World Electric Vehicle Journal

MDPI

Article

Utilizing Probabilistic Maps and Unscented-Kalman-Filtering-Based Sensor Fusion for Real-Time Monte Carlo Localization

Wael A. Farag * and Julien Moussa H. Barakat

College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait; julien.barakat@aum.edu.kw
* Correspondence: wael.farag@aum.edu.kw

Abstract: An autonomous car must know where it is with high precision in order to maneuver safely and reliably in both urban and highway environments. Thus, in this paper, a reliable and relatively precise position estimation (localization) technique for autonomous vehicles is proposed and implemented. In dealing with the obtained sensory data or given knowledge about the vehicle's surroundings, the proposed method takes a probabilistic approach. In this approach, the involved probability densities are expressed by keeping a collection of samples selected at random from them (Monte Carlo simulation). Consequently, this Monte Carlo sampling allows the resultant position estimates to be represented with any arbitrary distribution, not only a Gaussian one. The selected technique to implement this Monte-Carlo-based localization is Bayesian filtering with particle-based density representations (i.e., particle filters). The employed particle filter receives the surrounding object ranges from a carefully tuned Unscented Kalman Filter (UKF) that is used to fuse radar and lidar sensory readings. The sensory readings are used to detect pole-like static objects in the egocar's surroundings and compare them to the ones that exist in a supplied detailed reference map that contains pole-like landmarks that are produced offline and extracted from a 3D lidar scan. Comprehensive simulation tests were conducted to evaluate the outcome of the proposed technique in both lateral and longitudinal localization. The results show that the proposed technique outperforms the other techniques in terms of smaller lateral and longitudinal mean position errors.

Keywords: UKF; ADAS; autonomous driving; particle filter; Monte Carlo; localization; Kalman Filter; sensor fusion

Citation: Farag, W.A.; Barakat, J.M.H. Utilizing Probabilistic Maps and Unscented-Kalman-Filtering-Based Sensor Fusion for Real-Time Monte Carlo Localization. *World Electr. Veh. J.* **2024**, *15*, 5. https://doi.org/10.3390/wevj15010005

Academic Editors: Liguo Zang and Leilei Zhao

Received: 14 October 2023
Revised: 9 December 2023
Accepted: 12 December 2023
Published: 21 December 2023

1. Introduction

The technological challenges that need to be solved to attain complete autonomy are divided into four areas by designers and researchers in the autonomous driving field: perception, localization, path planning, and controls [1]. Perception is concerned with the task of detecting where objects like cars [2], trucks [3], bikes [4], and pedestrians are [5]; which lane the egocar is driving in [6]; where its boundaries are [7]; and so on. The solutions to these perception subtasks are researched using machine learning techniques that co-ordinate various sensors to accurately detect the surroundings of the egocar [8]. Sonar, radar, lidar, cameras, and other sensors are among them [9]. Further, there is also an additional layer of complexity here: certain objects do not move (static elements), while others do (dynamic elements), and it is critical to distinguish between the two [10].

On the other hand, localization is a vital function for self-driving cars [11], allowing them to locate their position within centimeters on a reference map [12]. This high degree of precision is necessary and allows a self-driving car to comprehend its surroundings and form an understanding of the road itself, road objects, and lane structures [13].

Additionally, the path-planning task is concerned with how an autonomous car drives from one point on the map (the initial position for the trip) to the final goal (the final position for the trip) [14]. It is divided into two subtasks: global path planning, which

is the high-level path on the map for the desired trip, and local path planning, which generates a trajectory profile and a velocity profile for the egocar to maneuver across its environment while avoiding any obstacles, changing lanes, passing other vehicles, etc. The path-planning module receives information from both the perception and the localization modules and sends its generated trajectory to the control module [15].

Moreover, the controls task in autonomous cars is concerned with the automatic application of force on the car actuators to achieve the reference-trajectory tracking goals [16]. In self-driving cars, the controllers' output is exerted on three actuators: steering, throttle, and brake systems [17]. The input information to these controllers takes several forms: the reference trajectory received from the egocar path-planning module, the egocar speed and acceleration signals, and the speed and acceleration of the preceding car [18].

This paper mainly focuses on localization and how to improve its accuracy, as it allows the autonomous car to better comprehend its surroundings and form an understanding of the road and lane structures (e.g., when a lane forks or merges, schedule lane changes and determine lane routes even when markers are obscured) [19].

Sensors play a key role in autonomous vehicle localization, with vehicles being typically equipped with a combination of GPS, IMU, cameras, lidar, radar, and odometry sensors. These sensors provide data about the vehicle's surroundings and its own motion, forming the foundation for accurate localization. Sensor fusion is a crucial technique that integrates data from multiple sensors to enhance accuracy and reliability. By combining information from sources like GPS, IMU, and lidar, a vehicle can compensate for the limitations of individual sensors, improving overall localization performance. Odometry is another important component estimating the vehicle's position based on changes in motion, such as wheel speed and steering angle. While useful, odometry tends to accumulate errors over time, prompting its use in conjunction with other localization methods. Map matching involves comparing sensor data with pre-existing maps of the environment, aiding in the identification of the vehicle's location. This technique is particularly beneficial in environments with recognizable landmarks, contributing to precise localization. Particle filters, a probabilistic method, maintain a set of potential vehicle poses based on sensor measurements and motion models. As the vehicle moves, this set of poses is updated to converge toward the most likely position, enhancing the accuracy of localization.

The 3D pose of an autonomous car inside a high-definition (HD) map, comprising 3D location, 3D orientation, and associated uncertainty, is provided using localization [20]. Unlike the usage of a navigation map with GPS, which only requires a few meters of precision, the localization of a self-driving car requires a far greater level of accuracy relative to the map, generally in the range of centimeters and a few tenths of degree [20].

A landmark-based reference map represents a balanced approach between no-map (SLAM) techniques and detailed high-definition mapping techniques and is thus adopted in this study. Moreover, as a contribution, this work does not depend on a single sensor (lidar), with its high resolution but well-known limitations (e.g., limited reach; severely affected by fog, snow, or dust [21]), but fuses it with radar (e.g., long reach, not affected by fog, but with low resolution [22]) to strengthen both methods and obtain the best out of them. Additionally, by using a carefully tailored UKF as the employed fusion technique, pole-like landmarks can be detected with greater precision and robustness. As an additional contribution, this paper addresses the uncertainties in the detected pole-like landmark co-ordinates. They are represented on the reference map in a probabilistic form. This allows Bayesian inference, applied using a tailored version of a particle filter, to estimate the egocar pose.

2. Literature Review

Satellite-based localization has been available for decades and has undergone various upgrades. Newer systems, such as RTK-GPS [23] or DGPS [24], provide efficient options since they attain centimeter-level precision without the use of any extra techniques. Nonetheless, there are significant uncertainties concerning their consistency. Structures

or other towering road obstacles that obscure the line of sight between the car and the satellites can reduce precision by some meters [25] in metropolitan areas [26]. Moreover, the latency for acquiring the signals usually comes with errors, which is a critical issue that needs to be addressed [27] by filtration, noise cancelation, and compensating for missing readings [28].

As an alternative to the RTK-GPS and according to [15], lidar approaches were shown to be the most promising in terms of performance for the localization of self-driving vehicle applications. Nevertheless, they demand high processing and computational power, in addition to their high cost. Thus, they are rendered to be impracticable in terms of commercialization and cost-affordability. Therefore, greater lidar technology enhancement or alternative methodologies like vision-based localization or ground-penetrating radar localization within lidar maps may open the door for more feasible systems from a commercial point of view. Nevertheless, before these systems can be mass-deployed, further study will be needed to evaluate their robustness, validate their performance across a range of driving circumstances, and refine operation settings.

Accordingly, using reference-dense maps [20] could provide a more reliable localization option [29]. These maps may take several forms like point clouds [30], grid maps [31], or polygon meshes [32]. Nevertheless, map-based systems have the fundamental disadvantage of requiring a substantial amount of memory, which quickly becomes pricey when larger-scale maps are employed [33]. To address this issue, "landmark maps" have sparked significant attention [11]. These maps contain only a very limited number of recognizable and designated features that have been extracted from huge volumes of raw sensory data (collected from cameras, radars, and lidars) and condensed [34], which, in return, reduces the needed memory use by several orders of magnitude.

Subsequently, several efforts have been made in recent research to tackle the challenge of car localization by the employment of lidar point clouds from which pole-like markers are identified. This issue is separated into two sub-problems. The first sub-problem is concerned with the detection of poles and the estimation of their positions, while the second one is concerned with the estimation of the egocar pose based on the position of the detected poles. For example, a pole detector is created by Weng et al. [35] by dividing the region around the egocar and counting the reflected-scan points in each voxel. The detection of poles is accomplished by recognizing the stacks of voxels that are vertically linked and all surpass a predefined threshold. Additionally, the detector employs the RANSAC algorithm [36] to fit all of the points associated with the discovered stacks of voxels to a cylindrical shape. When it comes to the egocar pose estimation, the nearest-neighborhood data association is utilized and paired with a particle filter.

Furthermore, the main emphasis of the Sefati et al. pole-detection approach is to eliminate the ground plane from the point cloud generated by the sensors [37]. A horizontal regular grid is then constructed from the projection of the remaining point-cloud points. The occupancy and height parameters are used to cluster the neighboring cells, and a cylinder is fitted to each of the resulting clusters. Similarly, for the egocar pose estimation, the nearest-neighborhood data association is also employed and paired with a particle filter.

Kummerle et al. improved on the previous work by fitting planes to point-cloud-constructed building facades and also fitting lines to lane markings extracted from stereo camera images [34]. The previous improvements enhance the pole detection, which consequently improves the pose estimation by employing a Monte Carlo method to tackle the data association stage and a nonlinear least-squares optimization technique to refine the finally computed pose.

A more comprehensive work was conducted by Schaefer et al. [38] as they propose three phases to construct a full localization system based on landmark detection. The first phase is the pole extractor, the second one is the mapping, and the third one is the localization. The method used for pole detection considers both the endpoint of the laser beams as well as the available open space between the lidar and those beam endpoints. Therefore, based on a map of only pole landmarks, the presented method

demonstrates precise and consistent car localization for large time scales. Experiments are carried out for 35 hours over the course of 15 months going through different circumstances like construction zones, weather and seasonal variations, different routes, and hordes of moving objects.

Several attempts have been made to remove the dependency on reference maps and only depend on the mounted sensors on the egocar. These endeavors focus on constructing a map and computing localization simultaneously in what is called simultaneous localization and mapping (SLAM) techniques [39]. However, these techniques are computationally much more expensive than the ones that depend on reference maps. Therefore, they usually rely on approximations to speed up processing and create a functional outcome [40]. Consequently, the lack of precision in these techniques hinders their applicability to autonomous driving.

The research gap addressed by this paper lies in the need for a nuanced and versatile mapping technique that strikes a balance between simultaneous localization and mapping (SLAM) methods, which lack mapping information, and high-definition mapping techniques, which can be overly detailed. In contrast to existing approaches, this work adopts a landmark-based reference map, providing a more comprehensive mapping solution.

Furthermore, the existing literature often relies on a single high-resolution sensor, such as lidar, which, despite its advantages, has well-known limitations, including restricted reach and vulnerability to environmental factors like fog, snow, or dust. This paper contributes by adopting a fusion approach, combining lidar with radar. While radar offers a longer reach and is less affected by adverse weather conditions, it has lower resolution. The integration of these complementary sensors is designed to capitalize on their individual strengths and mitigate their weaknesses.

