the results of the EKF will be close to the true value only when the system status and observation equations are close to linear and continuous [21]. In addition, the filtering result is affected by status and measurement noise [20,21]. Furthermore, If the noise covariance matrix is not estimated accurately enough, the cumulative error may lead to the divergence of the filter to an extent where a significant variation of the results from the ground truth data is seen [22]. As a result, the EKF leads to reliable results in ADAS algorithms under linear scenarios only [21,22]. While the latest developments in the EKF do tackle this problem to some extent, the EKF is still fundamentally built to perform best

under linear scenarios [23,24]. Such scenarios include sparse traffic scenarios on highways, where vehicles are moving with constant speed with limited variation of their speed and with their position relative to the ego-vehicle.

On the other hand, the Unscented Kalman Filter (hereby referred to as the UKF), is a method that applies the sampling strategy that is close to nonlinear distribution [23]. It uses the linear Kalman filter framework based on the Unscented Transform (UT) and uses a definite sampling strategy instead of a random sampling strategy [24]. The total number of sampling points in the UKF, also defined as 'Sigma-points', is small and are dependent upon the sampling strategy. The processing steps of the UKF include:

1. Initialise the status vector and status estimation error covariance;
2. Select the sampling points according to the status vector and error covariance, and calculate the weighted value;
3. Calculate the mean and covariance propagation through the equation of status, and update the sampling time by the selected sampling points;
4. Finish the measurement update through a nonlinear observation equation by the selected sampling points;
5. Update the Kalman filter coefficients.

Compared with the EKF, the UKF is more reliably implemented for non-linear scenarios and avoids calculating Hessian and Jacobian matrices [24]. With these benefits, the UKF offers convenience for real-time processing platforms for driving ADAS algorithms under non-linear scenarios. In general, the UKF-based tracking achieves successful tracking and feature mapping in an augmented reality environment [25].

As mentioned in the work by St-Pierre et al. [26], the authors have proven that the UKF's performance is better (albeit with a higher computing-power requirement) than the EKF when used as a fusion method in a positioning module of an integrated navigation information system, which is predominantly a non-linear system. We may further extend their conclusion and use the UKF as a tracker in ADAS algorithms for non-linear external scenarios. From Chen et al. [6], we learn that if an object surrounding an ego-vehicle takes abrupt turns (that is, demonstrates inherently nonlinear behaviour relative to the ego-vehicle), the nonlinear movement cannot be well-handled by the EKF framework. According to St-Pierre et al. [26], an UKF-based tracking algorithm may be used to optimise the nonlinear moving paths of the objects with occlusion. The UKF estimates both the location and velocity information of moving objects, and finally provides more accurate location information compared to the EKF. However, as stated by Wan et al. [27] and St-Pierre et al. [26], the UKF is computationally heavier than the EKF and, as a result, is slower to execute on any hardware platform.

Thus, theoretically, for a highly linear scenario, such as on a highway with sparse traffic, when vehicles are moving at lesser speed relative to each other, an EKF is bound to provide better results; while in an urban scenario with many variations and non-linearities, the UKF shall provide sound and reliable results [28,29]. Our experiments corroborate with these observations. It must be noted that recent developments in predictive filters can provide a more robust immunity against the noise present in systems like inertial navigation systems [29] and several ADAS applications [30]. However, at the same time, the principle that the UKF is more robust to non-linearities than the EKF remains true, as it is proven by recent advancements in UKF-based applications and its fine-tuning for autonomous vehicles, both conventional and otherwise [31,32]. In this work, we use the EKF and the UKF to prove the proposed concept of tracker switching. These trackers may be later replaced by any modified versions, such as the Modified Strong Tracking Unscented Kalman Filter (MSTUKF) [32]. Such techniques have been developed to avoid the loss of accuracy in estimation for scenarios involving the presence of spontaneous noise in the system [33,34]. Another technique to improve the performance of the UKF is also presented by Gao et al. [33], where the adaptive UKF is shown by combining the maximum likelihood principle and the moving horizon estimation technique. Using such techniques eliminates the limitations of the UKF caused by unknown real-time system noises [33,34].

The metric used to evaluate the performance of various Kalman filters is the Overlap Ratio (OR). The OR signifies the final overlap ratio in the percentage of the predicted and real object detection boxes; that is, it is the ratio of the intersection area of the predicted and real boxes to the union of area of the two boxes [21]. The maximum value, in the case of a perfect overlap, is 100%, and the minimum value, in the case of no overlap, is 0%. This value is used to reflect the closeness between the tracking result and the real object, also known as the success rate, which may be expressed by the following Equation (1):

$$\text{OR in \%} = \{[\text{Area (Detected} \cap \text{Ground-truth)}]/[\text{Area(Detected} \cup \text{Ground-truth)}]\} \times 100 \quad (1)$$

Thus, the EKF and the UKF both have their advantages and disadvantages. In our research, we construct an architecture which leverages the advantages of both these trackers, such that the right tracker is chosen by analysing the environment. For this analysis, we use the Traffic Sign Recognition and the Real-time Traffic Density Estimation algorithms, both of which are explained in the next sections.

### *2.4. Traffic Sign Recognition*

According to Fan et al. [35], a typical traffic sign detection algorithm consists of several applications in modern day ADAS. In the TSR algorithm, firstly, candidate regions are detected using the colour features of the pixels in the detection step. Following this, the cascaded feedforward neural networks with random weights (FNNRW) classifiers are used for shape and content recognition [36]. The experimental results indicate that the average running time of the whole system is less than 10 ms (which would imply the real-time basis of the algorithm), with high accuracy. Our results later corroborate with this. The work performed by Prakash et al. [36] also presents a technique wherein the accuracy of the TSR algorithm may be as high as 99% by using optimum deep learning methodologies. In our case, we shall use the off-the-shelf software base as worked upon by Fan et al. [35] to present the concept. The algorithm is able to classify detected road signs; as a result, it is possible to know whether the ego-vehicle is near any construction zones, pedestrian crossing zones, school zones, traffic signals, or junctions. We shall use this data to our advantage. Several examples of the emergency signs are shown in Figure 6.

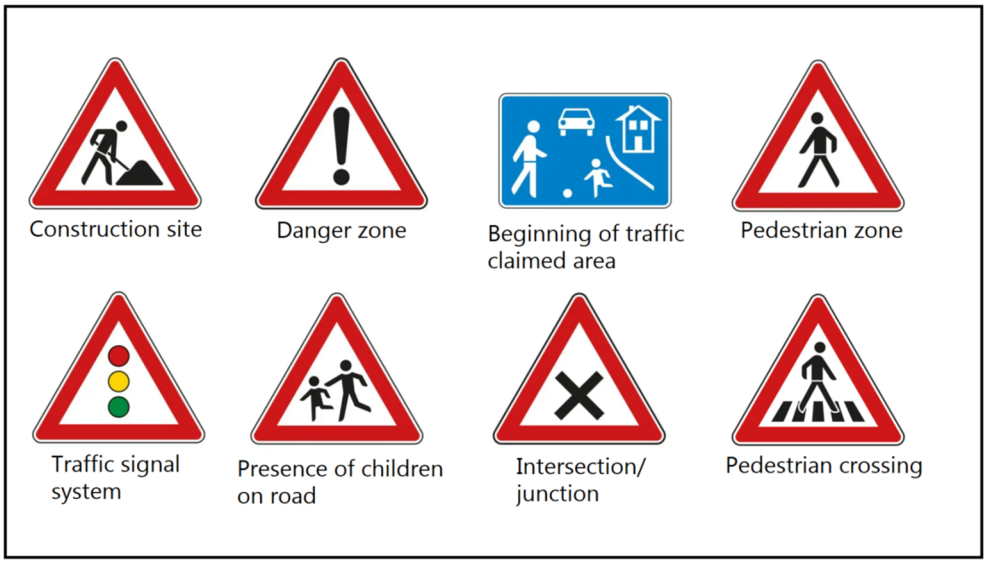


**Figure 6.** Examples of emergency traffic signs.

The TSR algorithm may be divided into two main parts—Detection and Classification. The functionality of this algorithm is shown in Figure 7.

