International Journal of Robotics and Control Systems
Vol. 5, No. 2, 2025, pp. 1348-1373
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

Real-Time Pose Estimation for Autonomous Vehicles Using Probabilistic Landmark Maps and Sensor Fusion

Wael A. Farag [a,1,*], Mohamed Fayed [b,2]

[a] Electrical Power Engineering Department, Cairo University, Giza 12613, Egypt
[b] College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait
[1] wael.farag@cu.edu.eg; [2] mohamed.fayed@aum.edu.kw
* Corresponding Author

ARTICLE INFO

Article history
Received March 29, 2025
Revised May 08, 2025
Accepted May 31, 2025

Keywords
UKF;
Localization;
ADAS;
Monte Carlo;
Autonomous Driving;
Particle Filter;
Sensor Fusion;
Kalman Filter

ABSTRACT

This study introduces a robust and accurate method for estimating autonomous vehicle position, facilitating safe navigation in urban and highway settings. The proposed technique employs a probabilistic particle filter framework, which, unlike approaches constrained by Gaussian assumptions, represents probability densities as samples, enabling more flexible position estimation. A key innovation lies in integrating a finely tuned Unscented Kalman Filter (UKF) to fuse radar and lidar data specifically for robust detection of pole-like static landmarks, whose positions and associated uncertainties are probabilistically modeled within an offline reference map. The particle filter leverages Bayesian filtering, associating UKF-derived landmark observations with this probabilistic map to refine the vehicle's pose. Broad simulation tests validate the method's effectiveness, achieving a mean localization error of approximately 11 cm in both longitudinal and lateral directions. Furthermore, the system demonstrates robustness, maintaining localization accuracy below 30 cm even with landmark position uncertainties up to 2 meters, and confirms real-time capability exceeding 100 Hz. These findings establish the approach as a reliable and precise solution for autonomous vehicle localization across various scenarios.

1. Introduction

Achieving full autonomy in vehicles requires overcoming significant technological hurdles [1], broadly categorized into perception [2]-[13], localization [14]-[16], path planning [17]-[19], and controls [21]-[24]. Among these, precise and reliable localization is paramount for safe operation. This involves accurately determining the vehicle's position and orientation (pose) [13], [25]-[28] within a reference map, often requiring centimeter-level accuracy [14], [15] under diverse and dynamic conditions. This high precision enables the vehicle to understand its environment, including the road network, objects, and lane boundaries [16], which is crucial for essential functions like lane-keeping, navigating complex intersections [25], and safe interaction with the environment [13]. However, maintaining this accuracy consistently, especially in challenging urban settings or adverse weather, remains a significant technological challenge [26]. This paper introduces a robust, real-time localization methodology, termed Real-Time Monte Carlo Localization (RTMCL), designed

specifically to address these challenges for autonomous driving using probabilistic sensor fusion and mapping techniques [28]-[32].

Various localization strategies exist, often relying on a suite of sensors including IMU, GPS, lidar, radar, cameras, and odometry [26]. Satellite-based systems like Real-Time Kinematic GPS (RTK-GPS) [33] or Differential GPS (DGPS) [34] can offer high precision but suffer from signal blockage in urban canyons [35], [36] and inherent latency issues [37], [38], limiting their reliability as a standalone solution [20]. Simultaneous Localization and Mapping (SLAM) techniques build maps dynamically [39], [52], avoiding reliance on pre-built ones [30], but often face challenges with computational cost, long-term drift, and meeting the strict real-time accuracy demands of autonomous driving [40], [53]. Consequently, map-based localization, which matches sensor data to a pre-existing map [27], is widely used. While dense High-Definition (HD) maps constructed from point clouds, grid maps, or polygon meshes [29], [41]-[44] provide rich environmental detail, their creation, storage (requiring massive memory [45]), and maintenance pose significant burdens. As a result, approaches using sparse 'landmark maps,' which encode only a restricted number of highly identifiable features extracted from sensor data [14], [46], offer a more scalable and computationally efficient alternative. This study focuses on enhancing localization performance using such landmark maps, specifically targeting pole-like features which are common and relatively stable in many road environments.

Many existing landmark-based localization methods, particularly those utilizing pole-like features, rely heavily on lidar sensors due to their high spatial resolution [47]-[49]. However, while lidar excels in high-resolution mapping, its performance is known to degrade significantly in adverse weather conditions such as fog, heavy rain, or snow, and its effective range can be limited compared to other sensors [31]. This sensitivity poses a critical risk to localization robustness. In contrast, radar offers longer range and remains largely unaffected by weather, although with lower resolution [32]. To overcome these individual sensor limitations, a key contribution of this work is the use of dynamic sensor fusion, specifically integrating lidar and radar data via a carefully tuned Unscented Kalman Filter (UKF) [related to general fusion concept in 26]. This approach leverages radar's superior range and all-weather capabilities [32] to complement lidar's precision [31], enabling more reliable and consistent detection and state estimation of landmarks across a wider range of operating conditions, thereby enhancing the input to the localization filter. While advancements in lidar alternatives like camera-based localization or ground-penetrating radar exist [39], [40], fusing lidar and radar offers a compelling balance for landmark detection currently.

Furthermore, detected landmark positions derived from any sensor data inherently contain measurement noise and uncertainty. Simply matching observed landmarks to deterministic map locations can lead to inaccuracies, especially when sensor noise is significant or map imperfections exist. Addressing this, another core contribution of this research lies in explicitly modeling and representing the uncertainty associated with landmark locations probabilistically within the reference map. This probabilistic map, generated offline, encodes not just the mean position but also the expected variance of each landmark. This richer map information is then integrated within a Monte Carlo localization framework implemented using a Particle Filter (PF) [27]. Particle filters are well-suited for this, maintaining a set of potential poses (particles) and updating them based on measurements and motion models [27], naturally incorporating these probabilistic landmark observations alongside vehicle motion data (odometry) [26]. This probabilistic approach effectively handles uncertainties and allows for non-Gaussian pose estimations, enhancing overall localization accuracy and robustness compared to methods relying on simpler error models or deterministic maps.