Additionally, the paper introduces a carefully tailored Unscented Kalman Filter (UKF) as the fusion technique, enhancing precision and robustness in detecting pole-like landmarks. This aspect represents a research gap in sensor fusion techniques for landmark-based mapping.

Moreover, the paper addresses a crucial research gap by acknowledging and handling uncertainties in the detected pole-like landmark co-ordinates. Unlike existing methods, this study represents uncertainties in a probabilistic form on the reference map, enabling Bayesian inference. The application of a tailored particle filter allows for the estimation of egocar pose while considering uncertainties, contributing to the overall robustness and reliability of the localization system. This unique approach to handling uncertainties in landmark-based mapping represents a novel contribution to the existing body of research in autonomous vehicle localization.

3. Overview of the Proposed Localization Algorithm

This work proposes a Real-Time Monte Carlo Localization (RTMCL) algorithm for driverless vehicles. The details and the workflow of this algorithm are shown below in Figure 1.

The information required as input by the RTMCL algorithm can be categorized into four groups as follows:

(1) Global position information: this information mainly comes from the GPS. If an IMU unit is also installed in the egocar, then a fusion between the GPS and IMU signals is implemented, resulting in a single output (initial egocar pose) that is used at a later stage by the particle filter. The fusion helps to correct errors that accumulate out of the dead reckoning [41] in periods with missing GPS position deliveries.
(2) Odometry measurements: the algorithm requires the readings of the egocar speed and the steering angle (to calculate the yaw rate). Both will be used to update the particle filter state after filtering from high noise.
(3) Object-detection sensory output: these are mainly lidar, radar, and camera data. Since the camera is not used in this work, only radar and lidar data are collected and fused using the tailored UKF, whose output is clustered using the GB-DBSCAN algorithm

to detect potential pole-like static objects. The velocity measurements (the Doppler signal) received from the radar will help screen out dynamic objects.

(4) Reference landmarks map: this map must be developed in a way to contain pole-like landmarks with full co-ordinates, which will be extracted in an offline process from 3D point-cloud lidar data. The map pole-like landmarks will be aligned with the detected pole-like landmarks (by GB-DBSCAN) through the "data association" process. The Iterative Closest Point (ICP) technique is employed to perform this "data association", which is a very critical step to achieving accurate localization [42].

The above categories of input signals are used by the tailored PF (which will be presented later in more detail [43]) to search for the most accurate egocar pose. The PF is initialized using the output signal of the GPS and IMU fusion and produces a much more precise pose with the help of the identified pole-like landmarks and the reference map.

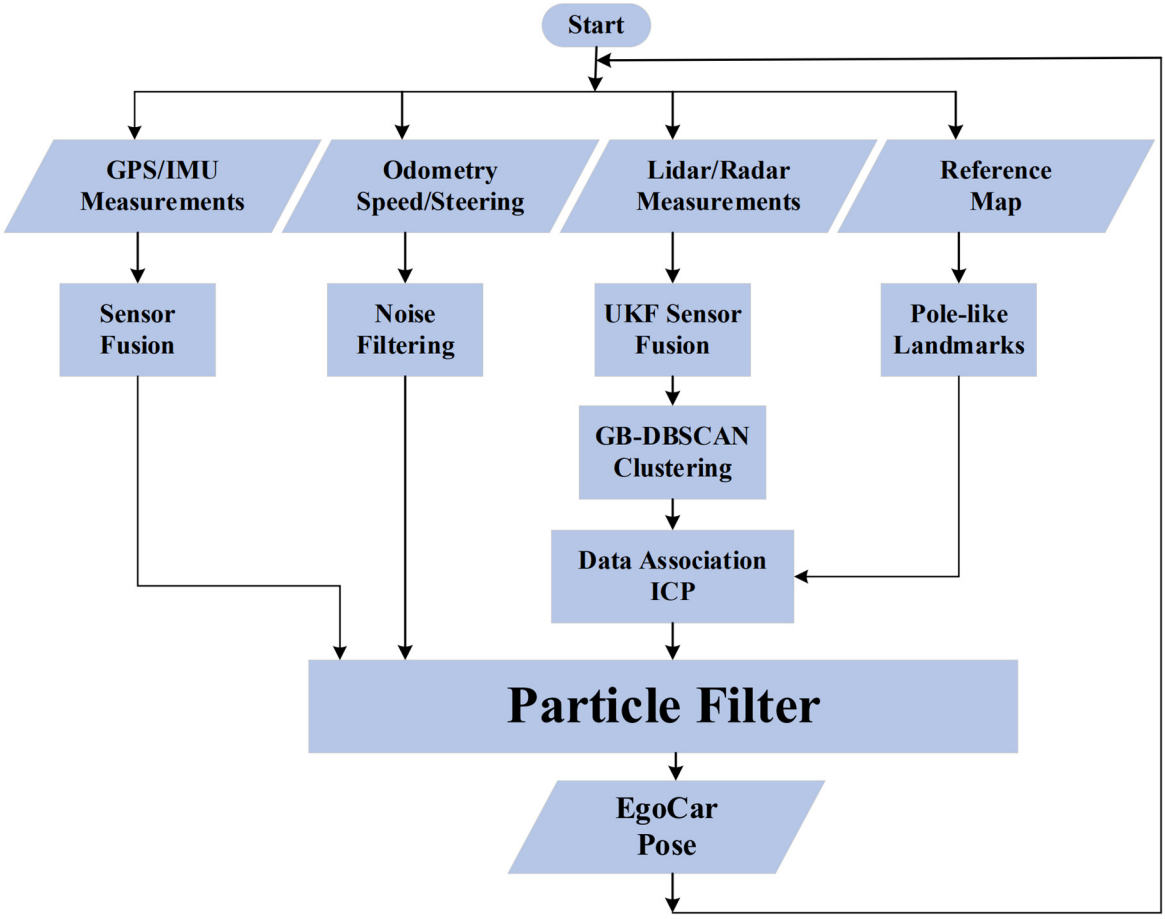

Figure 1. The RTMCL workflow.

4. Overview of the UKF

The KF is an equation-based system that works as a predictor–update cyclic optimum estimator that minimizes the estimated error covariance [44]. Given the measurement $z \in R^m$ of a discrete-time-controlled process described by a set of linear stochastic difference equations, the Kalman filter predicts the state $x \in R^n$.

Inapplicably, because KF is restricted to linear processes, it is incompatible with the radar measurement process, which is fundamentally nonlinear. To solve this restriction, the UKF was developed [45]. The UKF is a deterministic sampling-based derivative-free alternative to the extended Kalman filter [46]. The UKF also employs the predict–update two-step procedure, but it has been supplemented with additional processes such as sigma point production and prediction, as presented in Figure 2.

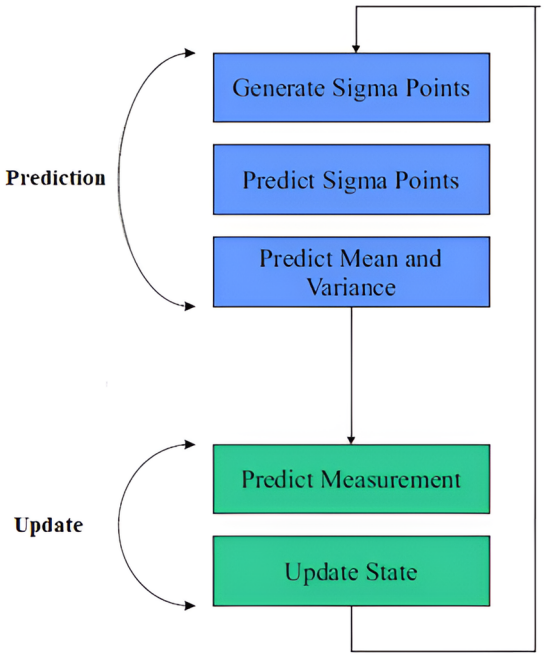

Figure 2. Workflow of the UKF.

The state Gaussian distribution is represented in the UKF process by a small number of wisely picked sample points known as sigma points. The $n_x = 2n + 1$ sigma points are chosen using the following formula:

$$X_k = \left[x_k \;\; x_k + \sqrt{(\lambda + n_x)P_k} \;\; x_k - \sqrt{(\lambda + n_x)P_k} \right] \tag{1}$$

where P_k is the Kalman filter process estimate covariance matrix, X_k is the sigma-point matrix containing n_x sigma-point vectors, and λ is a design parameter that defines the spread of the produced sigma points and often is calculated as $\lambda = 3 - n_x$.

Each produced sigma point is entered into the UKF nonlinear process model described in Equation (2) to build the matrix of predicted sigma points $\hat{X}$ with an $n \times n_x$ dimension in what is called the sigma-point prediction phase.

$$\hat{X}_{k+1} = f(X_k, \, \nu_k) \tag{2}$$

where ν_k is the white noise of the process, which is modeled as a Gaussian distribution ($\mathcal{N}$) with zero mean and covariance matrix Q_k.

The mean and covariance matrices of the predicted state are then computed from the projected sigma points using Equation (3):

$$\begin{aligned} \hat{x}_{k+1} &= \sum_{i=0}^{n_x} w_i \hat{X}_{k+1,\,i} \\ \hat{P}_{k+1} &= \textstyle\sum_{i=0}^{2n_x} w_i \left(\hat{X}_{k+1,i} - \hat{x}_{k+1} \right) \left(\hat{X}_{k+1,i} - \hat{x}_{k+1} \right)^T \end{aligned} \tag{3}$$

where w_i is the sigma-point weights that are applied to invert the sigma-point spreading. These weights are computed as follows in Equation (4):

$$\begin{aligned} w_i &= \tfrac{\lambda}{\lambda + n_x}, \;\; i = 0 \\ w_i &= \tfrac{1}{2(\lambda + n_x)}, \;\; i = 1 \ldots n_x \end{aligned} \tag{4}$$

Each produced sigma point is entered into the nonlinear UKF's measurement model described by Equation (5) to build the matrix of predicted measurement sigma points with an $n \times n_x$ dimension in the measurement prediction phase.

$$\hat{Z}_{k+1} = h\left(\hat{X}_{k+1}\right) \tag{5}$$

The mean and covariance matrices of the predicted measurement are then computed using the predicted sigma points and the covariance matrix R of the measurement noise, as shown in Equation (6):

$$\begin{gathered} \hat{z}_{k+1} = \sum_{i=0}^{n_x} w_i \hat{Z}_{k+1,\,i} \\ S_{k+1} = \sum_{i=0}^{2n_x} w_i \left(\hat{Z}_{k+1,i} - \hat{z}_{k+1}\right)\left(\hat{Z}_{k+1,i} - \hat{z}_{k+1}\right)^T + R \\ R = E\left\{\omega_k.\omega_k^T\right\} \end{gathered} \tag{6}$$

where w_i is the sigma-point weights computed in Equation (4), S_k is the measurement covariance matrix, and $E\{.\}$ Is the expected value of the measurement of white noise ω_k, which is modeled as a Gaussian distribution ($\mathcal{N}$) with zero mean and covariance matrix R.