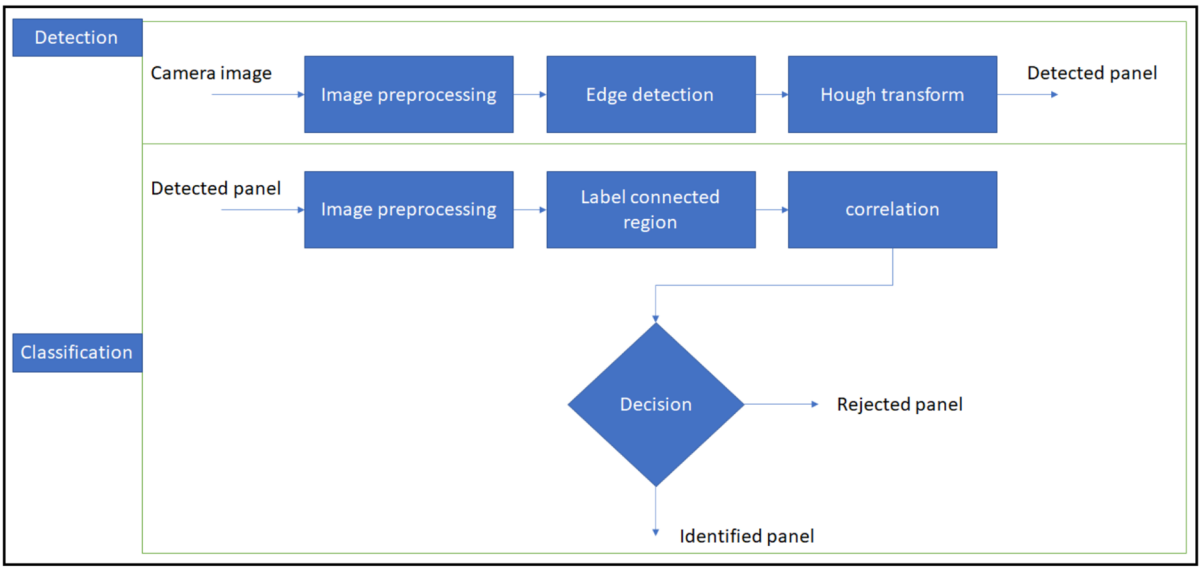


**Figure 7.** TSR algorithm [27].

### *2.5. Real-Time Traffic Density Estimation*

From a traffic engineering perspective, density is an important macroscopic parameter of urban and highway traffic flow. Traffic density is defined as the number of vehicles per unit length of roadway segment [37]. It may be calculated in two steps: vehicle recognition and counting. Considerable efforts have been made to detect traffic density surrounding the ego-vehicle, based on the available sensor data. Most traffic management systems are based on a point detector such as the loop detector; therefore, it is still difficult to accurately estimate the traffic density based on a roadway section [37,38]. However, in recent years, with the increasing number of vehicles equipped with several sensor systems—such as cameras and RaDARs—this is changing to include new traffic environments that may easily collect traffic data by using various vehicle sensors. From this perspective, the use of sensor information offers new opportunities for traffic density estimation [39,40].

Based on the work of Lee et al. [39], an estimation model was constructed based on the data available from vehicle sensors like cameras and RaDAR. The first step here was to develop a software module that could generate the error of distance measurement analogous to real road driving. In the second step, this study conducted traffic simulation experiments by combining the true distance value measured in the simulation, with the error value calculated by the error-generating module of the first step. Finally, the evaluation procedure was completed by comparing the estimation results of traffic density to the true density value. The functionality of this algorithm may be seen in Figure 8.

With the development of various technologies for traffic condition monitoring, a number of studies have been conducted to estimate traffic density using advanced technologies. It is proven that the use of on-vehicle devices of spacing measurement are highly reliable for estimating traffic density [39,40]. Toward this effort, a method based on the vehicle-to-vehicle distance, measured by probe vehicles equipped with ADAS, was developed by Erdem et al. [40]. The key characteristic of this method was to estimate flow, density, and speed from the ADAS vehicle sensors, and verify them with the probe vehicle data without any assumptions regarding the traffic flow characteristics. In our research, however, we shall be referring to the work performed by Lee et al. [39], and we will use the Real-time Traffic Density Estimation algorithm developed by the authors to understand whether the ego-vehicle is operating in a sparse or dense traffic environment.

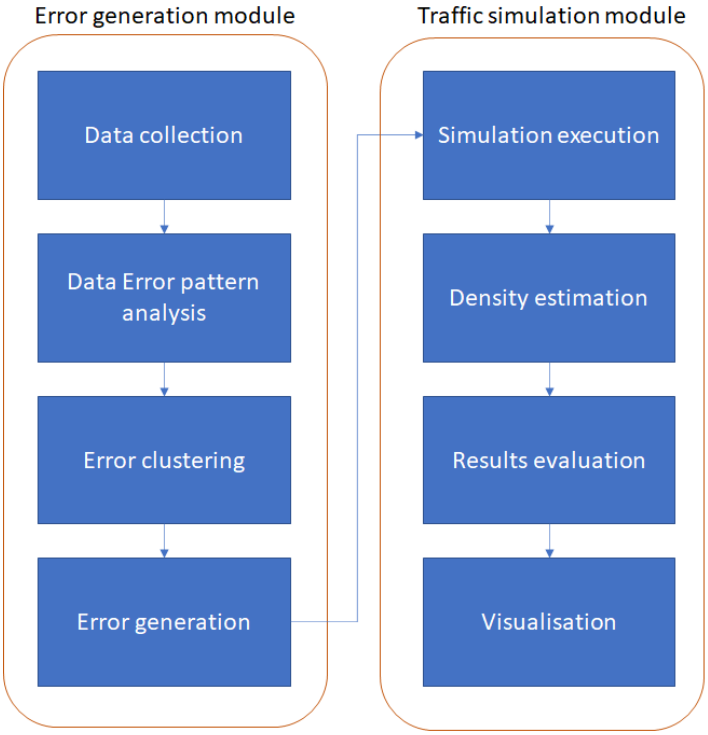


**Figure 8.** Traffic density estimation algorithm [40].

## 3. Proposed Solution of Sensor Fusion with Tracker Switching

In our experiments, we analysed multiple sets of videos from the KITTI dataset in highly dense and sparse traffic conditions. Dense traffic videos are those videos taken in fully urban environments, typically at junctions, traffic jams, and traffic signals. These videos have multiple scenarios consisting of traffic signals, pedestrians, cyclists, and numerous vehicles in the near-field of the ego-vehicle. Sparse traffic scenarios consist of highway and urban driving conditions, without any traffic jams and with a moderate presence of surrounding road objects. The accuracy of the fusion architecture with fixed tracker logic—first with the EKF and then with the UKF—is given in Table 1. The speed of execution for the two architectures may be seen in Table 2.

**Table 1.** Accuracy of different fusion architecture with fixed tracker.

| Video Scenario | Extended Kalman Filter OR (%) | Unscented Kalman Filter OR (%) |
|---|---|---|
| 100% Sparse traffic | 66.8 | 57.12 |
| 100% Dense traffic | 53.06 | 66.32 |

**Table 2.** Speed of different fusion architecture with fixed tracker.

| Video Scenario | Extended Kalman Filter fps | Unscented Kalman Filter fps |
|---|---|---|
| 100% Sparse traffic | 36.9 | 28.28 |
| 100% Dense traffic | 22.5 | 20.13 |

All experiments are conducted on an intel i5-based Windows system with NVIDIA GTX GeForce 1650 graphics card. If the hardware is changed or upgraded, the execution time for the algorithms is also expected to change; however, the accuracy ratings are likely to remain similar.

### *3.1. Fusion Architecture with Single Fixed Tracker*

Figure 9 shows the experimentation strategy that we have adapted to analyse the efficacy of the real-time tracker switching algorithm as compared to the single tracker fusion architecture.

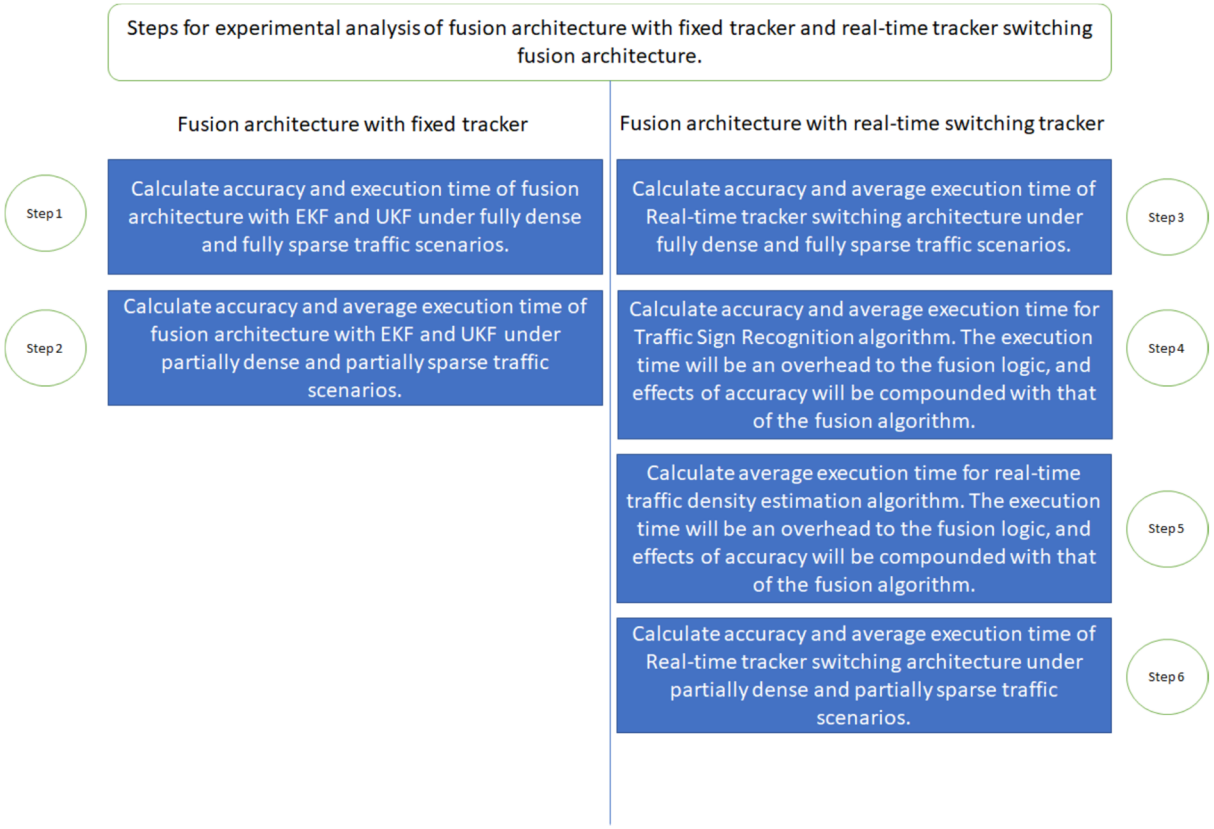


**Figure 9.** Steps followed for critical analysis of the fusion architecture with tracker switching.

To commence, we calculate the accuracy and execution time of the fusion architecture with the fixed EKF and UKF tracker under fully dense and fully sparse traffic scenarios. The videos in this case are chosen from scenarios in traffic jams at junctions (thereby leading to 100% dense traffic scenario) and on highways with a smaller number of road occupants surrounding the ego-vehicle (thereby leading to 100% sparse traffic scenario). This step shall give us a good picture of the peak performance of the fusion architectures equipped with either the UKF or the EKF.