The proposed RTMCL pipeline utilizes supporting techniques [50]-[54] such as clustering algorithms (specifically, GB-DBSCAN [62], based on DBSCAN concepts]) to extract landmark candidates from the fused UKF sensor output and the Iterative Closest Point (ICP) algorithm [55] for efficient data association between observed landmarks and the probabilistic map. While prior research has explored pole-based localization using lidar point clouds [47], [49], incorporating additional geometric features like building facades or lane lines [46], or employing specific pole detection algorithms based on lidar properties [51], this study distinguishes itself through the synergistic combination of robust multi-modal sensor fusion (Radar-Lidar via UKF) for enhanced landmark

detection resilience and the explicit probabilistic modeling of landmark uncertainties within the map, integrated into a real-time capable particle filter framework for superior localization performance.

The remainder of this paper is structured as follows: Section 2 details the proposed Real-Time Monte Carlo Localization (RTMCL) methodology, including the UKF sensor fusion, probabilistic landmark mapping, data association, and particle filter implementation. Section 3 describes the simulation setup and presents the experimental results evaluating the system's accuracy, robustness, and real-time performance. Section 4 discusses these results in the context of related work, and Section 5 provides concluding remarks and directions for future work.

2. The Method

The RTMCL technique depicted in Fig. 1 and detailed here distinguishes itself from prior landmark-based localization methods [e.g., 47, 49] by employing UKF-based radar-lidar fusion for enhanced landmark detection robustness in varied conditions and integrating these observations with a probabilistic landmark map within a particle filter framework to explicitly handle uncertainties. The RTMCL algorithm requires four categories of input information:

1. Global Position Information: Primarily derived from GPS data, this information can be augmented by integrating signals from an IMU unit if available. The fusion of GPS and IMU signals corrects accumulated errors during periods of dead reckoning when GPS data is unavailable [54]. This fused output, providing an initial pose of the vehicle, is subsequently used by the particle filter.
2. Odometry Measurements: The algorithm utilizes the vehicle's speed readings and steering angle (for calculating the yaw rate). These measurements, once filtered to reduce noise, are employed to update the state of the particle filter.
3. Object-Detection Sensory Output: The primary sensory inputs include data from lidar and radar. While camera data is not used in this work, the lidar and radar inputs are fused using a tailored UKF. The output from this fusion process is then clustered using the GB-DBSCAN procedure to identify potential pole-like stationary objects. Additionally, measurements of speed (Doppler signals) from the radar support to filter out dynamic objects.
4. Map of the Reference Landmarks: Such a map includes pole-like landmarks with complete coordinates, extracted offline from 3D point-cloud lidar data. During the localization process, these reference landmarks are associated with detected pole-like landmarks (identified by the GB-DBSCAN algorithm) through a “data association” procedure. The Iterative Closest Point (ICP) method is utilized for such a specific purpose, which is critical for accomplishing precise localization [55].

These classes of input measurements are processed using a tailored Particle Filter, which will be detailed later [56], to determine the most accurate pose of the vehicle. The particle filter is initialized via the fused GPS and IMU output and is further refined using the detected pole-like landmarks in addition to the reference map to produce a highly precise vehicle pose.

2.1. Outline of the UKF

The Kalman Filter (KF) is a widely used algorithm for state estimation in linear systems [57]. It operates through a cyclical process of prediction and update, aiming to minimize the estimated covariance of the error. For a given measurement $z \in R^m$ of a discrete-time controlled process defined by a set of linear difference equations of stochastic nature, the KF predicts the state $x \in R^n$.

However, the KF's applicability is limited by its requirement for linear processes. Real-world sensor measurements, such as those from radar systems, often exhibit non-linear characteristics. The UKF addresses this limitation by providing a robust alternative for non-linear systems [58]. Unlike the Extended Kalman Filter (EKF) which relies on linearization techniques, the UKF adopts a

deterministic sampling approach that avoids derivatives [59]. Similar to the KF, the UKF employs a two-stage prediction and update process. However, the UKF incorporates additional steps like sigma-point generation and prediction, as illustrated in Fig. 2. These additional steps enable the UKF to effectively handle non-linear system behavior.