The final stage is the UKF state update, in which the gain matrix (K) of the UKF is produced using the estimated cross-correlation matrix (T) between the sigma points in the state space and the measurement space, as in Equation (7). The gain is applied to both the UKF state vector (x) and the state covariance matrix (P).

$$\begin{gathered} T_{k+1} = \sum_{i=0}^{2n_x} w_i \left(\hat{X}_{k+1,i} - \hat{x}_{k+1}\right)\left(\hat{Z}_{k+1,i} - \hat{z}_{k+1}\right)^T \\ K_{k+1} = T_{k+1} S_{k+1}^{-1} \\ x_{k+1} = \hat{x}_{k+1} + K_{k+1}(\hat{z}_{k+1} - z_{k+1}) \\ P_{k+1} = \hat{P}_{k+1} - K_{k+1} S_{k+1} K_{k+1}^T \end{gathered} \tag{7}$$

5. Model of the Road Object

A motion model for a road object in the context of autonomous vehicles typically describes how the object's position and velocity evolve over time. The goal is to predict the future trajectory of the road object, enabling the autonomous vehicle to anticipate its movements and make informed decisions. Any object in the road is represented by five variables (state representation). These variables are grouped into one vector called the state vector x. Equation (8) presents x with its variables p_x, p_y, v, ψ, and $\dot{\psi}$, which are the object location in the x and y axes, the magnitude of the object's velocity calculated from its x and y components (v_x and v_y), the yaw angle (object orientation), and the rate at which the object-yaw angle changes, respectively, as shown in Figure 3.

$$x = \begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \end{bmatrix}, \; v = \sqrt{v_x^2 + v_y^2}, \psi = tan^{-1}\frac{v_y}{v_x} \tag{8}$$

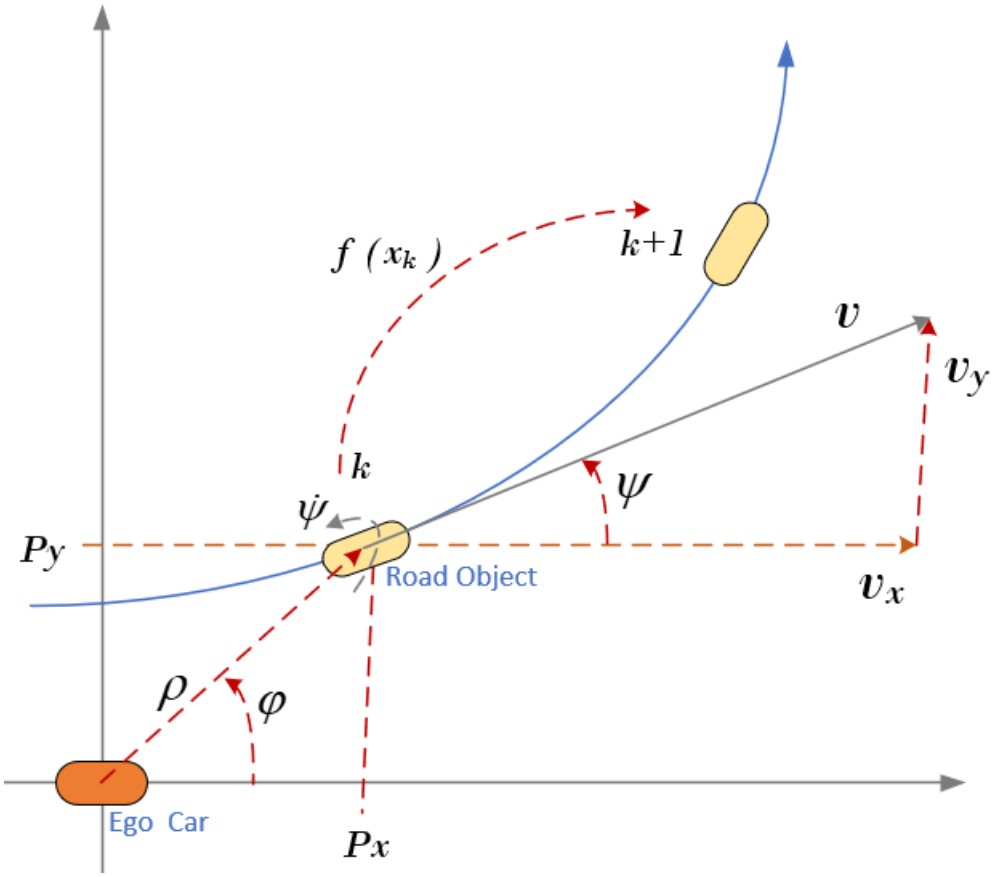

Figure 3. The motion model of an arbitrarily moving road object.

Based on the state vector x, the nonlinear $x_{k+1} = f(x_k, \nu_k)$ difference equation (dynamic equations) that represents the object's motion model is constructed and provided in Equations (9)–(11).

$$x_{k+1} = x_k + \begin{bmatrix} \frac{v_k}{\dot{\psi}_k}\left(sin\left(\psi_k + \dot{\psi}_k \Delta t\right) - sin(\psi_k)\right) \\ \frac{v_k}{\dot{\psi}_k}\left(-cos\left(\psi_k + \dot{\psi}_k \Delta t\right) + cos(\psi_k)\right) \\ 0 \\ \Delta t \\ 0 \end{bmatrix} + \nu_k \tag{9}$$

$$\nu_k = \begin{bmatrix} \frac{1}{2}(\Delta t)^2 cos(\psi_k).\nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 sin(\psi_k).\nu_{a,k} \\ \Delta t.\nu_{a,k} \\ \frac{1}{2}(\Delta t)^2.\nu_{\ddot{\psi},k} \\ \Delta t.\nu_{\ddot{\psi},k} \end{bmatrix} \tag{10}$$

$$\begin{gathered} \Delta t = t_{k+1} - t_k \\ \nu_{a,k} \sim \mathcal{N}\left(0, \sigma_a^2\right) \\ \nu_{\ddot{\psi},k} \sim \mathcal{N}\left(0, \sigma_{\ddot{\psi}}^2\right) \end{gathered} \tag{11}$$

where Δt is the time difference between two consecutive samples and is the longitudinal acceleration, $\nu_{\ddot{\psi},k}$ is the longitudinal acceleration noise (noise modeling) at sample k with a

standard deviation of σ_a^2, and $\ddot{\psi}$ is the yaw acceleration noise at sample k with a standard deviation of $\sigma_{\ddot{\psi}}^2$.

If the $\dot{\psi}$ is zero, and to avoid dividing by zero in Equation (9), the subsequent approximation is applied (linear model) to evaluate the prediction of p_x, and p_y:

$$\begin{aligned} p_{x_{k+1}} &= p_{x_k} + v_k cos(\psi_k)\Delta t \\ p_{y_{k+1}} &= p_{y_k} + v_k sin(\psi_k)\Delta t \end{aligned} \tag{12}$$

6. UKF-Based Lidar/Radar Fusion

The lidar measures the centroid of the object's position (moving or stationary) in Cartesian co-ordinates (p_x and p_y), as given by Equation (13), while the radar measures the same object's centroid position in polar co-ordinates (ρ, φ). Moreover, the radar also measures the object's velocity ($\dot{\rho}$), as given by Equation (14). Therefore, to unify the way of measurement, a mapping function is used (in Equation (15)) to convert the lidar Cartesian co-ordinates to polar form.

$$z_{lidar} = \begin{pmatrix} p_x \\ p_y \end{pmatrix}, \; z_{radar} = \begin{pmatrix} \rho \\ \varphi \\ \dot{\rho} \end{pmatrix} \tag{13}$$

$$h(x) = \begin{pmatrix} \rho \\ \varphi \\ \dot{\rho} \end{pmatrix} = \begin{pmatrix} \sqrt{p_x^2 + p_y^2} \\ arctan\left(\frac{p_y}{p_x}\right) \\ \frac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}} \end{pmatrix} \tag{14}$$

$$p_x = \rho cos(\varphi), \; p_y = \rho sin(\varphi) \tag{15}$$

As shown in Figure 4, the prediction step is conducted for both lidar and radar simultaneously. However, the update step is specific for each sensor since each sensor has its own measurement model. Moreover, the update of the belief is executed upon the arrival of a new sensor measurement (both sensors are not synchronized).

After the initialization steps for both the UKF and the measurement models for both the lidar and radar, the time step (Δt) is computed as shown in Figure 4 and, at the same time, using Equation (1), the sigma point (X_k) is created. The predicted sigma point ($\hat{X}_{k+1}$) for the next time step is calculated using Equation (2), utilizing the object's model given by Equation (9). Then, the predicted state mean vector ($\hat{x}_{k+1}$) and its covariance matrix ($\hat{P}_{k+1}$) is calculated by applying Equation (3).

For the update step in the fusion process, there are two branches. The first one is the lidar branch and the second one is the radar branch. The last received measured signal will decide which branch the workflow will assume (either lidar or radar signal). If a radar signal is received, the predicted measurement sigma point ($\hat{Z}_{k+1}$) is computed based on the model in Equation (14) and from $\hat{X}_{k+1}$ in Equation (5). Then, $\hat{z}_{k+1}$ states that the covariance S_{k+1} and the noise covariance R_{radar} are computed based on Equation (6). The R_{radar} covariance matrix is further defined as given in Equation (16) below:

$$R_{radar} = \begin{bmatrix} \sigma_\rho^2 & 0 & 0 \\ 0 & \sigma_\varphi^2 & 0 \\ 0 & 0 & \sigma_{\dot{\rho}}^2 \end{bmatrix} \tag{16}$$

where $\sigma_{\dot{\rho}}$, σ_φ, and σ_ρ are the noise SD of the object yaw rate, the noise SD of the object heading, and the noise SD of the object radial distance, respectively.

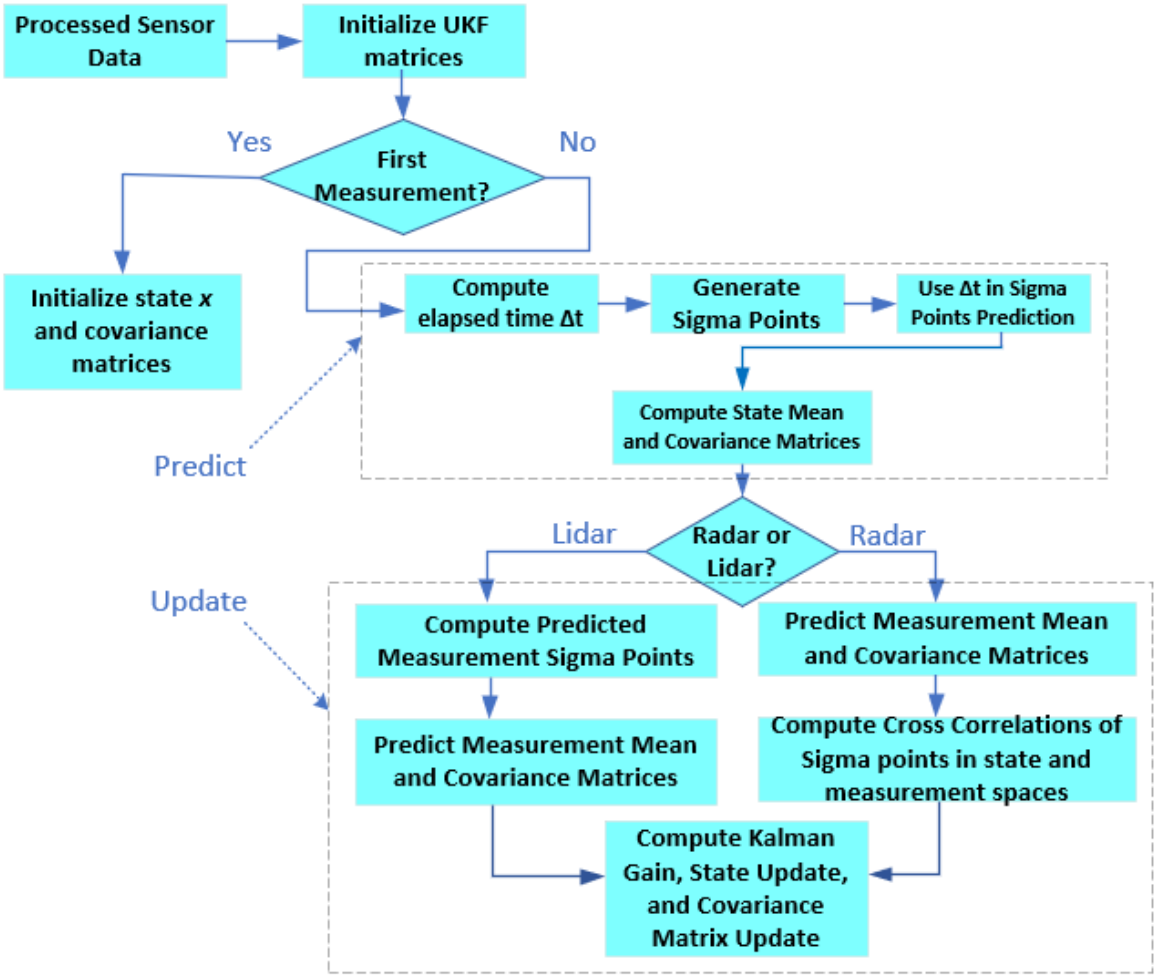

Figure 4. The workflow of the UK for lidar and radar fusion.