All accuracy values in this case are derived by comparing the output of our algorithm with the ground truth data provided in the KITTI dataset. Equation (1), as stated in Section 2.3, is used to calculate the OR value. The ground truth data for detections may be found at: http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark (as accessed on 7 September 2022). The execution time is calculated in terms of frame rate (frames per second, or fps). The higher the frame rate, the quicker the execution time, and vice-versa. Ideally, high OR and fps values are expected for a high-performance real-time algorithm.

It may be seen that the execution speed of the EKF is the best of the two in all scenarios, whereas the accuracy of the UKF is the best in dense traffic scenarios. In sparse traffic scenarios, however, the EKF provides the most reliable readings. Thus, while the EKF is more effective in sparse traffic conditions, and demonstrates the highest accuracy and speed of execution, in dense traffic scenarios, the UKF provides the highest accuracy with a slight drop in execution speed.

Following this, we assess the performance of the fusion architecture with a fixed tracker under varying traffic densities. The videos chosen for this are drawn from the

KITTI dataset, where the traffic situation gradually changes from highly sparse to dense, thereby exposing the fusion architecture to both highly dense and sparse traffic scenarios. The results for running these videos with the fixed tracker architecture only are shown in Table 3.

**Table 3.** Accuracy and execution time of fixed tracker fusion architecture.

| Videos with Varying Scenarios | EKF | | UKF | |
|---|---|---|---|---|
| | OR (%) | fps | OR (%) | fps |
| Video 1–100% sparse | 66.8 | 36.9 | 57.12 | 28.28 |
| Video 2–80% sparse/20% dense | 63.5 | 35.35 | 61.22 | 27.22 |
| Video 3–60% sparse/40% dense | 61.22 | 31.03 | 61.96 | 26.36 |
| Video 4–40% sparse/60% dense | 60.74 | 27.4 | 63.36 | 25.3 |
| Video 5–20% sparse/80% dense | 55.03 | 23.03 | 65.88 | 22.38 |
| Video 6–100% dense | 53.06 | 22.5 | 66.32 | 20.13 |
| **Average** | **58.71** | **27.862** | **63.748** | **24.278** |
| **Variance** | **22.22835** | | **9.584589** | |

In Table 3, the average accuracy, fps, and variance in accuracy are important and noteworthy parameters. It may be seen that as the videos move from completely sparse to completely dense scenarios, the performance of the UKF increases and that of the EKF deteriorates correspondingly. Any ADAS algorithm, however, is expected to perform consistently [2]. While the accuracy and execution speed are important parameters, the extent to which an algorithm diverges from its mean accuracy (which is variance) is also an important parameter, as it describes the consistency of the fusion architecture [41].

Considering the average accuracy, variance, and fps as reference, we now introduce the Traffic Sign Recognition and the Real-time Traffic Density Estimation algorithm before integrating them with our fusion architecture.

### *3.2. Traffic Sign Recognition*

Different countries use different sets of traffic signs. Accordingly, it is necessary to train the algorithm on appropriate regional signs. For our work, we deploy the traffic sign detection algorithm trained on Japanese road signs. We use the algorithm created by Wahyono et al. [42] and integrate that code base in our software architecture. It must be noted that road signs differ by geography, and every country/region requires traffic signs specific to the area. In our work, while we have chosen the Japanese road signs for training purposes, the basic symbols (road crossing, zebra crossing, traffic signal ahead, etc.) remain consistent globally. However, for a more geography-specific application, the TSR algorithm must be trained on a country-specific dataset.

In their work, Wahyono et al. [42] present a survey of various Traffic Sign Recognition models. Typically, the localisation is conducted with the colour information of the traffic sign using colour-based thresholding. Since the RGB colour space is very fragile regarding changes in lighting, the Hue Saturation Value (HSV) colour space is used instead. The HSV space models human vision better than the RGB and allows some variation in the lighting. However, using a fixed threshold is not reliable due to its dependency on lighting conditions and camera overexposure [43]. Therefore, in this work, the Maximally Stable Extremal Region (MSER) [44] method is employed for localising the candidate region. The MSER method localises the stable region in which the pixel values in the region possess low variance distribution.

Figures 10 and 11 show the output of the TSR algorithm implemented on the KITTI dataset videos. The detected traffic signs are displayed on the top by the algorithm and the

same may be read by our architecture to understand the road conditions surrounding the ego-vehicle.

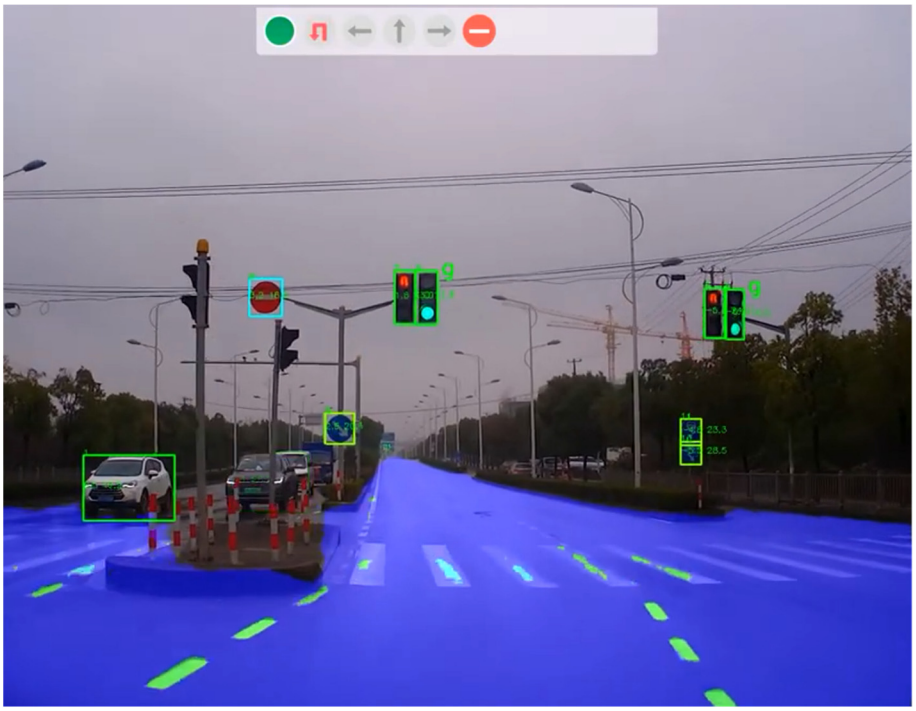

**Figure 10.** TSR algorithm running on KITTI dataset—example 1.

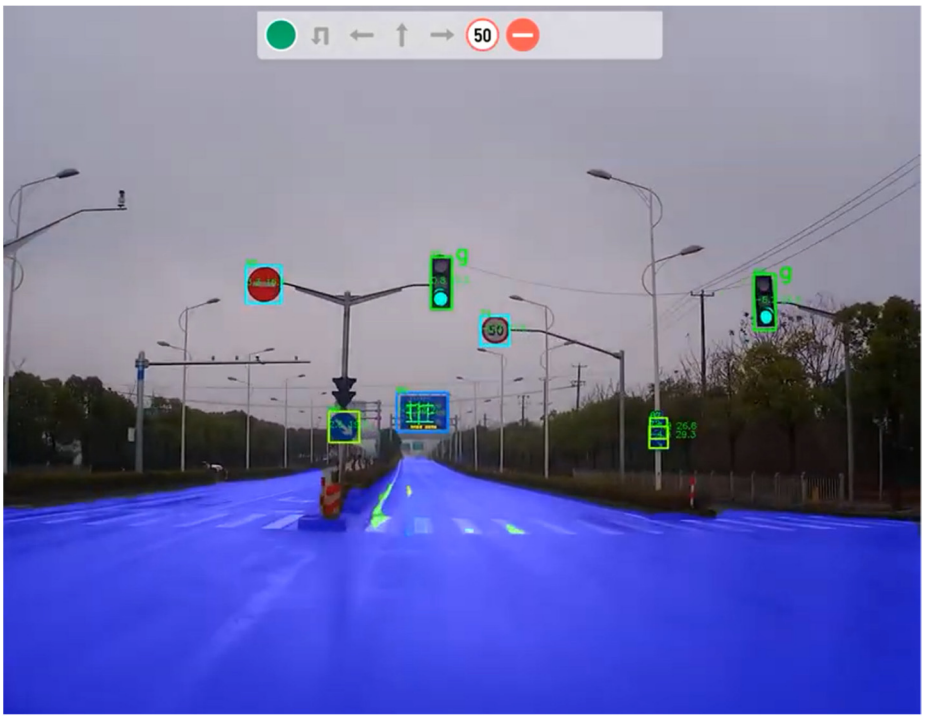


**Figure 11.** TSR algorithm running on KITTI dataset—example 2.

It may be seen that alongside the speed limits and emergency traffic signs stated in Figure 2, the state of the traffic junction signals is also detected. In our work, we shall not distinguish between the state of the traffic signal. If any emergency sign or traffic signal is detected on the projected road of the ego-vehicle, the scenario shall be declared an emergency scenario.