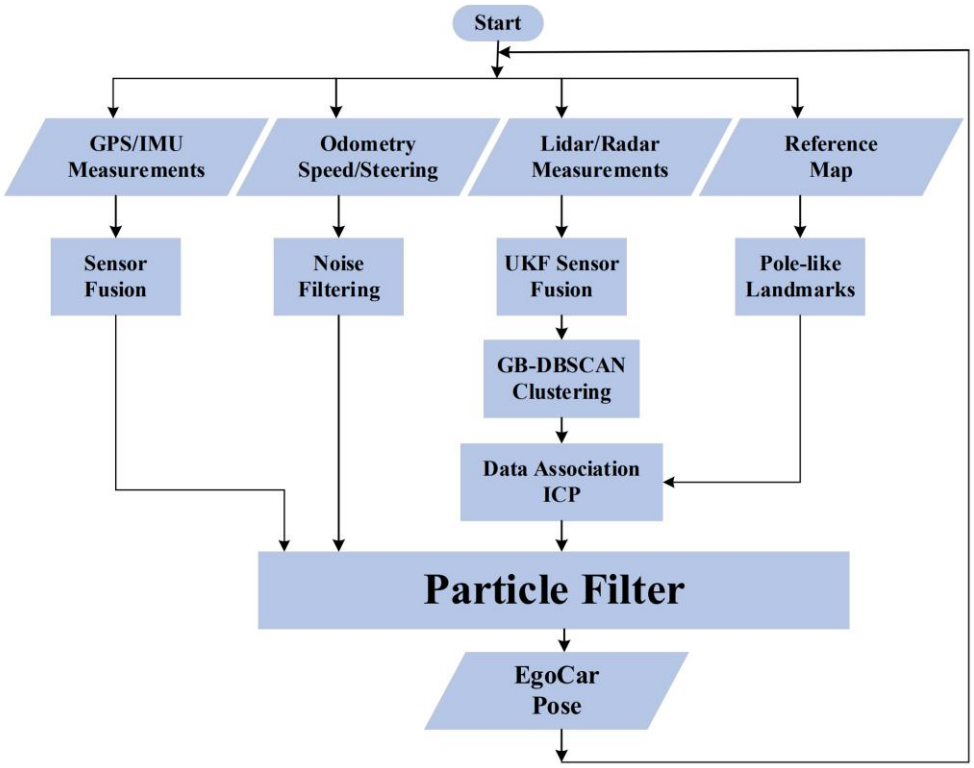

Fig. 1. The workflow of the RTMCL algorithm

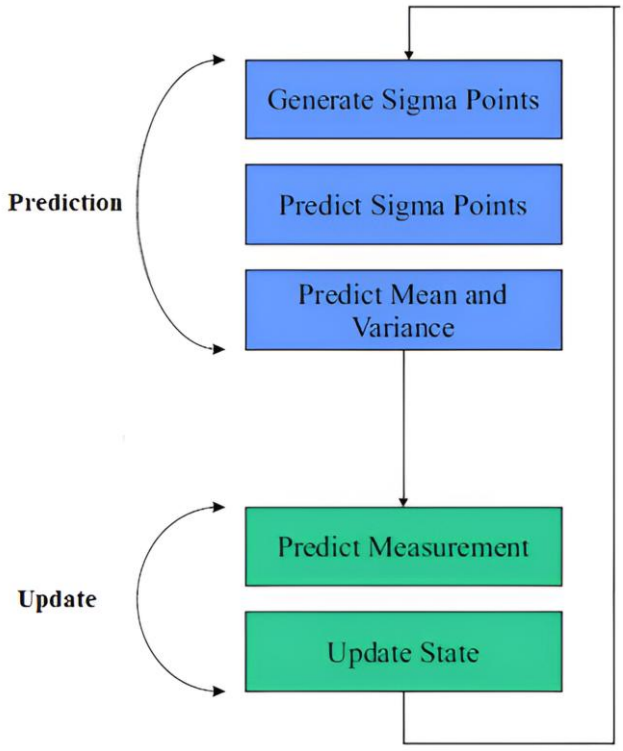

Fig. 2. Workflow of the UKF

In the UKF process, the state Gaussian distribution is represented by a small number of deterministically chosen sample points known as sigma points that are used to capture the mean and covariance of the state distribution. The number of these points are $n_x = 2n + 1$. They are selected 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 covariance matrix of the KF process estimate, X_k is the sigma-point matrix comprising n_x sigma-point vectors, and λ is a design parameter that expresses the sigma points spread, regularly computed as $\lambda = 3 - n_x$.

These sigma points are propagated through the non-linear system dynamics model by becoming input to the UKF's nonlinear process model, designated in Eq. (2), to construct the matrix of predicted sigma points $\hat{X}$ with dimensions $n \times n_x$. This stage is known as the sigma-point prediction phase:

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

where v_k represents the process white noise, represented by a Gaussian distribution ($\mathcal{N}$) with zero mean and covariance matrix Q_k.

The predicted state has the mean and covariance matrices being calculated from the predicted sigma points exploiting Eq. (3):

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

where $w_i's$ are the sigma-point weights used to reverse the sigma-point spreading. These weights are calculated as in Eq. (4):

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

Next, each sigma point is inserted into the nonlinear unscented Kalman filter measurement model, defined by Eq. (5), to construct the matrix of predicted measurement sigma points that has dimensions $n \times n_x$:

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

Then, the predicted measurement has the mean and covariance matrices being calculated by the predicted sigma points and the measurement noise covariance matrix R, as shown in Eq. (6):

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

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

The final stage of the UKF involves updating the state, where the gain matrix (K) is computed using the estimated cross-correlation matrix (T) between the sigma points in the state space and the measurement space, as described in Eq. (7). The state vector (x) and the state covariance matrix (P) are using the gain matrix (K):

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

2.2. The Road Object's Model

The UKF prediction step utilizes a standard constant turn rate and velocity (CTRV) motion model [60], whose non-linear difference equations describing the object's state evolution (position, velocity, yaw angle, yaw rate) are given by Eq. (8)-(12).

In this model, any road object is characterized by 5 variables assembled into one vector called the state vector x. These variables include the object's location on the x and y axes p_x and p_y, the magnitude of the object's velocity v, the yaw angle (ψ), and the yaw rate $\dot{\psi}$, as depicted in Fig. 3. The state vector is defined as follows in Eq. (8):

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

Founded on the state vector x, the nonlinear difference equation $x_{k+1} = f(x_k, \nu_k)$ that embodies the object's motion model is created. The dynamic equations are provided in Eqs. (9), (10), and (11):

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

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

$$\Delta t = t_{k+1} - t_k \tag{11}$$

In these equations, $v_{a,k} \sim \mathcal{N}(0, \sigma_a^2)$ represents the longitudinal acceleration noise at sample k with a standard deviation of σ_a^2, and $v_{\ddot{\psi},k} \sim \mathcal{N}(0, \sigma_{\ddot{\psi}}^2)$ represents the noise of the yaw acceleration at sample k with a standard deviation of $\sigma_{\ddot{\psi}}^2$.