The cross-correlation matrix (T_{k+1}) is then computed based on the resulting state vectors $\hat{x}_{k+1}$ and $\hat{z}_{k+1}$ with their counterparts $\hat{X}_{k+1}$ and $\hat{Z}_{k+1}$, respectively using Equation (7). The T_{k+1} is then used to compute the UKF gain (K_{k+1}), which is consequently used to compute the updated state vector (x_{k+1}) and covariance matrix (P_{k+1}), as shown in Equation (7). Then, x_{k+1} and P_{k+1} will be used to generate the new sigma points (X_{k+1}) in the next iteration, etc.

Instead, if a lidar signal is received, the predicted measurement sigma point ($\hat{Z}_{k+1}$) is computed based on the linear lidar measurement model in Equation (17) and directly from $\hat{X}_{k+1}$. Then, $\hat{z}_{k+1}$ states that the covariance S_{k+1} and the noise covariance R_{lidar} are computed based on Equation (6) with more detail of R_{lidar} in Equation (17).

$$H_{lidar} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$
$$R_{lidar} = E\left[\omega.\omega^T\right] = \begin{bmatrix} \sigma_{p_x}^2 & 0 \\ 0 & \sigma_{p_y}^2 \end{bmatrix} \quad (17)$$

where the object x and y positions, σ_{p_x} and σ_{p_y}, are the noise SDs, respectively. Likewise, the radar, the cross-correlation matrix (T_{k+1}), the KF gain (K_{k+1}), and the covariance matrix (P_{k+1}) are computed from Equation (7).

7. Point-Cloud Clustering and Association

The point cloud resulting from the application of the UKF fusion algorithm presented in Figure 4 offers details about the objects in the surroundings of the egocar. To extract each object's information (geometrical shape and pose), clustering is employed on the UKF point cloud to characterize each object in a source-point model form, which will significantly lower the computation overhead and memory prerequisite.

The clustering in this work is performed using the GB-DBSCAN algorithm, which is a variation of the original DBSCAN algorithm [47], which is an unsupervised learning algorithm that arranges data points with high density together in one group. Two parameters are used to tune the DBSCAN algorithm and define the allowed density. "ε" is the first parameter to tune, which defines the allowed radial distance from the point under evaluation "p". "*minPts*" is the second parameter that determines the lowest number of detection points that are located within a distance "ε" from "p", including "p" itself, to form a cluster. Therefore, the determination of the density of points to be grouped together to form a cluster is conducted by the proper selection of "ε" and "*minPts*". However, if the objects to be detected take several topologies (other than circular), as with the case of road objects, the DBSCAN algorithm is not enough. An improvement came from Dietmayer et al. [48] by proposing not to use fixed parameters like "ε" and "*minPts*" in the GB-DBSCAN but, instead, forming a polar grid taking into account the angular and radial resolution of the sensor. The search area is not necessarily a circular one with a fixed radius but can be a dynamic elliptic one. Therefore, this algorithm is very appropriate for pole-like objects, as their distinctive feature is that they produce a high density of detection points, much more than their surroundings.

The outcome of the GB-DBSCAN algorithm is a coarse clustering of the UKF fusion data. The application of the RANSAC algorithm fine-tunes these clusters and associates geometrical shape proposals with them [36]. For the pole-like landmarks, the most appropriate geometrical shape is the circular one, which is fitted to all the N points in each cluster. The RANSAC algorithm then finds the parameters of each fitted circle (the radius ($\hat{r}$) and the centroid ($\hat{x}_c, \hat{y}_c$)), which will represent a pole landmark, by finding the optimal solution of the following formula:

$$\min\left\{\frac{1}{N}\sum_{i=1}^{N}\left[\sqrt{(x_i-\hat{x}_c)^2+(y_i-\hat{y}_c)^2}-\hat{r}\right]^2\right\} \tag{18}$$

Once the pole-like landmarks are identified in the source UKF fusion data, the data association step takes place by linking these identified landmarks with their matched counterpart in the target (the supplied reference map). The employed PF relies heavily on precision in this step. The association is implemented using the ICP algorithm [42]. The standard ICP searches the whole point clouds in both the source and the target; however, here, only the centroids are considered during the objects' matching process, which, in return, saves a significant amount of memory and processing overhead.

The ICP algorithm is composed of two phases that run iteratively till convergence is reached. In our case, we have a set X representing the source points (UKF data) and another set Y representing the target points (a point-cloud map). The first phase is the matching between each point in X with the closest point in Y. The second phase is to find the optimal transform (X→Y) given the matched association. Storing the X and Y sets in the form of a KD tree data structure [49] is very crucial for the efficient matching by distance performed by the ICP. x_i and y_i are two matched points from X and Y sets, respectively. The 2D ICP algorithm finds the rotation angle "φ" and the translation parameter "t" that minimizes the summation of the quadratic distance between the target and the source points, as presented by Equation (19). Note that the rotation matrix $R(\varphi)$ uses angle "φ" as a variable.

$$min_{\varphi,t}\left\{\sum_{i=1}^{N}(y_i-(R(\varphi)x_i-t))^T(y_i-(R(\varphi)x_i-t))\right\} \tag{19}$$

The co-ordinate system must be unified between the egocar co-ordinates (x_c and y_c) and the reference map co-ordinates (x_m and y_m) to perform the data association correctly. Therefore, the homogenous transformation is used (in the form of a transformation matrix) as given by Equation (20). The translation and rotation are performed using map particle/egocar co-ordinates (x_p and y_p) and the rotation angle θ.

$$\begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & x_p \\ sin\theta & cos\theta & y_p \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} \tag{20}$$

8. Details of the Particle Filter

For a stochastic process that has noisy observations $p(z_t|x_t)$, the posterior distribution $bel(x_t)$ could be represented by a finite set of particles. This is considered an approximate implementation of the Bayesian filter in a recursive mode with a normalization factor ζ that is given by Equation (21):

$$bel(x_t) \leftarrow \zeta \; p(z_t|x_t) \; bel(x_{t-1}) \tag{21}$$

The higher the particles' number M, the more accurate the representation of the belief distribution $bel(x_t)$, where t denotes the time step at which the state of the set of the particles is considered, as given by Equation (22):

$$\chi_t = \left\{ x_t^{[i]} \middle| 1 \leq i \leq M \right\} \tag{22}$$

The actual state (the optimal solution) will be one of the particles included in the set χ_t at time t. Therefore, each $x_t^{[i]}$ represents a hypothesis for the optimal solution. The implementation of the employed PF in the work is presented in Algorithm 1. Moreover, the workflow of the PF is illustrated in Figure 5. Upon receipt of a new measurement (odometry) signal (u_t) or a pole-like object measurement update (z_t) from the UKF, a new search for an optimum egocar pose is initiated.

Algorithm 1. Workflow of the employed PF.

Procedure Particle Filter (χ_{t-1}, u_t, z_t):

Input: Set of particles χ_{t-1} at a time $(t-1)$, control inputs u_t, and a set of measurements z_t.
Output: The updated set of particles χ_t at time t.

Begin

1. Initialize Particles: $\overline{\chi}_t = \chi_t = \varnothing$.
2. *For* $m = 1$ *to* M *do*
   i. *generate* $x_i^{[m]} \sim p\left(x_t \middle| u_t, x_{t-1}^{[m]}\right)$
   ii. $w_t^{[m]} = p\left(z_t \middle| x_i^{[m]}\right)$
   iii. $\overline{\chi}_t = \overline{\chi}_t + x_i^{[m]}, w_t^{[m]}$
   iv. End for loop
3. *For* $m = 1$ *to* M *do*
   i. *draw i with probability* $\propto w_t^{[i]}$
   ii. *add* $x_t^{[i]}$ *to* χ_t
   iii. End for loop
4. Return χ_t

End.

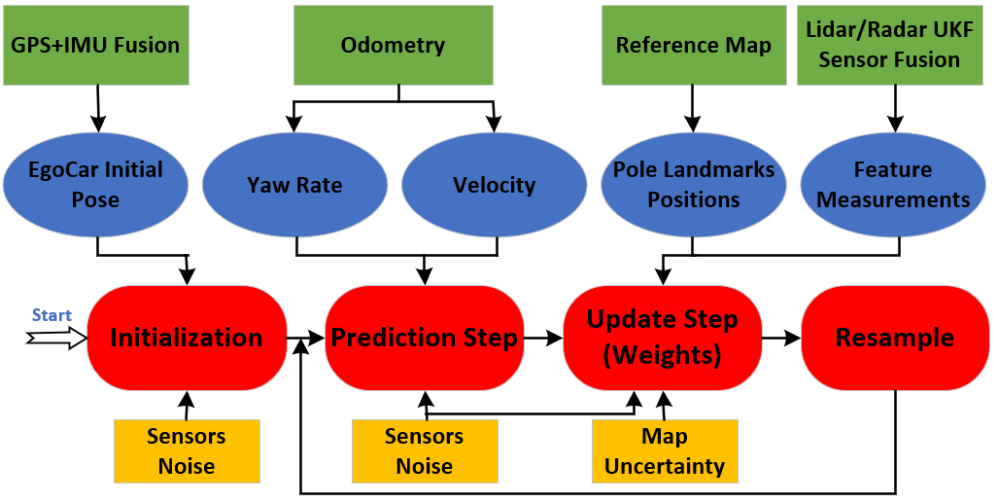

Figure 5. Overview and a flowchart of the particle filter algorithm.

The PF implementation is a predict–update cycle, in which the prediction portion of the cycle is implemented by step (2) in the procedure. For each particle (out of the M randomly generated particles at the initiation phase of the procedure), two values are calculated for each particle (which represents a possible egocar); the first one is the state hypothesis ($x_i^{[m]}$), and the second one is the state transition distribution $p\left(x_t \middle| u_t, x_{t-1}^{[m]}\right)$, which is calculated using the object's motion model explained in Section 5. A weight is created for each particle based on its generated state hypothesis (importance) among the other state hypotheses of the other particles, as given in Equation (23). Each weight takes the form of a multivariate Gaussian probability density function that is computed for each observation and the likelihood of all the observations are combined by taking their multiplication product.

$$w_t^{[m]} = \prod_{i=1}^{N} \frac{exp\left(-\frac{1}{2}\left(z_i^{[t]} - \mu_i^{[t]}\right)^T \Sigma^{-1}\left(z_i^{[t]} - \mu_i^{[t]}\right)\right)}{\sqrt{|2\pi\Sigma|}} \tag{23}$$

where N is the count of measurements for particle m, Σ is the measurements covariance matrix, $\mu_i^{[t]}$ is the state mean predicted measurement for the pole corresponding to the ith observation at step t, and $z_i^{[t]}$ is the ith pole observation for particle m at step t.