Abiding by the framed experimentation process, Table 4 shows the performance parameters of the Traffic Sign Detection algorithm. The accuracy here is calculated in terms of mAP (Mean Average Precision). By calculating the true positives, false positives, and false negatives values, the mAP values are calculated. Ideally, a higher mAP suggests a high-performance algorithm with the least deviation in output as compared to the ground truth data. Table 4 summarises the performance of the TSR algorithm.

**Table 4.** Performance of the TSR algorithm over various KITTI dataset videos.

| Video Scenarios | mAP (in %) | Time of Execution (in ms) |
|---|---|---|
| Video 1 | 94.7887 | 5.65 |
| Video 2 | 95.0902 | 5.98 |
| Video 3 | 93.381 | 5.39 |
| Video 4 | 93.7782 | 4.819 |
| **Average** | **94.2595** | **5.459** |

In this case, it must be noted that the time of execution for the TSR algorithm will be added as an overhead to the fusion logic. As a result, a lesser time of execution is preferred alongside a high mAP score.

### *3.3. Real-Time Traffic Estimation*

For real-time traffic density estimation, we use the work of Lee et al. [39]. In their work, the authors explored the possibility of using vehicle sensors to estimate traffic density; they also analysed the reliability change of the density estimation according to the change in sensing range of the ADAS and radar systems. Compared to the existing research [35], their study dealt with sensor-based measurement errors and reflected these errors in the traffic density estimation process. The performance evaluation of the experiment results was based on Mean Absolute Percentage Error (MAPE) statistics for the various test cases. Through the experiment simulation results and their analysis, this study led to the conclusion that using a front RaDAR and camera sensor to estimate traffic density surrounding the ego-vehicle was more effective than using only the camera sensor, as RaDAR gives access to the relative speed and the distance of target objects as well.

The inter-vehicle distance in this case is defined as the distance from the front bumper of the ego-vehicle to the rear bumper of the target vehicle. Therefore, the distance headway is calculated by adding the inter-vehicle distance to the length of the sensor-equipped vehicle. Although it is possible to measure the distance to the target vehicle in the surrounding lane, this study assumes that it is possible to measure only the vehicle in front of the same lane, considering the accuracy of the distance measurement. The RaDAR-based distance measurement is stable compared to the image-based measurement from a traffic density perspective. Considering these points, this study uses RaDAR-based distance data as a reference value in order to verify the reliability of ADAS camera-based distance data, and this study defines the difference between the RaDAR-based distance and camera-based distances as an error of image-based distance. Based on this definition, this study analysed how the error of image-based distance changes as the measurement distance gets longer [45]. It is particularly important to accurately reflect fluctuations in distance measurement error into simulation because it may affect the reliability of the density estimation in simulation environments where there are various distances between vehicles depending on traffic conditions. Although there are various traffic density estimation methods, it is effective to use the relationship between the traffic density and the distance headway.

It must be noted that this algorithm requires the fusion of the camera and RaDAR sensor data. Accordingly, the object list generated by the camera and RaDAR sensors is fed to this software block. At the output, depending on whether the density of traffic is high or low (by setting a threshold value), we may conclude whether the ego-vehicle is currently

in dense traffic or sparse traffic. This is printed on the output images on the top left corner as seen in the Figures below. Figures 12–15 show the output of the RTDE algorithm.


**Figure 12.** RTDE example output 1—SPARSE scenario.

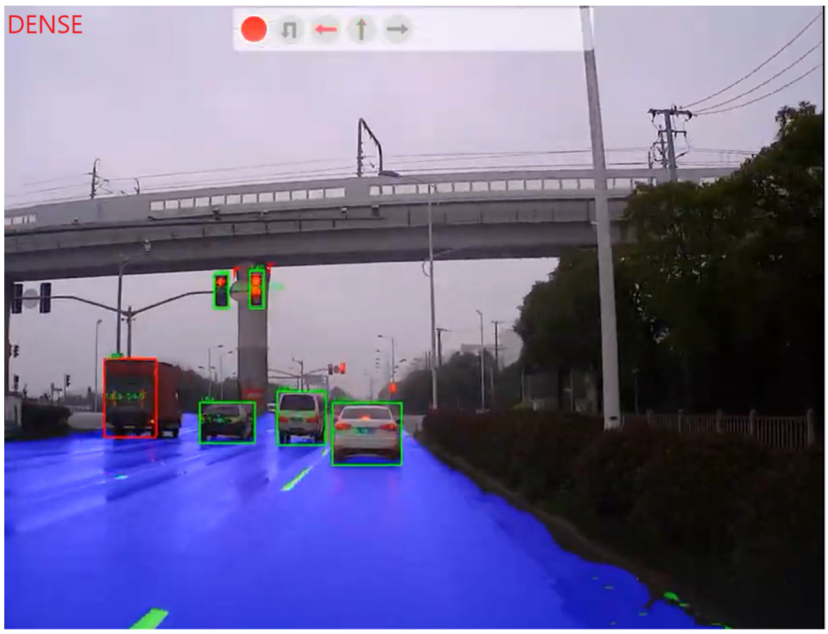


**Figure 13.** RTDE example output 2—DENSE scenario.

To demonstrate the working efficacy of this algorithm, we reconstruct virtualized traffic on MATLAB on a segment of simulated highway. This simulation will help us to know the accuracy and execution speed parameters of the RTDE. An example of the scenario recreated in MATLAB may be seen in Figure 16.