If $\dot{\psi}$ is zero, to avoid division by zero in Eq. (9), a linear model approximation is used to predict p_x and p_y:

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

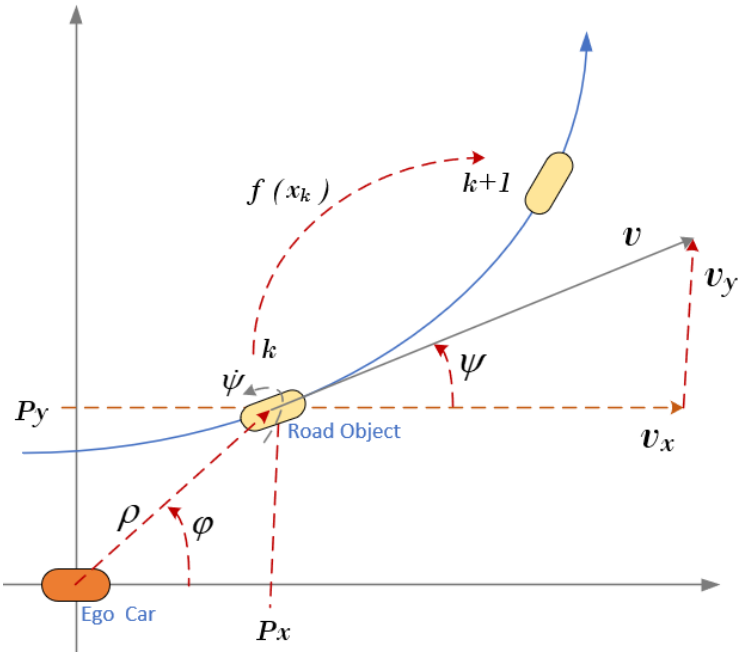

Fig. 3. An arbitrary road object's motion model

2.3. Lidar and Radar Fusion Based on the UKF

The lidar sensor measures the centroid of an object's position (whether stationary or moving) in Cartesian coordinates (p_x and p_y), as described in Eq. (13). Conversely, the radar sensor measures the same object's centroid position in polar coordinates (ρ and φ) and also captures the object's velocity ($\dot{\rho}$), as outlined in Eq. (14). To standardize these measurements, a mapping function is employed to convert the lidar's Cartesian coordinates to the radar's polar form, as shown in Eq. (15).

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

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

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

As illustrated in Fig. 4, the prediction step is performed simultaneously for both radar and lidar sensors. Nevertheless, the update step is sensor-specific, as each sensor has its unique measurement model on its own. Additionally, the belief update is carried out on the appearance of a freshly received sensor measurement, acknowledging that the sensors are not synchronized.

Following initializing both the unscented Kalman filter and the radar and lidar measurement models, the time step (Δt) is calculated, as presented in Fig. 4. Simultaneously, the sigma points (X_k) are generated using Eq. (1). The predicted sigma points ($\hat{X}_{k+1}$) for the next time step are then computed by Eq. (2) and the object's model described using Eq. (9). The predicted state mean vector ($\hat{x}_{k+1}$) and its covariance matrix ($\hat{P}_{k+1}$) are obtained through the application of Eq. (3).

In the update step of the fusion process, two branches are there: the lidar branch and the radar branch. The branch chosen depends on the most recently received measurement data. If radar data is obtained, the predicted measurement sigma points ($\hat{Z}_{k+1}$) are calculated based on the model in Eq. (14) and from $\hat{X}_{k+1}$ using Eq. (5). The predicted measurement mean ($\hat{z}_{k+1}$), the covariance matrix (S_{k+1}), and the noise covariance matrix (R_{radar}) are then computed using Eq. (6). The R_{radar} covariance matrix is detailed in Eq. (16):

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

where $\sigma_{\dot{\rho}}$, σ_φ, and σ_ρ are the noise standard deviations (SDs) for the object's radial velocity, heading, and radial distance, respectively.

The cross-correlation matrix (T_{k+1}) is calculated using the state vectors $\hat{x}_{k+1}$ and $\hat{z}_{k+1}$, along with their sigma points $\hat{X}_{k+1}$ and $\hat{Z}_{k+1}$, using Eq. (7). This matrix is employed to calculate the unscented Kalman filter gain (K_{k+1}), which is then employed to update the state vector (x_{k+1}) and the covariance matrix (P_{k+1}) as presented in Eq. (7). The updated state vector and covariance matrix are employed to generate updated sigma points (X_{k+1}) for the next iteration.

As an alternative, if lidar data is received, the predicted measurement sigma points ($\hat{Z}_{k+1}$) are calculated based on the linear lidar measurement model (H_{lidar}) in Eq. (17) and directly from $\hat{X}_{k+1}$. The predicted measurement mean ($\hat{z}_{k+1}$), the covariance matrix (S_{k+1}), and the noise covariance matrix (R_{lidar}) are then calculated using Eq. (6), with more details of R_{lidar} provided in Eq. (17):

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

where σ_{p_x} and σ_{p_y} are the noise SDs for the object's *x* and *y* positions, respectively. Similar to the radar branch, the cross-correlation matrix (T_{k+1}), the UKF gain (K_{k+1}), and the updated covariance matrix (P_{k+1}) are calculated using Eq. (7).