After that, the set of particles ($\overline{\chi}_t$) is resampled proportional to the previously produced weights ($\alpha w_t^{[i]}$), where α is a normalization coefficient, generating new M particles and, hence, χ_t (the updated posterior approximation) is produced. χ_t will usually contain the strongest particles with multiple copies replacing the weak particles that are left out (have small weights (less important)). And, so on, the algorithm continues till χ_t will contain M copies of one particle (convergence) that represents the solution or the required egocar pose.

The convergence indicator for the PF is the weighted mean error ($Error_{weighted}$) computed over all the particles, as shown in Equation (24). $Error_{weighted}$ is achieved by calculating the RMSE between the ground truth g and the state of each particle p_i and multiplied by its weight, then summing the product for all particles, and dividing the result by the summation of all the weights.

$$Error_{weighted} = \frac{\sum_{i=1}^{M} w_i \sqrt{|p_i - g|}}{\sum_{i=1}^{M} w_i} \tag{24}$$

9. Realization of the RTMCL

The development and coding of the RTMCL are accomplished using C++ programming language [50], as it is well known for its high performance, especially for real-time applications [51]. The developed code runs on the Ubuntu Linux operating system [52]. Moreover, the development used the efficient numerical solving package Eigen, which is used to perform all the vector and matrix computations involved in the execution of the objects' model and the prediction and update steps [53].

The sensor data are processed using the NVIDIA DRIVE AGX platform. The lidar is Velodyne Lidar VLP-16 (16 channels, 100 m range, 360° horizontal field of view (FoV), 30° vertical FoV, 300,000 points/sec) and the radar is Continental ARS430 (77 GHz, 250 m range, wide azimuth and elevation coverage, 0.1 m accuracy).

In order to sensitively execute the motion models for several objects (furnished by Equations (9)–(11)), various noise parameters must be carefully determined. Their values are fine-tuned and set and presented in Table 1.

Table 1. The noise parameters of the object model, PF, and UKF.

Parameter	UKF/PF	Parameter	UKF
σ_a m/sec^2	1.0	σ_{p_y} (lidar) m	0.15
$\sigma_{\ddot{\psi}}$ rad/sec^2	0.6	σ_ρ (radar) m	0.3
$\sigma_{\dot{\psi}}$ rad/sec	0.06	σ_φ (radar) rad	0.03
σ_{p_x} (lidar) m	0.15	$\sigma_{\dot{\rho}}$ (radar) m/sec	0.3

To evaluate the consistency of the UKF design, the NIS measure that is averaged over time is employed [54] to fine-tune the above noise parameters. Moreover, to make sure the UKF design is unbiased and consistent, the estimation error is aggregated and its mean value is calculated. This value should be around zero, besides the UKF's actual MSE corresponding to the UKF-computed state covariance. Equation (25) calculates the NIS value at each time sample k and then uses a moving N-sample window of measurements to compute its average value ($NIS_{Average}$).

$$\begin{gathered} NIS_k = (z_{k+1} - \hat{z}_k)^T S_k^{-1} (z_{k+1} - \hat{z}_k) \\ NIS_{Average} = \frac{1}{N} \sum_{k=1}^{k=N} NIS_k \end{gathered} \tag{25}$$

The UKF performance depends heavily on how properly it is initialized [54]. The estimated state vector (x) and its estimated state covariance matrix (P) are considered the most important initialized variables. p_x and p_y (the first and second terms of x in Equation (8)) are simply initialized by associating them to the early obtained unrefined sensor measurements. For the following three terms of the x, trial-and-error endeavors in addition to some intuition are used to initialize these variables, as given in Table 2. Moreover, the P matrix is constructed as a diagonal matrix, as given in Equation (26), and includes the covariance values of the estimate of each term in x.

$$P = diag\left(\sigma_{\hat{p}_x}^2, \sigma_{\hat{p}_y}^2, \sigma_{\hat{v}}^2, \sigma_{\hat{\psi}}^2, \sigma_{\hat{\dot{\psi}}}^2\right) \tag{26}$$

Table 2. UKF state variable initialization.

Parameter	UKF	Parameter	UKF
p_x m	1st raw x-reading	p_y m	1st raw y-reading
v m/sec	0.0	ψ rad	0.0
$\dot{\psi}$ rad/sec	0.0	$\sigma_{\hat{p}_x}$ m	1.0
$\sigma_{\hat{p}_y}$ m	1.0	$\sigma_{\hat{v}}$ m/sec	$\sqrt{1000}$
$\sigma_{\hat{\psi}}$ rad	$\sqrt{1000}$	$\sigma_{\hat{\dot{\psi}}}$ m/sec^2	$\sqrt{1000}$

The calculation of the RMSE, as given in Equation (27), is used to evaluate the performance of the UKF, which is defined as how close the estimated ranges are from the true ranges (the ground truth). An N-sample moving window of estimates is used to calculate this metric.

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{k=N}\left(x_k^{est} - x_k^{true}\right)^2} \tag{27}$$

where x_k^{true} is the ground-truth state vector generated by the motion driving simulator [55] or supplied as training data during the UKF design phase and x_k^{est} is the UKF's output (the estimated state vector).

Regarding the PF, like the UKF, the proper initialization is very critical for its successful execution. The initialization process is as follows:

(a) The PF particles' count M usually falls in the range from 100 to 1000 [20], as per the literature [43]. The higher the particles' count, the higher the accuracy but the slower the speed of computation, and vice versa. Therefore, a compromise must be made. After many trials, $M = 50$ is selected after showing approved real-time performance with the required precision.

(b) The result of the GPS/IMU fusion is the initial pose of the egocar ($p_{x_{GPS}}$, $p_{y_{GPS}}$, θ_{GPS}), which will be used by the PF in the initialization of all the M particles' state vectors as follows:

$$\begin{aligned} p_x^{[m]} &\sim \mathcal{N}\left(p_{x_{GPS}}, \sigma^2_{x_{GPS}} + \sigma^2_{x_{artificial}}\right) \\ p_y^{[m]} &\sim \mathcal{N}\left(p_{y_{GPS}}, \sigma^2_{y_{GPS}} + \sigma^2_{y_{artificial}}\right) \\ \theta_{Particle}^{[m]} &\sim \mathcal{N}\left(\theta_{GPS}, \sigma^2_{\theta_{GPS}} + \sigma^2_{\theta_{artificial}}\right) \end{aligned} \tag{28}$$

where the particle m initialized pose is represented by $p_x^{[m]}$, $p_y^{[m]}$, and $\theta_{Particle}^{[m]}$. The standard deviations of the noise of the GPS/IMU fusion are $\sigma_{x_{GPS}}$, $\sigma_{y_{GPS}}$, and $\sigma_{\theta_{GPS}}$. Moreover, the artificial noise amounts added to pose variables are $\sigma_{x_{artificial}}$, $\sigma_{y_{artificial}}$, and $\sigma_{\theta_{artificial}}$. They are used to add randomization to these variables to improve the chances of convergence of the PF. Table 3 lists the values used to initialize these parameters.

(c) The particles' weights that value their importance use the uniform distribution $w^{[m]} = \frac{1}{M}$ for initialization.

(d) The landmarks that take a pole shape in the reference map and being used by the RTMCL algorithm are represented by $\mathcal{N}\left(p_{x_{Pole}}, \sigma^2_{x_{Pole}}\right)$ and $\mathcal{N}\left(p_{y_{Pole}}, \sigma^2_{y_{Pole}}\right)$, which are Gaussian distributions for both x and y positions, respectively. These distributions are used to model these positions' uncertainties. The standard deviation $\sigma_{x_{pole}}$ and $\sigma_{y_{pole}}$ values are shown in Table 3.

Table 3. Values for the PF initialized parameters.

Parameter	PF	Parameter	PF
$\sigma_{x_{GPS}}$	0.3 m	$\sigma_{x_{artificial}}$	10 m
$\sigma_{y_{GPS}}$	0.3 m	$\sigma_{y_{artificial}}$	10 m
$\sigma_{\theta_{GPS}}$	0.01 rad	$\sigma_{\theta_{artificial}}$	0.05 rad
$\sigma_{x_{pole}}$	0.3 m	$\sigma_{y_{pole}}$	0.3 m

Equation (29) presents the aggregated mean absolute error (MAE) of each estimated pose variable compared to the ground truth. This metric is used in order to evaluate the performance of the PF. It is computed by employing an N-measurement window that moves across the incoming pose estimates.

$$\begin{aligned} X_{error} &= \frac{1}{N}\sum_{i=1}^{N}\left|x_i^{best} - x_i^{gt}\right| \\ Y_{error} &= \frac{1}{N}\sum_{i=1}^{N}\left|y_i^{best} - y_i^{gt}\right| \\ Yaw_{error} &= \frac{1}{N}\sum_{i=1}^{N}\left|\theta_i^{best} - \theta_i^{gt}\right| \end{aligned} \tag{29}$$

where x_i^{gt}, y_i^{gt}, and θ_i^{gt} are the variables of the ground truth that are generated by the motion driving simulator [56] or provided as training data throughout the particle filter design phase, and x_i^{best}, y_i^{best}, and θ_i^{best} are the best-estimated variables of the PF particles' poses.

10. Testing and Evaluation Results

To fine-tune the hyperparameters of the RTMCL algorithm, broad trial-and-error tryouts have been carried out. For proper assessment, numerical KPIs are proposed and implemented as given by Equations (24), (26), and (28) to assess the RTMCL under several hyperparameter configurations.

Moreover, through an iterative tuning process, the performance of the RTMCL is evaluated on several testing tracks while under various sets of hyperparameters. Figure 6 presents an example of one of the employed testing tracks. The length of this track is 754 m and has curvatures at several points. It also contains 42 landmarks that resemble poles to sufficiently emulate an urban driving environment.

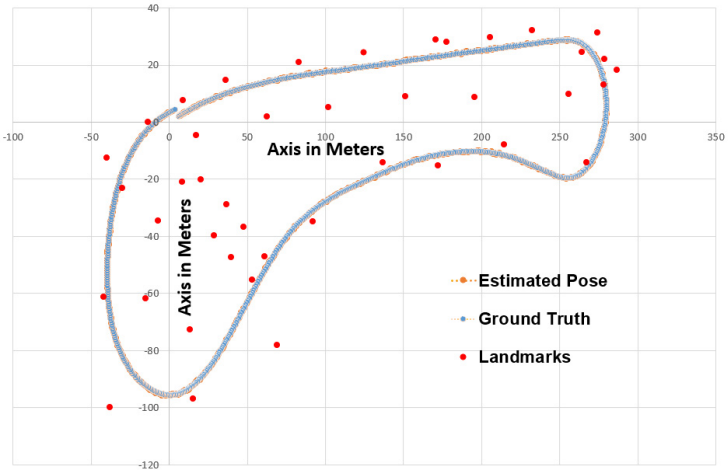

Figure 6. Egocar localization outcomes in the shown test track.

The results of UKF testing performing the fusion of the lidar/radar are presented in Table 4. Several objects are detected, including pole-like landmarks, cars, cyclists, and pedestrians. The five state variables, p_x, p_y, v_x, v_y, and ψ, are measured and their *RMSE* are

listed in Table 5 as a KPI (Equation (26)). A comparison between the ground truth of each state variable to the estimated value of this state, then obtaining the error is performed by this KPI. Better detection is achieved with a lower value of this KPI.