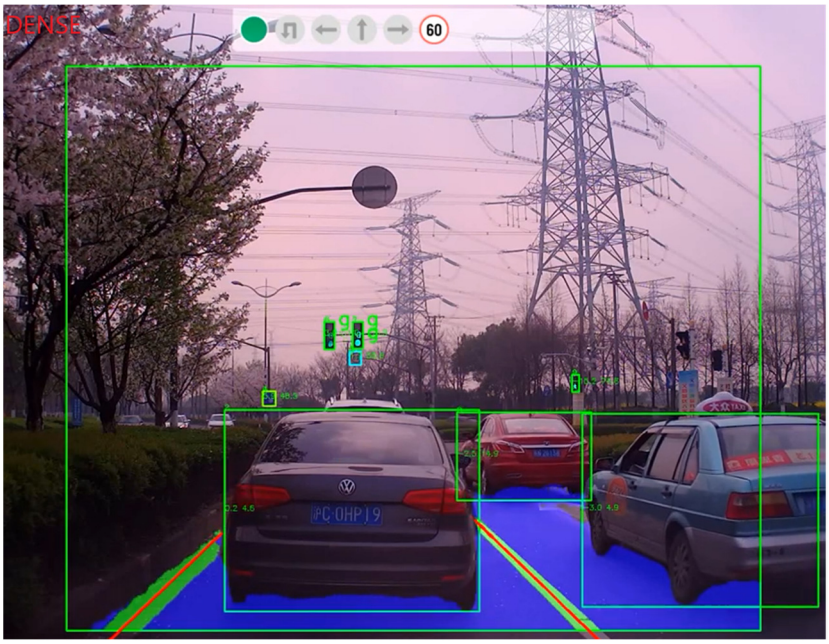


**Figure 14.** RTDE example output 3—DENSE scenario.

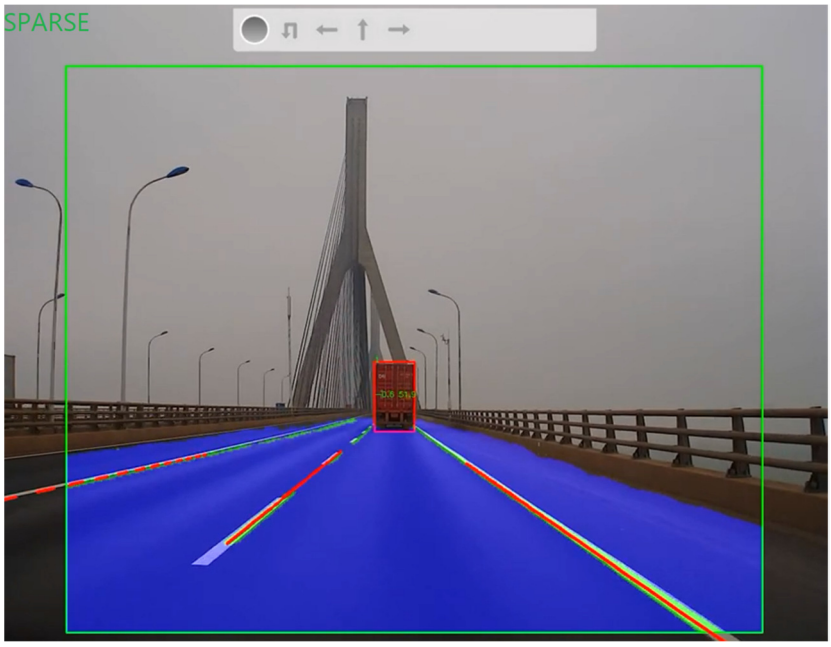


**Figure 15.** RTDE example output 4—SPARSE scenario.

The sensor data used for these experiments derives from the Next Generation Simulation (NGSIM) program. This sensor data value is loaded in the scenario by using the MATLAB Scenario Generator Tool. The roadway on which the data is measured is a stretch of highway with moderate curvature and four lanes. Once the traffic density is estimated, the value is compared with the ground truth values. In this algorithm, there is still a small error which arises when the objects are not always correctly detected by the fusion architecture (when there is a false positive or negative). As a result, there is a small error associated with the true traffic density predicted by the algorithm. However, in our current work, since the error is negligible, we may consider that we always receive true traffic density status.

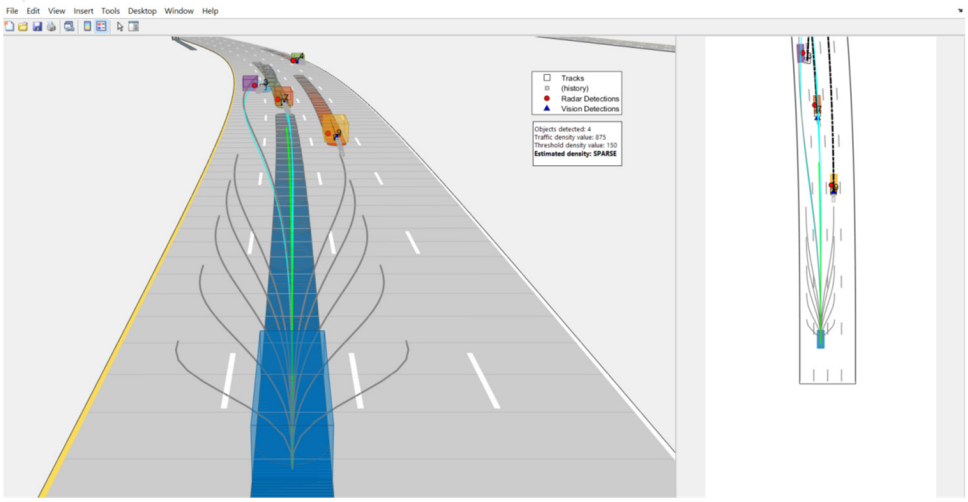


**Figure 16.** Simulation of the highway scenario in MATLAB to calculate the efficacy of the RTDE.

From Tables 4 and 5, the time overhead added by Traffic Sign Recognition and traffic density estimation algorithms is 5.459 ms (for TSR) and 5.221 ms (for RTDE). Thus, a total of 10.68 ms overhead is added to the fusion architecture for every frame. It must be noted that this time overhead may be reduced by optimising the software and using a powerful, hardware-supporting parallel computer. The important factor here is to observe that the accuracy ratings for both algorithms is high (>90%). This ensures that no error is added by these algorithms to the sensor fusion architecture.

**Table 5.** Performance of the RTDE algorithm.

| Video Scenarios | Accuracy (in %) | Time of Execution (in ms) |
|---|---|---|
| Video 1 | 98.9097 | 5.23 |
| Video 2 | 99.461 | 5.112 |
| Video 3 | 99.5568 | 5.365 |
| Video 4 | 99.002 | 5.178 |
| **Average** | **99.23** | **5.221** |

### *3.4. Fusion with Tracker Switching*

As we have seen before, both the UKF and the EKF have their own advantages and disadvantages. In this architecture, we leverage the pros of both trackers by adding a small overhead of the TSR and the RTDE, as mentioned in Sections 2.2 and 2.3. The functionality of our architecture may be understood from the information in Figure 17.

In this architecture, it is important to note that the TSR and the RTDE are initiated simultaneously alongside the sensor object detection algorithms. This is also important as the camera and RaDAR algorithms are required to drive the TSR and the RTDE.

If the TSR algorithm detects an emergency traffic sign, it shall provide a TRUE output. Similarly, if the RTDE algorithm detects a highly dense traffic, the TRUE output is seen. If either of the TSR or RTDE provide a TRUE output, the UKF tracker is selected to operate on the fusion architecture. If both algorithms provide a FALSE output, the EKF tracker is selected. This functionality is shown in Table 6.

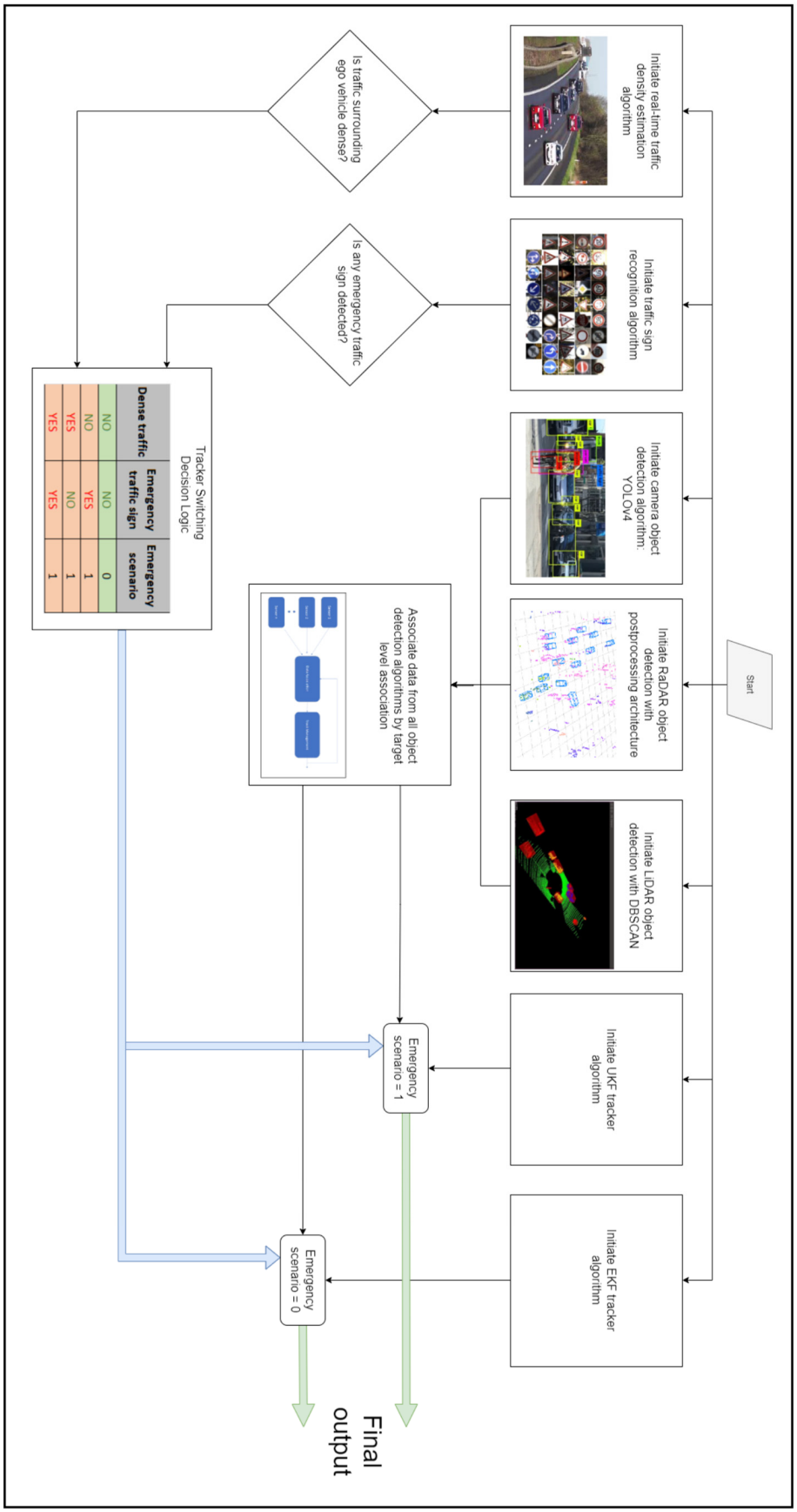


**Figure 17.** Architecture of proposed fusion logic with tracker switching.

**Table 6.** Tracker selection based on the output of TSR and RTDE.

| Dense Traffic | Emergency Traffic Sign | Emergency Scenario | Tracker Selected |
|---|---|---|---|
| FALSE | FALSE | NO | EKF |
| TRUE | TRUE | YES | UKF |
| TRUE | FALSE | YES | UKF |
| TRUE | TRUE | YES | UKF |

To analyse this architecture, we run the same videos used in Section 2. The OR and fps values are calculated using the same principles as stated in Section 2. The efficacy parameters are as shown in Table 7.