This fusion process assumes accurate time synchronization between the lidar and radar sensors, typically achieved through hardware triggers or precise software timestamping; significant timing errors could degrade fusion performance. The branching logic depicted in Fig. 3 inherently handles sensor asynchronicity, processing whichever sensor measurement (lidar or radar) arrives next. While fusion enhances robustness against conditions affecting one sensor (e.g., fog for lidar), extreme weather simultaneously degrading both radar (e.g., signal attenuation in torrential rain) and lidar (e.g., whiteout snow) could still negatively impact landmark detection quality and subsequent localization accuracy.

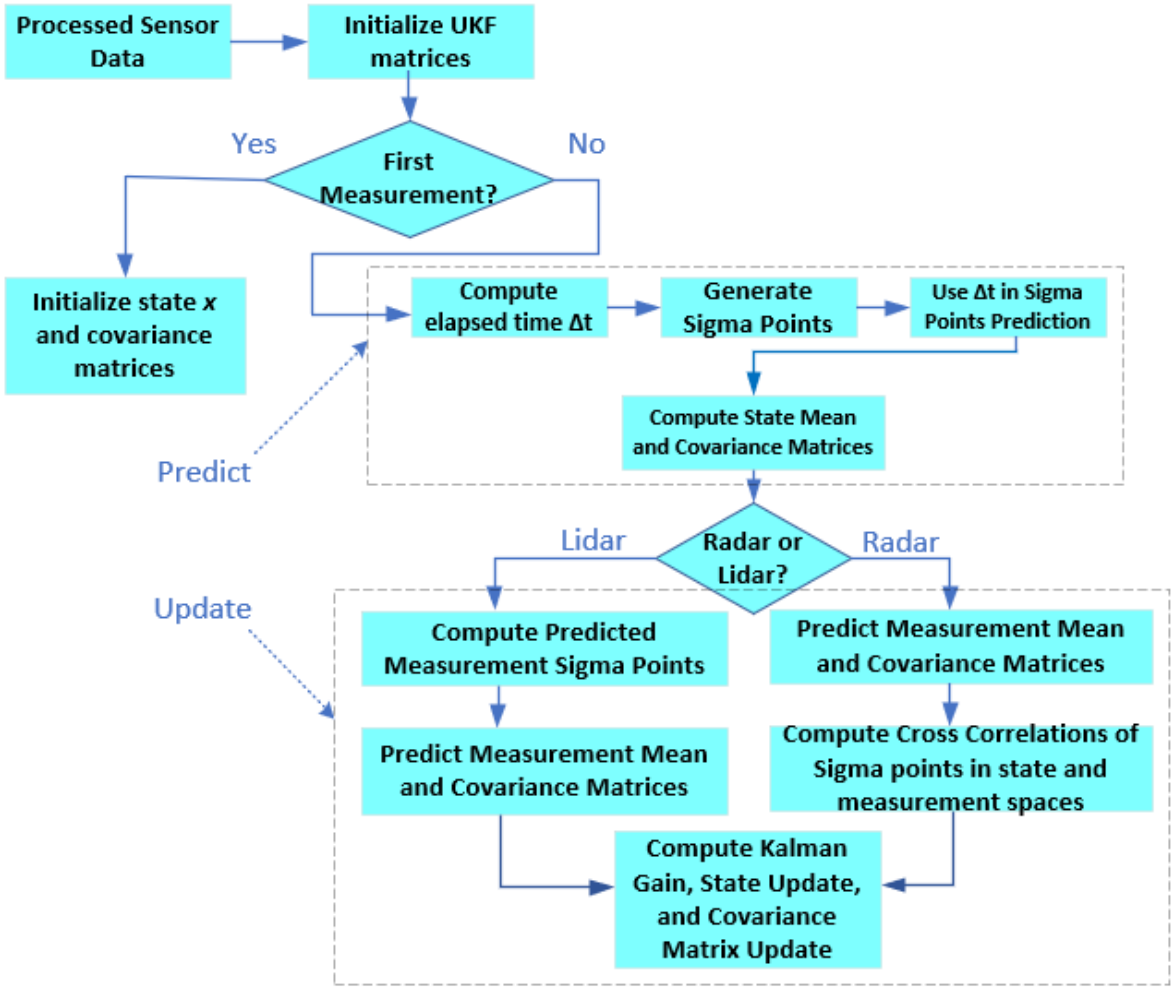

Fig. 4. The fusion workflow of the radar and lidar while employing the UKF

2.4. Data Clustering and Association of the Point-Cloud

The point cloud generated by the unscented Kalman filter fusion procedure, depicted in Fig. 4, provides detailed information regarding the objects surrounding the autonomous vehicle. To obtain the distinct object's information, including its pose and geometrical shape, clustering is applied to the unscented Kalman filter output data. This process reduces computational overhead and memory requirements by representing each object in a simplified source-point model.

In this study, clustering is executed using the GB-DBSCAN process, which is a variation of the primary DBSCAN process [61]. DBSCAN is an unsupervised learning process that groups data points with high density into clusters. Two parameters, "ε" and "*minPts*," are employed to tune DBSCAN and describe the density criteria. "ε" represents the allowable radial distance from the point under valuation ("*p*"), while "minPts" specifies the minimum number of points, including "*p*" itself, that must be within distance "ε" to form a cluster. The proper choice of "ε" and "*minPts*" determines the density of points to be clustered together.

For road objects with varying topologies, the standard DBSCAN algorithm is insufficient. Dietmayer et al. [62] improved this by not using fixed parameters like "ε" and "*minPts*" in GB-DBSCAN. As an alternative, they formed a polar grid that considers the sensor's angular and radial resolution. The search area becomes dynamic, adapting to the object's shape, making this algorithm particularly suitable for pole-like objects that generate a high density of revealing points compared to their neighborhood and prospective surroundings.