Table 4. The UKF's performance assessment.

State Var	Cyclist	Car	Pedestrian	Pole
p_x	0.0648	0.1857	0.0652	0.0324
p_y	0.0809	0.1899	0.0605	0.0433
v_x	0.1452	0.4745	0.5332	0.0032
v_y	0.1592	0.5075	0.5442	0.0054
ψ	0.0392	0.2580	0.2075	0.0075

Table 5. Assessment of the UKF performance for lidar/radar sensor fusion.

	Lidar + Radar	Lidar Only	Radar Only
RMSE-p_x	0.0648	0.1612	0.2031
RMSE-p_y	0.0809	0.1464	0.2539
RMSE-v_x	0.1452	0.2082	0.1971
RMSE-v_y	0.1592	0.2129	0.1871
RMSE-ψ	0.0392	0.0540	0.0480
NIS-Average	2.2797	1.6941	2.6576
NIS-Min	0.0012	0.04874	0.11309
NIS-Max	14.749	12.997	12.183
NIS > 95% Threshold	2.2%	3.2%	5.2%

The UKF is employed and tested in three different ways: the first one with only lidar signals, the second way with only radar signals, and the third one with the fusion between the lidar and the radar. This way of testing evaluates how significant the fusion is for the accuracy of object detection and tracking. Table 5 presents the results of the three ways of testing on the bicycle track. It is clear how significant the fusion is at all pose variables, all of them have much better RMSE. Table 5 clearly shows that fusion reduces the RMSE for all the pose state variables and makes a big difference in the accuracy of detection. As an example, the error of the detection position in the x-axis (p_x) is lowered by 60% compared to the one with the lidar alone and 70% compared to the one with radar alone. Another example is the error of the velocity detection in the x-axis (v_x) is lowered by 30% more than the one with the lidar alone and 26% more than the one with radar alone. Furthermore, the NIS KPI is computed as well for the three previous cases, showing that the fusion has significantly improved the UKF's consistency. The fusion NIS quantities that are higher than the threshold of 95% have been lowered by 31% compared to the "only lidar" and 38.5% compared to the "only radar" values.

An example of testing the whole pipeline of the RTMCL, which includes the combination of the PF and UKF with the employment of the probabilistic reference map, Figures 6–8 present the simulation results of egocar driving on the test track in Figure 6. The GB-DBSCAN is used for the data association step for the localization of the egocar on the global map. Figures 7 and 8 show both the ground truth and the estimated egocar pose values drawn overlapping each other due to the computed small errors as reported in Table 6. The PF performance is evaluated using the RMSE KPI given in Equation (28). This KPI is computed for the three variables of the pose (x, y, and θ). It evaluates each of these estimated values against its ground truth and aggregates the resulting errors. The performance of the PF is better if these KPI values are lower. Several counts of particles

are used to choose the most appropriate configuration for the employed PF and to test its performance in real time. Hence, Table 6 shows that the minimum acceptable particle count is around 25 particles. Lower than this number, the PF starts to divert. However, to observe more robustness and improve further the accuracy, 50 particles are selected in this work to be employed by the PF, as higher numbers will not add much to the accuracy at the expense of much higher execution time. The count of 50 particles is considered a good balance between robustness and execution cost (real-time performance). The selection of 50 particles shows convergence in all the extensive performed tryouts.

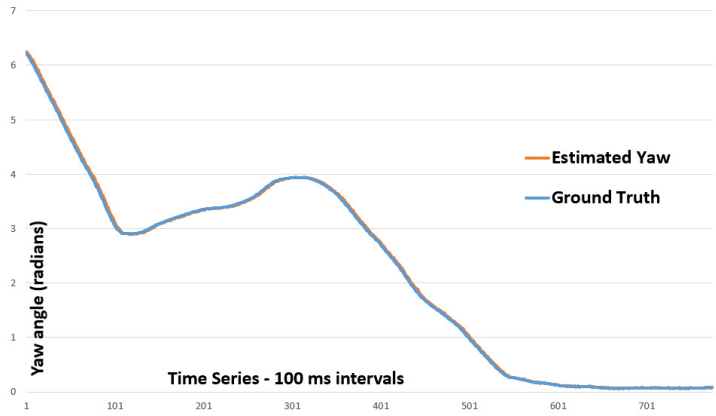

Figure 7. Egocar orientation (yaw angle) estimation and ground truth in the testing track.

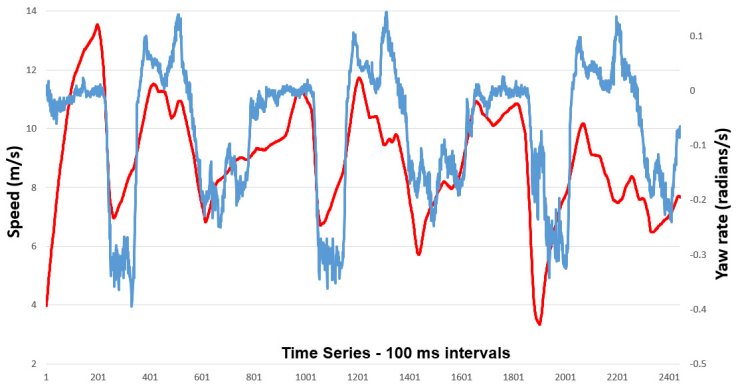

Figure 8. Performance of the yaw rate and speed of the egocar through a $3-$ lap driving on the testing track (blue is the yaw rate and red is the speed).

Table 6. The PF uses numerous counts of particles.

# Particles	x-Error	y-Error	Yaw-Error	Exec. Time
15	122.34	33.002	1.5959	0.268 ms
25	0.1382	0.1240	0.0048	0.486 ms
50	0.1143	0.1154	0.0040	0.739 ms
100	0.1154	0.1071	0.0037	1.224 ms
150	0.1098	0.1060	0.0037	2.086 ms
200	0.1102	0.1039	0.0036	2.403 ms

Red is the worst-case scenario, green is the best-case, and # is no. of.

The uncertainties in the reference map are modeled by the standard deviations associated with pole-like landmarks. The results of using the uncertainties in the map positions are reported in Table 7. Several standard deviations are tested and show how significantly

it affects the final egocar pose estimation produced by the PF. The table shows that the RTMCL can handle a significant amount of uncertainty in the landmark position, which can reach up to 2.0 m or $2\sigma_{pole}$, and the computed error in the egocar localization is still below 30 cm.

Table 7. The influence of the landmark standard deviation.

$\sigma_{x_{pole}}$	$\sigma_{y_{pole}}$	x-Error	y-Error	Yaw-Error
0.3	0.3	0.1143	0.1154	0.0040
0.5	0.5	0.1730	0.1633	0.0057
1.0	1.0	0.2926	0.2736	0.0098

The performance of the PF during the start-up phase is very crucial for its convergence, stability, and performance later on. Therefore, Figure 9 presents the start-up segment of the PF employing 50 particles till it becomes stabilized (after around 100 time steps). As a sign or indicator of the robustness of the PF performance, Figure 10 presents the computed particles' weights throughout a single lap driving on the test track. The best weight at each time step is significantly higher than the average weight. This is considered a good indicator of robustness [57].

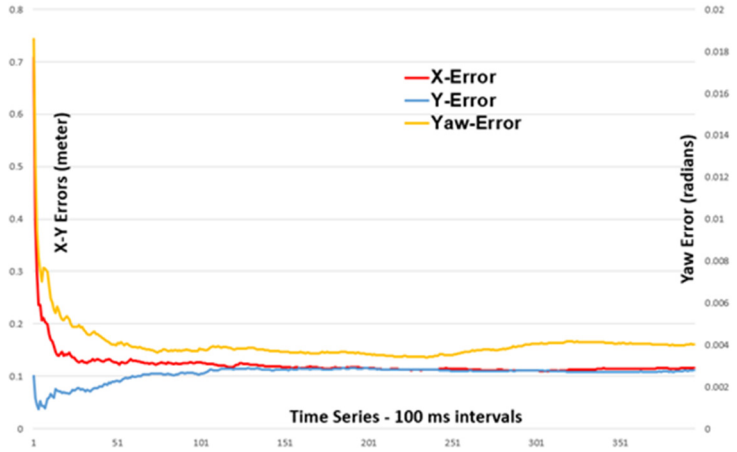

Figure 9. Errors decay throughout the PF start-up phase.

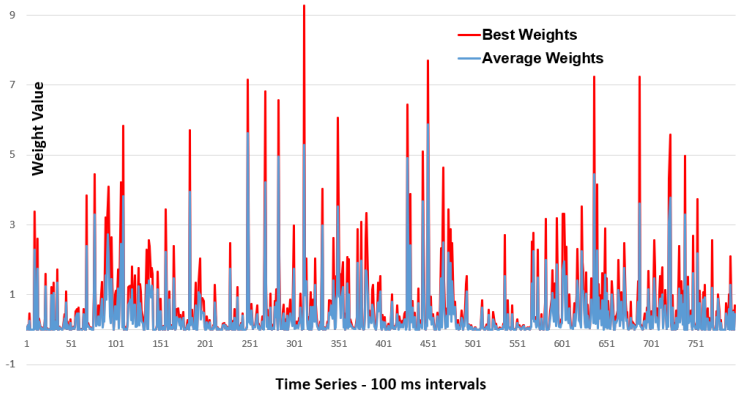

Figure 10. The distribution of particles' weights during a single-lap tour by the egocar.

It is observed that there is an inverse relation between these weight values (best and average) and the number of detected poles. Equation (30) below introduces another streamlined form of Equation (22), where the values of the weights are computed as the multiplication of the likelihoods of the observation of each pole-like landmark represented

by a multivariate Gaussian probability density function. There is a suitable number of observed poles. If the number of observed poles is high, there is a chance that one or more of these poles have small Gaussian probability density values that can bring the whole product down and, consequently, the associated weight. After many tryouts, it is found that the most appropriate count of identified poles at each time step for the smooth operation of the RTMCL is in the scope of 4 to 12 poles as shown in Figure 11.

$$w_t^{[m]} = \prod_{j=1}^{N} \frac{exp\left(-\frac{\left(z_{x_j}^{[t]}-\mu_{x_j}^{[t]}\right)}{2\sigma_{x_{pole}}^2} - \frac{\left(z_{y_j}^{[t]}-\mu_{y_j}^{[t]}\right)}{2\sigma_{y_{pole}}^2}\right)}{2\pi\sigma_{x_{pole}}\sigma_{y_{pole}}} \tag{30}$$

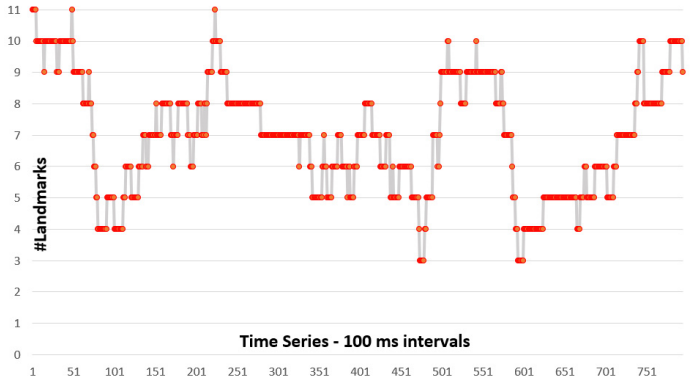

Figure 11. The distribution of detected poles (red dots) during a single-lap touring by the egocar.