**Table 7.** Performance of fusion logic with switched trackers.

| Videos with Varying Scenarios | Switched Tracker Architecture | |
|---|---|---|
| | OR (%) | fps |
| Video 1–100% sparse | 66.8 | 29.21 |
| Video 2–80% sparse/20% dense | 63.41 | 27.03 |
| Video 3–60% sparse/40% dense | 62.02 | 25.16 |
| Video 4–40% sparse/60% dense | 62.94 | 23.36 |
| Video 5–20% sparse/80% dense | 64.25 | 20.55 |
| Video 6–100% dense | 65.23 | 19.28 |
| **Average** | **63.57** | **23.076** |
| **Variance** | **2.46** | |

Figures 18–20 show several outputs of the same video captured at an instance where the tracker switches from the UKF to the EKF, as the scenario gradually changes from dense traffic to sparse. The video is obtained as the ego-vehicle begins from a junction and all the surrounding vehicles gradually space out. As a result, for the first few seconds, we see that the ego-vehicle is in densely congested traffic, and gradually enters a sparse traffic environment.

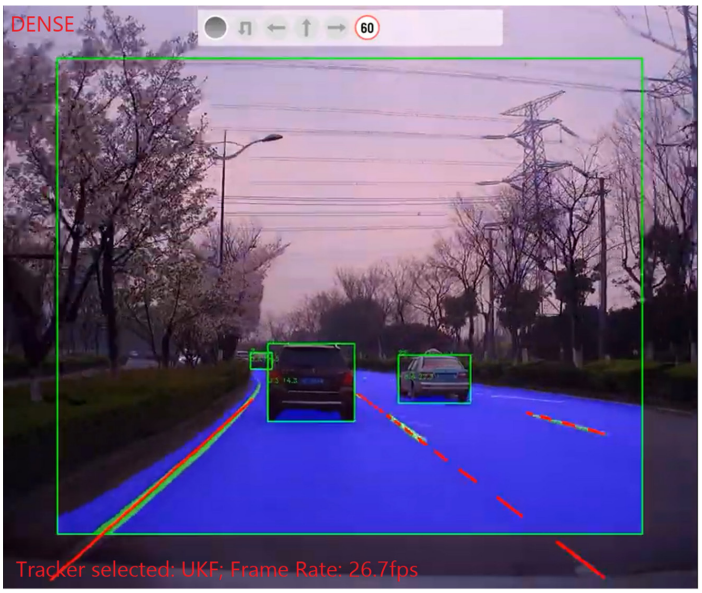


**Figure 18.** Image taken at t = 12.33 s. Traffic state: Dense. Tracker—UKF.

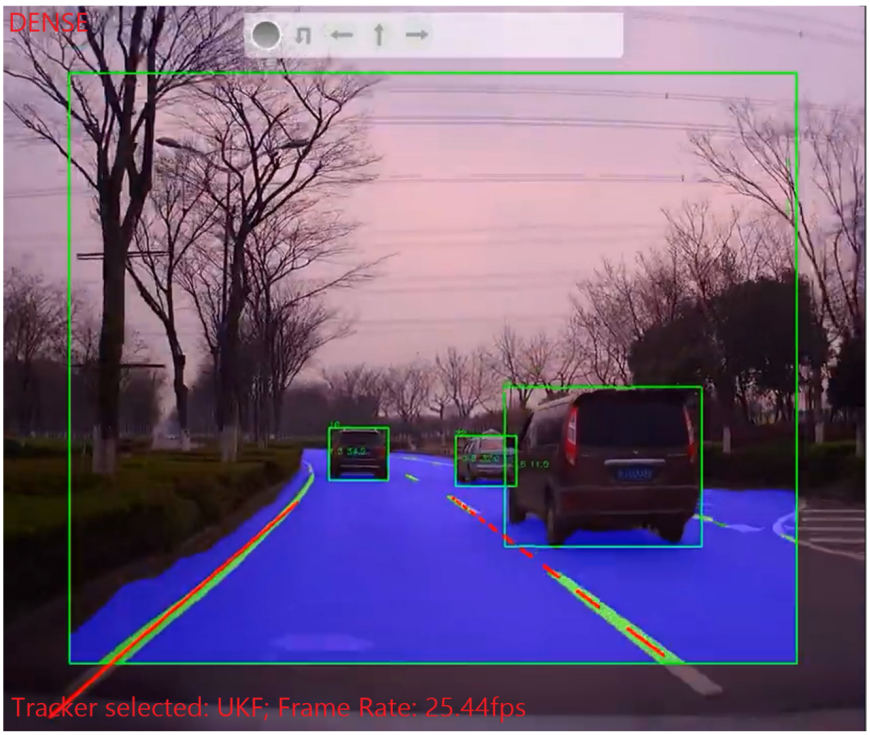


**Figure 19.** Image taken at t = 13.03 s. Traffic state: Dense. Tracker—UKF.


**Figure 20.** Image taken at t = 13.89 s. Traffic state: Sparse. Tracker: EKF.

It may be seen in Figures 18–20 that the fusion architecture switches the tracker algorithm in real time. As a result, optimum performance is observed as the best possible tracker is selected, due to the switching logic. Table 8 shows the comparison of the OR and fps values of the switching tracker architecture against the fusion architecture with the fixed EKF and the UKF.

**Table 8.** Comparison of all 3 architectures—switched tracker-based fusion, EKF- and UKF-based fusion.

| Videos with Varying Scenarios | EKF | | UKF | | Switching Tracker Architecture | |
|---|---|---|---|---|---|---|
| | OR (%) | fps | OR (%) | fps | OR (%) | fps |
| Video 1–100% sparse | 66.8 | 36.9 | 57.12 | 28.28 | 66.8 | 29.21 |
| Video 2–80% sparse/20% dense | 63.5 | 35.35 | 61.22 | 27.22 | 63.41 | 27.03 |
| Video 3–60% sparse/40% dense | 61.22 | 31.03 | 61.96 | 26.36 | 62.02 | 25.16 |
| Video 4–40% sparse/60% dense | 60.74 | 27.4 | 63.36 | 25.3 | 62.94 | 23.36 |
| Video 5–20% sparse/80% dense | 55.03 | 23.03 | 65.88 | 22.38 | 64.25 | 20.55 |
| Video 6–100% dense | 53.06 | 22.5 | 66.32 | 20.13 | 65.23 | 19.28 |
| **Average** | **58.71** | **27.862** | **63.748** | **24.278** | **63.57** | **23.076** |
| **Variance** | **22.22** | | **9.58** | | **2.46** | |

We may draw three important conclusions from Table 8, as given below:

Firstly, the average accuracy obtained by using the switching tracker fusion architecture is comparable to that of the UKF-based fusion. This value is considerably higher than the fusion architecture that is based on the EKF.

Secondly, the execution speed of the switching tracker fusion architecture is marginally less than that of the UKF-based architecture. Both these values are considerably less than that of the EKF-based architecture.

Thirdly, and most importantly, the variance of accuracy is the lowest in the switching tracker-based fusion architecture, as compared to both the EKF and UKF-based fixed tracker fusion architectures.

Understandably, the switching tracker fusion architecture maintains high accuracy as it chooses the tracker which provides the highest accuracy. Figure 21 shows the change in accuracy for all three architectures as the scenario gradually changes from highly dense to highly sparse.

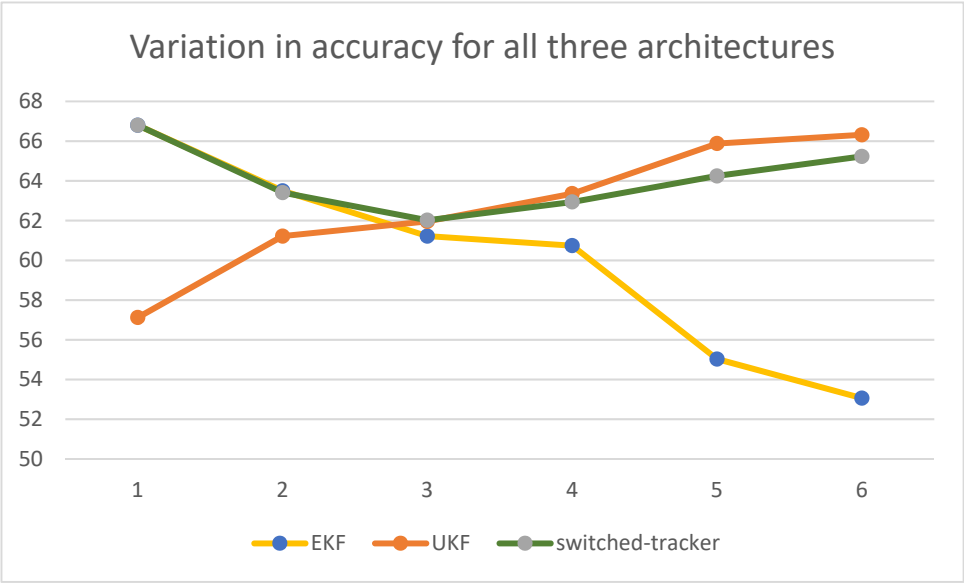


**Figure 21.** Variation in accuracy of the three architectures. The video number is represented on *X*-axis (traffic density increases as video index number increases) and average OR value is represented on the *y*-axis.