The GB-DBSCAN procedure initially performs coarse clustering of the unscented-Kalman-filter fusion data. These clusters are then refined using the RANSAC algorithm, which proposes geometrical shapes for the clusters [48]. For pole-like landmarks, the circular shape is the most suitable and is formfitting to all points in each cluster. The RANSAC technique determines the parameters of each formfitted circle, specifically the radius ($\hat{r}$) and the centroid ($\hat{x}_c, \hat{y}_c$), by optimizing the following formula:

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

After identifying the pole-like landmarks in the supplied fusion data from the unscented-Kalman-filter, the data association step matches these landmarks with their counterparts in the reference map. This step is crucial for the Particle Filter (PF) to maintain precision. The Iterative Closest Point (ICP) algorithm [55] is used for this purpose. Unlike the standard ICP, which searches entire point clouds, this approach only considers centroids during the matching process, significantly reducing processing and memory demands.

The ICP algorithm iteratively performs two phases until convergence. The first phase matches each point in the source set X (UKF data) with the closest point in the target set Y (point-cloud map). The second phase finds the optimal transform (X→Y) based on the matched points. Efficient matching is achieved by storing the X and Y sets in a KD tree data structure [63]. For matched points x_i from X and y_i from Y, the two-dimensional ICP algorithm uncovers the rotation angle "φ" and the translation parameter "t" that minimize the quadratic distance between the target and source points:

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

where $R(\varphi)$ is the rotation matrix using angle "φ" as a variable.

To ensure correct data association, the coordinate systems between the ego car (x_c and y_c) and the reference map (x_m and y_m) must be unified. This is achieved using a homogeneous transformation matrix, as provided by Eq. (20). The rotation and translation are implemented using map particle or ego car coordinates (x_p and y_p) and the rotation angle θ:

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

While algorithm choices like GB-DBSCAN [62] and ICP [63] aim for efficiency, their computational cost can scale with point cloud density (for clustering/RANSAC) and the number of landmarks (for ICP association). In particularly dense or complex environments, these steps could require further optimization or hardware acceleration to guarantee strict real-time performance.

The effectiveness of GB-DBSCAN and RANSAC depends on appropriate parameter tuning (e.g., density criteria, inlier thresholds) and sufficient data points defining the target shape; they may misclassify clutter or fail on poorly defined landmarks. ICP's convergence relies on a reasonable initial pose estimate from the PF prediction and can be sensitive to symmetric or ambiguous landmark configurations.

2.5. Details of the Particle Filter

For a stochastic process with noisy observations $p(z_t|x_t)$, the posterior distribution $bel(x_t)$ can be represented by a finite set of particles. This approach serves as an approximate implementation of the Bayesian filter in a recursive mode, with a normalization factor ζ as given by Eq. (21):

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

The accuracy of the belief distribution $bel(x_t)$ improves with a higher number of particles M, where t denotes the time step at which the particle set's state is considered, as shown in Eq. (22):

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

The optimal solution which represents the actual state is one of the particles in the set χ_t at time t. Each $x_t^{[i]}$ represents a hypothesis for the optimal solution. The Particle Filter (PF) employed in this work is detailed in Table 1, and its workflow is illustrated in Fig. 5. When a new measurement (odometry signal u_t) or a pole-like object's measurement update (z_t) from the unscented Kalman filter is received, and a fresh search for the optimum vehicle pose is launched.

Table 1. The particle filter streamlined workflow

Function Particle Filter (χ_{t-1}, u_t, z_t):
Inputs: Particles' set χ_{t-1} at a time $(t-1)$, control input u_t, and measurements set z_t. **Outputs:** The updated particles' set χ_t at time t.
Begin
Initialize Particles: $\bar{\chi}_t = \chi_t = \emptyset$.
$For\ m = 1\ to\ M\ do$
$generate\ x_i^{[m]} \sim p\left(x_t|u_t, x_{t-1}^{[m]}\right)$
calculate $w_t^{[m]} = p\left(z_t|x_i^{[m]}\right)$
update $\bar{\chi}_t = \bar{\chi}_t + \langle x_i^{[m]}, w_t^{[m]}\rangle$
$For\ m = 1\ to\ M\ do$
$draw\ i\ with\ probability\ \alpha w_t^{[i]}$
$add\ x_t^{[i]}\ to\ \chi_t$
Return χ_t
End.

The Particle Filter implementation involves a predict-update cycle. In the prediction step (step 2 in the function), two values are computed for each particle: the state hypothesis ($x_i^{[m]}$) and the state transition distribution $p\left(x_t|u_t, x_{t-1}^{[m]}\right)$, based on the object's motion model. Each particle is assigned a weight reflecting its importance among other particles, calculated using a multivariate Gaussian probability density function as shown in Eq. (23):

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

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

Subsequently, the set of particles ($\bar{\chi}_t$) is resampled in proportion to their weights ($\alpha w_t^{[i]}$), where α is a coefficient for normalization. This generates a new set of Mparticles, resulting in the updated posterior approximation χ_t. Typically, χ_t will contain multiple copies of the strongest particles, while weaker particles are discarded. The algorithm iterates until χ_t contains M copies of a single particle, representing the solution or the needed vehicle pose.