The real-time performance of the RTMCL is tested using extensive experimentation and tryouts proving that its execution is fast enough. Table 8 presents the measurements of the execution times of several tasks within the RTMCL pipeline on a very moderate computational platform: an Intel 1.6-GHz Core i5 with 8 GB of RAM. These measurements are collected for a single estimation of the egocar pose based on the 12-pole detection.

Table 8. Details of an individual pose estimation processing time by the RTMCL.

Process	Execution Time
State estimation by the UKF for 12 poles	12 × 439 µs
Clustering using GB-DBSCAN & RANSAC + data association using the ICP.	835 µs
Pose estimation by the Particle Filter	739 µs
Overhead by control tasks—20%	1368 µs
The sum of execution times	8210 µs

It is recommended from the literature that the localization pipeline iterates at a speed of 10 Hz to 30 Hz. Accordingly, the measured RTMCL execution time per iteration, which is 8.2 ms (122 Hz) in Table 8, is considered very well suited at even the upper limit of the recommendation.

Moreover, the values in Table 8 show that more than 50 pole detections can be employed and, still, the UKF is capable of meeting the requirements of collecting the lidar and radar measurements at a rate of 30 Hz (33.3 ms cycle) according to [1]. All the above analyses show that the real-time performance of the RTMCL is very convenient, allowing its robustness to be improved by either increasing the number of particles or allowing more pole detections.

Moreover, the results achieved in the work are significantly advantageous if compared with other works in the literature. The mean error for both the lateral and longitudinal

positions (11 cm) is less than the 20 cm obtained in [58] using a pole-based reference map, stereo camera system as a main sensor, GPS as a secondary sensor, particle filter, and Kalman filter.

Additionally, the achieved mean error by RTMCL is also lower than the one achieved by [35], which is 16.4 cm (with an absolute max. error of 99.6 cm), obtained using a pole-based mapping, a lidar fused with RTK-GPS, IMU, wheel encoder, a Bayesian filter, and the egocar's CTRV motion model.

Furthermore, the RTMCL outperforms the localization system proposed by [41], which fuses a digital map, GPS, and IMU, along with lanes and symbolic road markings (SRMs) recognized by a front camera via a particle filter. The latter-mentioned system produced a 95 cm longitudinal error and a 49 cm lateral error. The full comparative analysis is presented in Table 9 below.

Table 9. Comparing the proposed approach with other stand-out methods.

Comparison Aspect	This Work (RTMCL)	(Spangenberg et al. [58])	(Weng et al. [35])	(Suhr et al. [41])
Mean Error (Lateral)	11 cm	20 cm	16.4 cm (Max: 99.6 cm)	49 cm
Mean Error (Longitudinal)	11 cm	20 cm	16.4 cm (Max: 99.6 cm)	95 cm
Sensor Configuration	GPS, IMU, radar, lidar, particle filter, Kalman filter.	Stereo camera, GPS, particle filter, Kalman filter	Lidar, RTK-GPS, IMU, wheel encoder, Bayesian filter.	Digital map, GPS, IMU, front camera.
Localization Approach	Real-Time Monte Carlo, Pole-based reference map, particle filter, Kalman filter.	Pole-based reference map, stereo camera, GPS, particle filter, Kalman filter.	Pole-based mapping, lidar, RTK-GPS, IMU, wheel encoder, Bayesian filter.	Digital map, GPS, IMU, front camera, particle filter.
Results Comparison	Significantly advantageous	20 cm mean error	16.4 cm mean error (Max: 99.6 cm)	95 cm longitudinal, 49 cm lateral error

Moreover, the proposed RTMCL is compared to that of other research work based on fusion techniques, as summarized in Table 10. The first work discusses Gao et al.'s deep learning method for autonomous vehicle object detection [59], emphasizing the fusion of vision and lidar data to enhance classification accuracy. It highlights the upsampling of lidar point clouds, conversion into a depth feature map, and integration with RGB data for CNN input. The study employs a vehicle-mounted camera and lidar, using NVIDIA® GeForce GTX Titan X and NVIDIA® Jetson TX1 (NVIDIA, Santa Clara, CA, USA) for offline detection and classification on a public dataset.

The second work outlines Gao et al.'s integrated framework for predicting cyclist trajectories at unsignalized intersections [60]. It focuses on intent inference using a Dynamic Bayesian Network (DBN) considering motion, ego vehicle, and environmental features. The approach employs LSTM with encoder–decoder for online trajectory prediction, achieving predictions within 0.9 s of entering the intersection. The text underscores the approach's outperformance over baseline methods (KF and DBN + KF) across various prediction horizons and its potential benefits for intelligent vehicles in road-user protection and path planning. Future research considerations include addressing cyclist–ego vehicle interaction and enhancing method robustness and interpretability.

Table 10. Comparing the proposed approach with work based on other fusion techniques.

Aspect	Gao et al. [59]	Gao et al. [60]	Proposed Approach
Topic	Autonomous vehicle object detection.	Cyclist trajectory prediction.	Real-Time Monte Carlo Localization (RTMCL) for autonomous vehicles.
Methodology	Deep learning combining vision and lidar data.	Integrated framework with DBN and LSTM for cyclist intent prediction.	RTMCL with GPS, IMU, radar, and lidar data, incorporating clustering and particle filter techniques.
Key Techniques	Upsampling lidar, depth feature map, CNN.	DBN, LSTM with encoder-decoder, clustering algorithms	Particle filter, UKF, clustering algorithms, ICP.
Performance Metrics	Improved classification accuracy using fusion.	Predicts cyclist intentions within 0.9 s, and outperforms baseline methods.	Achieves 11 cm mean error, and handles uncertainties in pole-like landmark positions.
Implementation	Vehicle-mounted camera and lidar, NVIDIA® GeForce GTX Titan X, and NVIDIA® Jetson TX1	Lidar (Velodyne VLP-16, 10 fps) and a mono camera (IDS UI-5250CP-C-HQ, 15 fps).	Implemented in C++ on Intel 1.6-GHz Core i5 with 8 GB of RAM, real-time performance demonstrated.
Future Directions	Conduct real-world experiments based on a vehicle-mounted domain controller.	Explore cyclist-ego vehicle interaction, enhance robustness and interoperability	Add a front camera, incorporate additional road objects, and explore machine-learning approaches for improved localization.

11. Conclusions

The proposed real-time Mont Carlo Localization (RTMCL) approach for self-driving vehicles is a pipeline of tasks that starts with the fusion between the GPS and the IMU to produce an unrefined egocar pose that is used to initialize the employed particle filter. Then, in the next task, a tailored UKF is used to fuse the collected data from the installed radar and lidar sensors on the egocar. The output of the UKF contains information about objects in the surrounding of the egocar and, thus, needs further processing to detect these objects and identify them. This task is carried out using the clustering algorithms GB-DBSCAN and RANSAC then produces the poses of the detected pole-like landmarks. In another task. The ICP algorithm is employed to perform the association between the identified poles in the previous task to the ones embedded in the provided reference map. In the final task, a tailored particle filter is developed and all the outputs of the previous tasks generate the fine-tuned egocar pose as the final localization measured reference to the co-ordinates of the global map.

The implementation of the whole RTMCL is conducted using C++ to ensure real-time performance. The testing results show that the RTMCL reached an accuracy of around 11 cm of mean error using merely 50 particles. The recorded execution times on affordable CPUs show the number of particles can further increase without much effect on the real-time performance.

Furthermore, testing the pipeline of the RTMCL has shown that it can run smoothly at 30 Hz while handling up to 50 poles at a time. Moreover, up to 2.0 m in uncertainty in the pose positions of the pole-like landmarks in the reference map can be handled successfully by the RTMCL, as its estimated egocar pose error will not exceed 30 cm in such a case.

It is planned to supplement a front camera as well as range finders [61] to the current fusion approach in the future and further examine the returns they will contribute to overall localization performance [62]. In addition, the RTMCL will be supplemented with additional road objects such as curbs, sidewalks, guardrails, and intersection features, among others. It is also worth considering the use of deep learning and other machine learning approaches.

Author Contributions: Conceptualization, W.A.F.; methodology, W.A.F.; software, W.A.F.; validation, W.A.F.; formal analysis, W.A.F.; investigation, W.A.F.; resources, W.A.F.; data curation, W.A.F.; writing—original draft preparation, W.A.F.; writing—review and editing, J.M.H.B.; visualization, W.A.F.; supervision, W.A.F.; project administration, W.A.F.; funding acquisition, J.M.H.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by The American University of the Middle East (AUM), Kuwait. The APC was funded by The American University of the Middle East (AUM), Kuwait.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to the need of permission from the funding source.

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

Nomenclature

ADAS	Advanced Driving Assistance Systems
CTRV	Constant Turn Rate and Velocity
egocar	The reference vehicle contains sensors that perceive the environment around it.
EKF	Extended Kalman Filter
DGPS	Differential Global Positioning Systems
GB-DBSCAN	Grid-Based Density-Based Spatial Clustering of Applications with Noise
GPS	Global Positioning System
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
KF	Kalman Filter
KPI	Key Performance Indicator
RANSAC	RANdom SAmple Consensus
RTK-GPS	Real-Time Kinematic Global Positioning System
RTMCL	Real-Time Monte Carlo Localization
RMSE	Root Mean Squared Error
NIS	Normalized Innovation Squared
PF	Particle Filter
UKF	Unscented Kalman Filter
SDC	Self-Driving Car
SD	Standard Deviation
ψ	The object orientation yaw angle (heading)
v	The magnitude of object velocity
(,)	The moving object's centroid position
λ	Spread of the generated sigma points (a design parameter).
X	The UKF sate sigma-point matrix
p_x, p_y	The object's current x and y positions.
P	UKF state covariance matrix
K	UKF gain matrix
T	UKF cross-correlation matrix
R	Measurement noise covariance matrix
S	The predicted measurement covariance matrix
Δt	The difference in time between two consecutive timestamps.
M	The number of particles.
$bel(x_t)$	Posterior distribution.
χ_t	Denoted set of particles at time step t
$p\left(x_t \middle\| u_t, x_{t-1}^{[m]}\right)$	PF state transition distribution

References

1. Yurtsever ELambert Carballo, J.; Takeda, A.K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* **2020**, *8*, 58443–58469. [CrossRef]
2. Farag, W. A lightweight vehicle detection and tracking technique for advanced driving assistance systems. *J. Intell. Fuzzy Syst.* **2020**, *39*, 2693–2710. [CrossRef]