Figure 21 and Table 8 show that the switched tracker architecture provides the most consistently accurate output whilst still maintaining a ~23 fps average frame rate. The high accuracy is achieved because it combines the benefits of both tracking algorithms. The speed of execution is reduced as the architecture also uses both the EKF and the UKF as and when necessary, which adds to the computing complexity, thereby leading to a lesser fps temporarily. Moreover, the TSR and RTDE algorithms also add an overhead time of ~10 ms per frame, which leads to the decrease of approximately 5 fps. Figure 22 shows the performance comparison of the three fusion architectures based on the three important parameters—variance, execution speed, and accuracy.

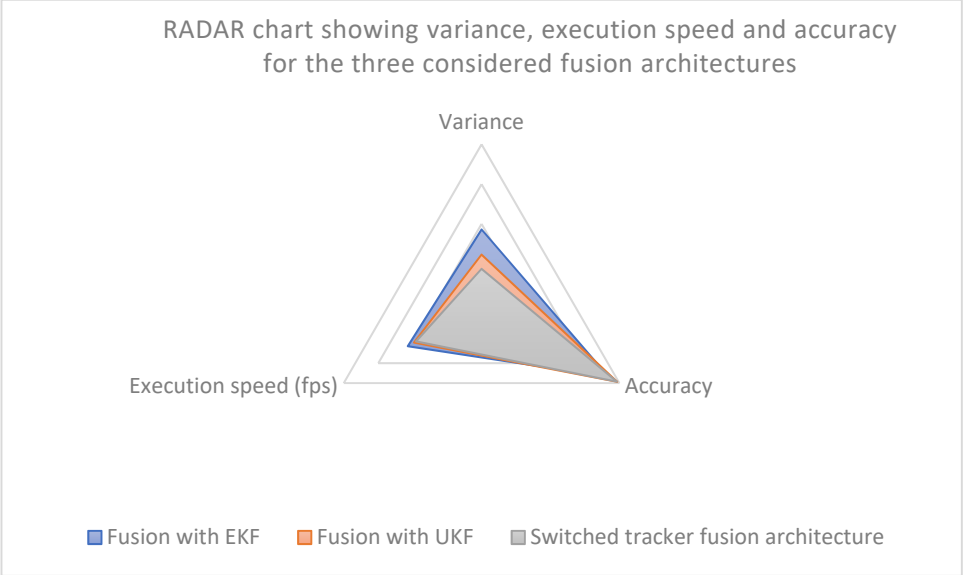


**Figure 22.** RADAR chart of the performance of the three considered fusion architectures based on variance, execution speed, and accuracy.

However, the speed of execution is still respectable considering that it leads to an 88.94% increase in consistency compared to the fusion architecture with the EKF, and a 74.32% increase compared to the UKF-based fusion architecture. These values are calculated based on the percentage change in the variance of accuracy that is achieved by running these three architectures across the same set of videos. The accuracy of switched fusion architecture will always be capped by the highest accuracy achievable by its constituent tracking algorithms. However, using such a technique leads to an output which is the most accurate, considering that the logic selects the tracker best suited for specific environments. The modified architecture leads to a drop in the average execution speed as compared to the fusion with the EKF and the UKF; however, as discussed previously, this may be reduced when a higher-end computing device is used to optimise the code further.

## 4. Conclusions

It is understood that all ADAS algorithms possess unique advantages and disadvantages, and the best choice of algorithms is defined by the requirement of the feature [41,46]. For most ADAS algorithms, maintaining consistency across all external scenarios is critical [47]. Following the critical analysis of the fusion architecture based on the switching tracker mechanism, it must be noted that by using algorithms like the TSR and the RTDE, we include certain shortcomings. The added expanse of code affects both the space and time complexity of the overall logic. Therefore, a question arises regarding whether the gains are worth the sacrifice of simplicity. In this case, even though the execution speed drops by a maximum of approximately 18%, it still remains above 20 fps in most cases.

The gains, on the other hand, are significant. Not only do we see an optimally performing architecture, but the consistency of output increases (that is, the variance

in accuracy decreases) by 74.32% and 88.94%, respectively, when compared with fusion architectures which employ the UKF and the EKF alone. After using the switching tracker fusion logic, the performance is stable across a variety of videos, without any dependency on the type of scenario being fed to the fusion algorithm. The variance in accuracy decreases substantially. As seen in Figures 18–20, as the algorithm switches in real time between the UKF to the EKF, a smooth consistency in performance is achieved whilst maintaining a moderate execution speed. In real life, driving scenario environments change rapidly from dense traffic to sparse. In such cases, sensor fusion architectures that are equipped with the switched tracker would be able to perform consistently by benefiting from the best from both worlds—high accuracy in dense traffic provided by the UKF, and in sparse traffic by the EKF, respectively.

For future scope, this architecture may be further improved by optimising the TSR and RTDE codes, for the creation of a minimum overhead. Moreover, different trackers (like JPDA—Joint Probabilistic Data Association filter) and a greater variety of such trackers, may be used to allow the system to become more diverse and immune to several environmental factors.

**Author Contributions:** Conceptualization, A.D. and V.P.; Data curation, A.D.; Formal analysis, A.D.; Methodology, A.D. and V.P.; Project administration, V.P.; Resources, A.D. and V.P.; Software, A.D.; Supervision, V.P.; Validation, A.D. and V.P.; Visualization, A.D. and V.P.; Writing—original draft, A.D. and V.P.; Writing—review & editing, A.D. and V.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research has received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Kim, J.; Han, D.S.; Senouci, B. Radar and Vision Sensor Fusion for Object Detection in Autonomous Vehicle Surroundings. In Proceedings of the 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), Prague, Czech Republic, 3–6 July 2018; pp. 76–78. [CrossRef]
2. Kaur, P.; Sobti, R. Sensor Fusion Algorithm for Software Based Advanced Driver-Assistance Intelligent Systems. In Proceedings of the 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India, 15–17 December 2018; pp. 457–460. [CrossRef]
3. Warren, M.E. Automotive LIDAR Technology. In Proceedings of the 2019 Symposium on VLSI Circuits, Kyoto, Japan, 9–14 June 2019; p. 254. [CrossRef]
4. Wang, Y.; Liu, D.; Matson, E. Accurate Perception for Autonomous Driving: Application of Kalman Filter for Sensor Fusion. In Proceedings of the 2020 IEEE Sensors Applications Symposium (SAS), Kuala Lumpur, Malaysia, 9–11 March 2020; pp. 1–6. [CrossRef]
5. Deo, A.; Palade, V.; Huda, M.N. Centralised and Decentralised Sensor Fusion-Based Emergency Brake Assist. *Sensors* **2021**, *21*, 5422. [CrossRef] [PubMed]
6. Chen, X.; Wang, X.; Xuan, J. Tracking Multiple Moving Objects Using Unscented Kalman Filtering Techniques. *arXiv* **2018**, arXiv:1802.01235.
7. Tibba, G.; Malz, C.; Stoermer, C.; Nagarajan, N.; Zhang, L.; Chakraborty, S. Testing automotive embedded systems under X-in-the-loop setups. In Proceedings of the 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), New York, NY, USA, 7–10 November 2016; pp. 1–8. [CrossRef]
8. Schlegl, T.; Bretterklieber, T.; Neumayer, M.; Zangl, H. A novel sensor fusion concept for distance measurement in automotive applications. In Proceedings of the IEEE SENSORS, Waikoloa, HI, USA, 1–4 November 2010; pp. 775–778.
9. Kumar, C.; Punitha, R. YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications. In Proceedings of the 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 20–22 August 2020; pp. 1316–1321.
10. Manjunath, A.; Liu, Y.; Henriques, B.; Engstle, A. Radar Based Object Detection and Tracking for Autonomous Driving. In Proceedings of the 2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM), Munich, Germany, 15–17 April 2018; pp. 1–4. [CrossRef]