The gauge for convergence for the particle filter is the weighted mean error ($Error_{weighted}$), computed over all particles as shown in Eq. (24). The $Error_{weighted}$ is calculated by determining

the RMSE between the ground truth g and the state of each particle p_i, weighted by its importance, then summing the product for all particles and dividing by the sum of all weights:

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

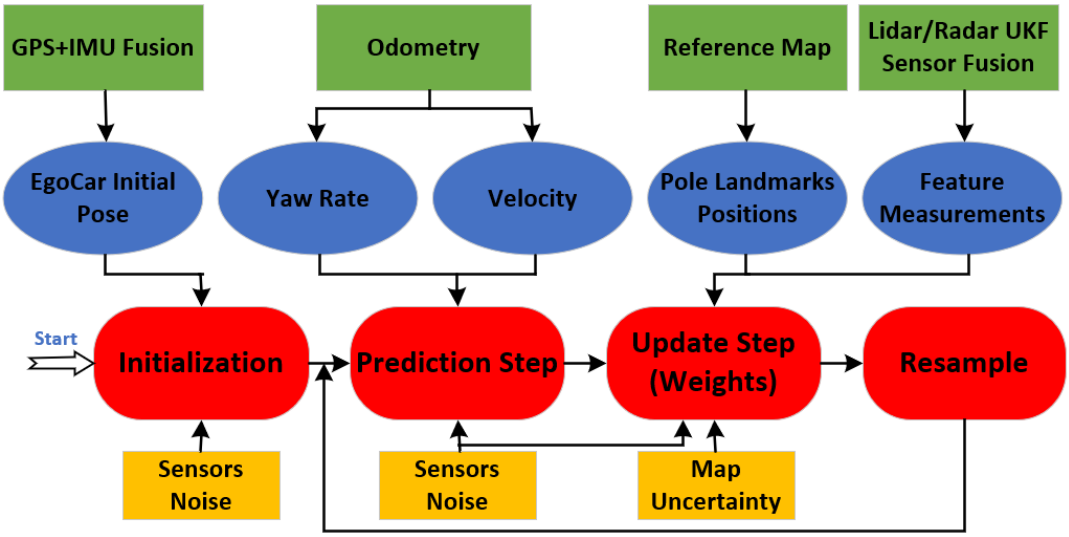

Fig. 5. A flowchart presents an overview of the particle filter methodology

The number of particles, M, presents a critical trade-off between localization accuracy and computational load [29, 56]. While more particles can represent complex probability distributions more accurately, increasing computational demand, simulation experiments (detailed in Sec 3, Table 7) demonstrated that M=50 provided a suitable balance for the tested scenarios. This count achieved the target sub-15cm mean error while maintaining an update cycle time (approx. 0.74 ms for the PF stage, derived from Table 9) well within the requirements for real-time operation (>100Hz).

The performance of this landmark-based PF is fundamentally dependent on the availability, density, and geometric distinguishability of pole-like landmarks in the environment. In areas with very sparse landmarks (e.g., open highways) or highly repetitive/ambiguous configurations, localization accuracy may degrade, potentially requiring integration of other features or sensors.

RTMCL initialization leverages fused GPS/IMU data for an initial pose estimate. In scenarios with prolonged GPS signal denial (e.g., tunnels, deep urban canyons), this initial estimate can drift significantly. While the filter continues relative updates using odometry and landmarks, accumulated error might necessitate re-initialization upon GPS reacquisition. No specific contingency beyond standard filter recovery mechanisms is implemented for complete sensor failure.

2.6. Execution of the RTMCL

The RTMCL system prioritizes real-time performance, reflected in its development and implementation choices. The RTMCL algorithm is implemented in C++ [64], a programming language renowned for its prominent performance, particularly for real-time applications [65]. This choice ensures the system's ability to process data and generate results swiftly. The RTMCL code runs on a Linux-based operating system (Ubuntu) [66], providing a stable and well-supported platform. For numerical computations, the Eigen library is leveraged [67]. This library offers efficient handling of all vector and matrix operations, essential for executing the object models and the prediction-update stages within the RTMCL framework.

Sensor data processing is handled by the NVIDIA DRIVE AGX platform, a powerful platform designed for processing sensor data in autonomous vehicles. The lidar sensor employed is the

Velodyne Lidar VLP-16 [68]. This sensor boasts 16 channels, a range of 100 meters, a 360° horizontal field of view, a 30° vertical field of view, and an impressive capacity of 300,000 points per second. Additionally, the Continental ARS430 radar [68] operates at 77 GHz, offering a range of 250 meters, broad azimuth and elevation coverage, and an accuracy of 0.1 meters. This combination of hardware provides RTMCL with a rich and detailed picture of the surrounding environment.

The accurate execution of motion models for various objects, as described in Equations (9) to (11), hinges on meticulously determined noise parameters [68]. A crucial step involves fine-tuning these parameters, which are then represented in Table 2. To guarantee consistency in the UKF design, the Normalized Innovation Squared (NIS) metric is continuously monitored and employed for further refinement of the noise parameters [69]. This ensures the UKF remains unbiased and delivers consistent results. The estimation error is averaged, aiming for a mean value close to zero, to verify the UKF's lack of bias. The UKF's actual Mean Squared Error (MSE) is compared with the state covariance computed by the UKF itself. Finally, the NIS value at each time step is calculated using Equation (25), and a moving window of N measurement samples is employed to compute the average NIS value ($NIS_{Average}$). These steps ensure the UKF is precisely calibrated and delivers reliable state estimates.

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

The noise parameters governing the UKF motion/measurement models (Table 2, Table 3) and the PF artificial noise/map uncertainty (Table 4) were determined empirically through iterative tuning during offline simulation. Initial values were informed by typical sensor specifications (e.g., radar/lidar accuracy) and motion assumptions. Parameters were then refined by monitoring UKF consistency via the Normalized Innovation Squared (NIS) metric [69] (as described by Eq. (25), aiming for values consistent with the expected chi-squared distribution) and minimizing the overall localization error (RMSE/MAE, Eq. (27)/(29) on validation datasets. It is important to note that these parameters may require re-calibration for different sensor suites, vehicle platforms, or significantly different operating environments to maintain optimal performance.