3. Fisser, H.; Khorsandi, E.; Wegmann, M.; Baier, F. Detecting Moving Trucks on Roads Using Sentinel-2 Data. *Remote Sens.* **2021**, *14*, 1595. [CrossRef]
4. Farag, W. Multiple Road-Objects Detection and Tracking for Autonomous Driving. *J. Eng. Res.* **2022**, *10*, 237–262. [CrossRef]
5. Farag, W. Lidar and Radar Fusion for Real-Time Road-Objects Detection and Tracking. *Intell. Decis. Technol.* **2021**, *15*, 291–304. [CrossRef]
6. Farag, W. A Comprehensive Real-Time Road-Lanes Tracking Technique for Autonomous Driving. *Int. J. Comput. Digit. Syst.* **2020**, *9*, 349–362. [CrossRef]
7. Farag, W. Real-Time Detection of Road Lane-Lines for Autonomous Driving. *Recent Adv. Comput. Sci. Commun.* **2020**, *13*, 265–274. [CrossRef]
8. Guastella, D.; Muscato, G. Learning-Based Methods of Perception and Navigation for Ground Vehicles in Unstructured Environments: A Review. *Sensors* **2021**, *21*, 73. [CrossRef]
9. Babak, S.-J.; A Hussain, S.; Karakas, B.; Cetin, S. Control of autonomous ground vehicles: A brief technical review. In Proceedings of the 4th International Conference on Mechanics and Mechatronics Research (ICMMR 2017), Xi'an, China, 20–24 June 2017. [CrossRef]
10. Farag, W. Road-objects tracking for autonomous driving using lidar and radar fusion. *J. Electr. Eng.* **2020**, *71*, 138–149. [CrossRef]
11. Woo, A.; Fidan, B.; Melek, W.W. Localization for Autonomous Driving. In *Handbook of Position Location: Theory, Practice, and Advances*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2019.
12. Zekavat, R.; Buehrer, R.M. Localization for Autonomous Driving. In *Handbook of Position Location: Theory, Practice, and Advances*; IEEE: Piscataway, NJ, USA, 2019; pp. 1051–1087. [CrossRef]
13. Smit, R.; Van Mourik, H.; Verroen, E.; Pieters, M.; Bakker, D.; Snelder, M. Will self-driving cars impact the long-term investment strategy for the Dutch national trunk road system? In *Autonomous Vehicles and Future Mobility*; Elsevier: Amsterdam, The Netherlands, 2018; pp. 57–67. [CrossRef]
14. Ma, H.; Pei, W.; Zhang, Q. Research on Path Planning Algorithm for Driverless Vehicles. *Mathematics* **2022**, *10*, 2555. [CrossRef]
15. Kuutti, S.; Fallah, S.; Katsaros, K.; Dianati, M.; Mccullough, F.; Mouzakitis, A. A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications. *IEEE Internet Things J.* **2018**, *5*, 829–846. [CrossRef]
16. Homolla, T.; Winner, H. Encapsulated trajectory tracking control for autonomous vehicles. *Automot. Engine Technol.* **2022**, *7*, 295–306. [CrossRef]
17. Farag, W. Complex-Track Following in Real-Time Using Model-Based Predictive Control. *Int. J. Intell. Transp. Syst. Res.* **2021**, *19*, 112–127. [CrossRef]
18. Liu, Y.; Pei, X.; Guo, X.; Chen, C.; Zhou, H. An Integration Planning and Control Method of Intelligent Vehicles based on the Iterative Linear Quadratic Regulator. *J. Frankl. Inst.* **2024**, *360*, 265–282. [CrossRef]
19. Khare, V.; Jain, A. Predict the performance of driverless car through the cognitive data analysis and reliability analysis based approach. *E-Prime-Adv. Electr. Eng. Electron. Energy* **2023**, *6*, 100344. [CrossRef]
20. Levinson, J.; Montemerlo, M.; Thrun, S. Map-Based Precision Vehicle Localization in Urban Environments. In Proceedings of the Conference: Robotics: Science and Systems III, Virtual, 27–30 June 2007; Georgia Institute of Technology: Atlanta, GA, USA, 2007.
21. Veronese, L.; Auat-Cheein, F.; Mutz, F.; Oliveira-Santos, T.; Guivant, J.E.; de Aguiar, E.; Badue, C.; De Souza, A.F. Evaluating the Limits of a LiDAR for an Autonomous Driving Localization. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 1449–1458. [CrossRef]
22. Zhou, T.; Yang, M.; Jiang, K.; Wong, H.; Yang, D. MMW Radar-Based Technologies in Autonomous Driving: A Review. *Sensors* **2020**, *20*, 7283. [CrossRef] [PubMed]
23. Takanose, A.; Atsumi, Y.; Takikawa, K.; Meguro, J. Improvement of Reliability Determination Performance of Real-Time Kinematic Solutions Using Height Trajectory. *Sensors* **2021**, *21*, 657. [CrossRef]
24. Rathour, S.S.; Boyali, A.; Zheming, L.; Mita, S.; John, V. A Map-Based Lateral and Longitudinal DGPS/DR Bias Estimation Method for Autonomous Driving. *Int. J. Mach. Learn. Comput.* **2017**, *7*, 67–71. [CrossRef]
25. Carlevaris-Bianco, N.; Ushani, A.K.; Eustice, R.M. University of Michigan North Campus long-term vision and lidar dataset. *Int. J. Robot. Res.* **2015**, *35*, 1023–1035. [CrossRef]
26. Modsching, M.; Kramer, R.; Hagen, K. Field trial on GPS accuracy in a medium-size city: The influence of built-up. In Proceedings of the 3rd Workshop on Positioning, Navigation, and Communication, Hannover, Germany, 16 March 2006; pp. 209–218.
27. Rakhmanov, A.; Wiseman, Y. Compression of GNSS Data with the Aim of Speeding up Communication to Autonomous Vehicles. *Remote Sens.* **2023**, *15*, 2165. [CrossRef]
28. Li, W.; Li, Z.; Jiang, W.; Chen, Q.; Zhu, G.; Wang, J. A New Spatial Filtering Algorithm for Noisy and Missing GNSS Position Time Series Using Weighted Expectation Maximization Principal Component Analysis: A Case Study for Regional GNSS Network in Xinjiang Province. *Remote Sens.* **2022**, *14*, 1295. [CrossRef]
29. Levinson, J.; Thrun, S. Robust vehicle localization in urban environments using probabilistic maps. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, Alaska, 3–8 May 2010; pp. 4372–4378.
30. Wikipedia. Point Cloud. 10 February 2023. Available online: https://en.wikipedia.org/wiki/Point_cloud (accessed on 14 March 2023).
31. Metabase. Visualizing Data with Maps. 2023. Available online: https://www.metabase.com/learn/visualization/maps (accessed on 14 March 2023).

32. Wikipedia. Polygon Mesh. 11 January 2023. Available online: https://en.wikipedia.org/wiki/Polygon_mesh (accessed on 14 March 2023).
33. Mapping 2023. Why Better Mapping Technology Is Critical to Autonomous Vehicles. Available online: https://semiengineering.com/why-better-mapping-technology-is-critical-to-autonomous-vehicles/ (accessed on 4 March 2023).
34. Kummerle, J.; Sons, M.; Poggenhans, F.; Kuehner, T.; Lauer, M.; Stiller, C. Accurate and efficient self-localization on roads using basic geometric primitives. In Proceedings of the 2019 IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019.
35. Weng, L.; Yang, M.; Guo, L.; Wang, B.; Wang, C. Pole-based realtime localization for autonomous driving in congested urban scenarios. In Proceedings of the 2018 IEEE International Conference on Real-Time Computing and Robotics, Kandima, Maldives, 1–5 August 2018; pp. 96–101.
36. Fischler, M.; Bolles, R. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
37. Sefati, M.; Daum, M.; Sondermann, B.; Kreisk, K.D.; Kampker, A. Improving vehicle localization using semantic and pole-like landmarks. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium, Los Angeles, CA, USA, 11–14 June 2017; pp. 13–19.
38. Schaefer, A.; Büscher, D.; Vertens, J.; Luft, L.; Burgard, W. Long-Term Urban Vehicle Localization Using Pole Landmarks Extracted from 3-D Lidar Scans. In Proceedings of the European Conference on Mobile Robots (ECMR), Prague, Czech Republic, 4–6 September 2019.
39. Chiang, K.; Chiu, Y.; Srinara, S.; Tsai, M. Performance of LiDAR-SLAM-based PNT with initial poses based on NDT scan matching algorithm. *Satell. Navig.* **2023**, *4*, 3. [CrossRef]
40. Taheri, H.; Xia, Z.C. SLAM; definition and evolution. *Eng. Appl. Artif. Intell.* **2021**, *97*, 104032. [CrossRef]
41. Suhr, J.K.; Jang, J.; Min, D.; Jung, H.G. Sensor Fusion-Based Low-Cost Vehicle Localization System for Complex Urban Environments. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1078–1086. [CrossRef]
42. Lu, F.; Milios, E. Robot pose estimation in unknown environments by matching 2D range scans. *J. Intell. Robot. Syst.* **1997**, *18*, 249–275. [CrossRef]
43. Thrun, S. Particle Filters in Robotics. In Proceedings of the 18th Annual Conference on Uncertainty in AI (UAI), Edmonton, AB, Canada, 1–4 August 2002.
44. Zarchan, P.; Musoff, H. *Fundamentals of Kalman Filtering: A Practical Approach*, 4th ed.; American Institute of Aeronautics and Astronautics, Incorporated: Reston, VA, USA, 2013; ISBN 978-1-62410-276-9.
45. Wan, E.A.; Van Der Merwe, R. The unscented Kalman filter for nonlinear estimation. In Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium, Lake Louise, AB, Canada, 4 October 2000.
46. Einicke, G.A.; White, L.B. Robust Extended Kalman Filtering. *IEEE Trans. Signal Process.* **1999**, *47*, 2596–2599. [CrossRef]
47. Sander, J.; Xu, X.; Ester, M.; Kriegel, H.-P. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996; pp. 226–231.
48. Dietmayer, K.; Kellner, D.; Klappstein, J. Grid-based dbscan for clustering extended objects in radar data. In Proceedings of the 2012 IEEE Intelligent Vehicles Symposium, Alcala de Henares, Spain, 3–7 June 2012.
49. Wikipedia. K-D Tree. 28 February 2023. Available online: https://en.wikipedia.org/wiki/K-d_tree (accessed on 21 March 2023).
50. GCC, The GNU Compiler Collection. 2023. Available online: https://gcc.gnu.org/ (accessed on 22 March 2023).
51. Wikipedia. Real-Time Computing. 16 November 2023. Available online: https://en.wikipedia.org/wiki/Real-time_computing (accessed on 1 December 2023).
52. Ubuntu Linux. 2023. Available online: https://www.ubuntu.com/ (accessed on 22 March 2023).
53. Eigen. 2023. Available online: http://eigen.tuxfamily.org/index.php?title=Main_Page (accessed on 22 March 2023).
54. Zhao, S.; Huang, B. On Initialization of the Kalman Filter. In Proceedings of the 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP), Taipei, Taiwan, 28–31 May 2017.
55. Piché, R. Online tests of Kalman filter consistency. *Int. J. Adapt. Control Signal Process.* **2016**, *30*, 115–124. [CrossRef]
56. CARLA. Open-Source Simulator for Autonomous Driving Research. 2023. Available online: https://carla.org/ (accessed on 31 March 2023).
57. Farag, W. Real-Time Autonomous Vehicle Localization Based on Particle and Unscented Kalman Filters. *J. Control Autom. Electr. Syst.* **2021**, *32*, 309–325. [CrossRef]
58. Spangenberg, R.; Goehring, D.; Rojas, R. Pole-based Localization for Autonomous Vehicles in Urban Scenarios. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016.
59. Gao, H.; Cheng, B.; Wang, J.; Li, K.; Zhao, J.; Li, D. Object classification using CNn-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4224–4231. [CrossRef]
60. Gao, H.; Su, H.; Cai, Y.; Wu, R.; Hao, Z.; Xu, Y.; Wu, W.; Wang, J.; Li, Z.; Kan, Z. Trajectory prediction of cyclist based on dynamic Bayesian network and long short-term memory model at unsignalized intersections. *Sci. China Inf. Sci.* **2021**, *64*, 172207. [CrossRef]

61. Hosur, P.; Shettar, R.B.; Potdar, M.B. Environmental awareness around vehicle using ultrasonic sensors. In Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, 21–24 September 2016; pp. 1154–1159. [CrossRef]
62. Wiseman, Y. Ancillary Ultrasonic Rangefinder for Autonomous Vehicles. *Int. J. Secur. Its Appl.* **2018**, *12*, 49–58. [CrossRef]