11. Deng, D. DBSCAN Clustering Algorithm Based on Density. In Proceedings of the 2020 7th International Forum on Electrical Engineering and Automation, Hefei, China, 25–27 September 2020; pp. 949–953.
12. Tan, S.; Lu, G.; Jiang, Z.; Huang, L. Improved YOLOv5 Network Model and Application in Safety Helmet Detection. In Proceedings of the 2021 IEEE International Conference on Intelligence and Safety for Robotics (ISR), Tokoname, Japan, 4–6 March 2021; pp. 330–333. [CrossRef]
13. Zhang, F.; Clarke, D.; Knoll, A. Vehicle detection based on LiDAR and camera fusion. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 8–11 October 2014; pp. 1620–1625. [CrossRef]
14. Thakur, R. Scanning LIDAR in Advanced Driver Assistance Systems and Beyond: Building a road map for next-generation LIDAR technology. *IEEE Consum. Electron. Mag.* **2016**, *5*, 48–54. [CrossRef]
15. Yabroudi, M.E.; Awedat, K.; Chabaan, R.C.; Abudayyeh, O.; Abdel-Qader, I. Adaptive DBSCAN LiDAR Point Cloud Clustering for Autonomous Driving Applications. In Proceedings of the 2022 IEEE International Conference on Electro Information Technology (eIT), Mankato, MN, USA, 19–21 May 2022; pp. 221–224. [CrossRef]
16. Kellner, D.; Barjenbruch, M.; Dietmayer, K.; Klappstein, J.; Dickmann, J. Tracking of Extended Objects with High-Resolution Doppler Radar. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 1341–1353. [CrossRef]
17. Khalil, M.; Eltrass, A.S.; Elzaafarany, O.; Galal, B.; Walid, K.; Tarek, A.; Ahmadien, O. An Improved Approach for Multi-Target Detection and Tracking in Automotive Radar Systems. In Proceedings of the 2016 International Conference on Electromagnetics in Advanced Applications (ICEAA), Cairns, QLD, Australia, 19–23 September 2016; pp. 480–483.
18. Lindenmaier, L.; Tihanyi, V. Comparison of Different Tracking Approaches on Pre-Fused Data for Automotive Perception System. In Proceedings of the 2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo), Szeged, Hungar, 14–16 November 2019; pp. 199–204. [CrossRef]
19. Aeberhard, M.; Rauch, A.; Rabiega, M.; Kaempchen, N.; Bertram, T. Track-to-track fusion with asynchronous sensors and out-of-sequence tracks using information matrix fusion for advanced driver assistance systems. In Proceedings of the 2012 IEEE Intelligent Vehicles Symposium, Madrid, Spain, 3–7 June 2012; pp. 1–6. [CrossRef]
20. Baek, J.W.; Han, B.-G.; Kang, H.; Chung, Y.; Lee, S.-I. Fast and reliable tracking algorithm for on-road vehicle detection systems. In Proceedings of the 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), Vienna, Austria, 5–8 July 2016; pp. 70–72. [CrossRef]
21. Madhukar, P.S.; Prasad, L.B. State Estimation using Extended Kalman Filter and Unscented Kalman Filter. In Proceedings of the 2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3), Lakshmangarh, India, 21–22 February 2020; pp. 1–4. [CrossRef]
22. Mochnac, J.; Marchevsky, S.; Kocan, P. Bayesian filtering techniques: Kalman and extended Kalman filter basics. In Proceedings of the 2009 19th International Conference Radioelektronika, Bratislava, Slovakia, 22–23 April 2009; pp. 119–122. [CrossRef]
23. Bersani, M.; Vignati, M.; Mentasti, S.; Arrigoni, S.; Cheli, F. Vehicle state estimation based on Kalman filters. In Proceedings of the 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), Turin, Italy, 2–4 July 2019; pp. 1–6. [CrossRef]
24. Hu, G.; Gao, B.; Zhong, Y.; Ni, L.; Gu, C. Robust Unscented Kalman Filtering with Measurement Error Detection for Tightly Coupled INS/GNSS Integration in Hypersonic Vehicle Navigation. *IEEE Access* **2019**, *7*, 151409–151421. [CrossRef]
25. Zhao, Y.; Zhang, J.; Hu, G.; Zhong, Y. Set-Membership Based Hybrid Kalman Filter for Nonlinear State Estimation under Systematic Uncertainty. *Sensors* **2020**, *20*, 627. [CrossRef] [PubMed]
26. St-Pierre, M.; Gingras, D. Comparison between the unscented Kalman filter and the extended Kalman filter for the position estimation module of an integrated navigation information system. In Proceedings of the IEEE Intelligent Vehicles Symposium, Parma, Italy, 14–17 June 2004; pp. 831–835. [CrossRef]
27. Wan, E.A.; Van Der Merwe, R. The unscented Kalman filter for nonlinear estimation. In Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373), Lake Louise, AB, Canada, 4 October 2000; pp. 153–158.
28. Lee, D.-J. Nonlinear Estimation and Multiple Sensor Fusion Using Unscented Information Filtering. *IEEE Signal Process. Lett.* **2008**, *15*, 861–864. [CrossRef]
29. Gao, G.; Gao, B.; Gao, S.; Hu, G.; Zhong, Y. A Hypothesis Test-Constrained Robust Kalman Filter for INS/GNSS Integration with Abnormal Measurement. *IEEE Trans. Veh. Technol.* **2022**. [CrossRef]
30. Bhat, S.; Kunthe, S.S.; Kadiwal, V.; Iyer, N.C.; Maralappanavar, S. Kalman Filter Based Motion Estimation for ADAS Applications. In Proceedings of the 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, 12–13 October 2018; pp. 739–743. [CrossRef]
31. Zhang, X.; Fei, X.; Zhu, Y.; Mu, X.; Lv, P.; Liu, H.; He, B.; Yan, T. Novel Improved UKF Algorithm and Its Application in AUV Navigation System. In Proceedings of the 2018 OCEANS—MTS/IEEE Kobe Techno-Oceans (OTO), Kobe, Japan, 28–31 May 2018; pp. 1–4. [CrossRef]
32. Hu, G.; Gao, S.; Zhong, Y.; Gao, B.; Subic, A. Modified strong tracking unscented Kalman filter for nonlinear state estimation with process model uncertainty. *Int. J. Adapt. Control. Signal Process.* **2015**, *29*, 1561–1577. [CrossRef]
33. Gao, B.; Gao, S.; Hu, G.; Zhong, Y.; Gu, C. Maximum likelihood principle and moving horizon estimation based adaptive unscented Kalman filter. *Aerosp. Sci. Technol.* **2018**, *73*, 184–196. [CrossRef]

34. Gao, Z.; Mu, D.; Gao, S.; Zhong, Y.; Gu, C. Adaptive unscented Kalman filter based on maximum posterior and random weighting. *Aerosp. Sci. Technol.* **2017**, *71*, 12–24. [CrossRef]
35. Fan, Y.; Zhang, W. Traffic sign detection and classification for Advanced Driver Assistant Systems. In Proceedings of the 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, China, 15–17 August 2015; pp. 1335–1339. [CrossRef]
36. Prakash, A.S.; Vigneshwaran, D.; Ayyalu, R.S.; Sree, S.J. Traffic Sign Recognition using Deeplearning for Autonomous Driverless Vehicles. In Proceedings of the 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 8–10 April 2021; pp. 1569–1572. [CrossRef]
37. George, R.; Vanajakshi, L.D.; Subramanian, S.C. Area Occupancy-Based Adaptive Density Estimation for Mixed Road Traffic. *IEEE Access* **2020**, *8*, 5502–5514. [CrossRef]
38. Lim, D.; Seo, Y.; Ko, E.; So, J.; Kim, H. S patiotemporal Traffic Density Estimation Based on ADAS Probe Data. *J. Adv. Transp.* **2022**, *2022*, 5929725. [CrossRef]
39. Lee, H.; Lee, J.; Chung, Y. Traffic density estimation using vehicle sensor data. *J. Intell. Transp. Syst.* **2022**, *26*, 1–12. [CrossRef]
40. Erdem, M.; Özdemir, M.K. A simple approach to traffic density estimation by using Kernel Density Estimation. In Proceedings of the 2015 23nd Signal Processing and Communications Applications Conference (SIU), Malatya, Turkey, 16–19 May 2015; pp. 1865–1868. [CrossRef]
41. Cummings, M.L.; Bauchwitz, B. Safety Implications of Variability in Autonomous Driving Assist Alerting. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 12039–12049. [CrossRef]
42. Wahyono; Kurnianggoro, L.; Jo, K.-H. Traffic sign recognition and tracking for a vision-based autonomous vehicle using optimally selected features. In Proceedings of the 2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Hangzhou, China, 28–30 July 2015; pp. 1419–1422. [CrossRef]
43. Wang, Y.; Shi, M.; Wu, T. A Method of Fast and Robust for Traffic Sign Recognition. In Proceedings of the 2009 Fifth International Conference on Image and Graphics, Xi'an, China, 20–23 September 2009; pp. 891–895. [CrossRef]
44. Donoser, M.; Riemenschneider, H.; Bischof, H. Shape Guided Maximally Stable Extremal Region (MSER) Tracking. In Proceedings of the 2010 20th International Conference on Pattern Recognition, Istanbul, Turkey, 23–26 August 2010; pp. 1800–1803. [CrossRef]
45. Ye, Z.Z.; Amirzzodi, J.; Alotaibi, M.; Tshiojwe, I.M.; Al-Harthi, M.; Yaz, E.E. Single-sensor based nonlinear density estimation for traffic networks with multiple routes and sections. In Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228), Orlando, FL, USA, 4–7 December 2001; Volume 5, pp. 4146–4151. [CrossRef]
46. Gao, B.; Gao, S.; Zhong, Y.; Hu, G.; Gu, C. Interacting multiple model estimation-based adaptive robust unscented Kalman filter. *Int. J. Control Autom. Syst.* **2017**, *15*, 2013–2025. [CrossRef]
47. Palade, V.; Deo, A. Artificial Intelligence in Cars: How Close Yet Far Are We from Fully Autonomous Vehicles? *Int. J. Artif. Intell. Tools* **2022**, *31*, 2241005. [CrossRef]