Table 2. Setting of the object model, particle filter, and unscented kalman filter (noise parameters)

Parameter	Value (UKF/PF)	Parameter	Value (UKF)
σ_a m/s2	1.000	σ_{p_y} m (lidar)	0.150
$\sigma_{\dot{\psi}}$ rad/s2	0.600	σ_ρ m (radar)	0.300
$\sigma_{\dot{\psi}}$ rad/s	0.060	σ_φ rad (radar)	0.030
σ_{p_x}m (lidar)	0.150	$\sigma_{\dot{\rho}}$ m/s (radar)	0.300

The unscented Kalman filter performance depends heavily on its proper initialization [68]. The estimated state vector (x) and its estimated state covariance matrix (P) are critical variables. The initial values for p_x and p_y (the first two terms of x in Eq. (8)) are derived from early sensor measurements. Regarding the remaining 3 terms, a trial and error process combined with intuition is used for initialization, as shown in Table 3. The P matrix is constructed as a diagonal matrix, including the covariance values of each term in x, as given in Eq. (26).

The Root Mean Square Error (RMSE) as defined in Eq. (27) evaluates the UKF's performance, measuring the closeness of the estimated ranges to the ground truth (true ranges). This metric is calculated using an N-sample moving window.

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

Table 3. Initialization of the state-variables of the UKF

Parameter	Value (UKF)	Parameter	Value (UKF)
p_x m	x-reading (1st raw)	p_y m	y-reading (1st raw)
v m/s	0.00	ψ rad	0.00
$\dot{\psi}$ rad/s	0.00	$\sigma_{\hat{p}_x}$ m	1.00
$\sigma_{\hat{p}_y}$ m	1.00	$\sigma_{\hat{v}}$ m/sec	$\sqrt{1000.0}$
$\sigma_{\hat{\psi}}$ rad	$\sqrt{1000.0}$	$\sigma_{\hat{\psi}}$ m/sec2	$\sqrt{1000.0}$

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

where x_k^{true} is the ground-truth state vector generated by a motion driving simulator [70] or supplied as training data during the UKF design phase, and x_k^{est} is the UKF's estimated state vector.

Proper initialization is equally crucial for the Particle Filter (PF). The initialization process includes the following steps:

1. The PF particles' count M is typically between 100 and 1000 [29], as per the literature [56]. A balance between accuracy and computation speed is necessary, and after many trials, $M = 50$ was chosen for its real-time performance alongside the needed accuracy.
2. The initial pose of the ego car $(p_{x_{GPS}}, p_{y_{GPS}}, \theta_{GPS})$ from the GPS/IMU fusion is used to initialize all M particles' state vectors:

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

where $p_x^{[m]}$, $p_y^{[m]}$, and $\theta_{Particle}^{[m]}$ are the initialized poses of particle m. The standard deviations for GPS/IMU fusion noise are $\sigma_{x_{GPS}}$, $\sigma_{y_{GPS}}$, and $\sigma_{\theta_{GPS}}$. Artificial noise $\sigma_{x_{artificial}}$, $\sigma_{\theta_{artificial}}$, and $\sigma_{\theta_{artificial}}$ adds randomness to improve convergence. Table 4 lists the initialization values.

3. Particles' weights, indicating their importance, are initialized using a uniform distribution: $w^{[m]} = \frac{1}{M}$.
4. Landmarks in the reference map, particularly pole-shaped ones, are modeled by Gaussian distributions for x and y positions, represented as $\mathcal{N}(p_{x_{Pole}}, \sigma_{x_{Pole}}^2)$ and $\mathcal{N}(p_{y_{Pole}}, \sigma_{y_{Pole}}^2)$. These distributions model position uncertainties. The standard deviation values $\sigma_{x_{pole}}$ and $\sigma_{y_{pole}}$ are itemized in Table 4.

The MAE (mean absolute error) for each estimated variable of the pose, relative to the ground truth, is calculated using an N-measurements moving window to evaluate PF performance, as presented in Eq. (29):

Table 4. Initialized values for the particle-filter parameters

Parameter	Value (PF)	Parameter	Value (PF)
$\sigma_{x_{GPS}}$	0.300 m	$\sigma_{x_{artificial}}$	10 m
$\sigma_{y_{GPS}}$	0.300 m	$\sigma_{y_{artificial}}$	10 m
$\sigma_{\theta_{GPS}}$	0.010 rad	$\sigma_{\theta_{artificial}}$	0.050 rad
$\sigma_{x_{pole}}$	0.300 m	$\sigma_{y_{pole}}$	0.300 m

where $x_i^{best}, y_i^{best}, and\ \theta_i^{best}$ are the best-estimated values of the poses of the particles of the PF, and $x_i^{gt}, y_i^{gt},\ and\ \theta_i^{gt}$ are the ground truth values produced by motion-driving simulation software or offered as training data during the PF design phase.

A comparative analysis of these tuned parameters against typical values or those used in related works could be presented in the Discussion section to highlight optimization gains.

$$X_{error} = \frac{1}{N}\sum_{i=1}^{N}\left|x_i^{best} - x_i^{gt}\right|$$
$$Y_{error} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i^{best} - y_i^{gt}\right| \quad (29)$$
$$Yaw_{error} = \frac{1}{N}\sum_{i=1}^{N}\left|\theta_i^{best} - \theta_i^{gt}\right|$$

3. Results

3.1. Performance Evaluation and Analysis

The tuning process, iterative and guided by KPIs (Eqs. (24), (26), (27), (29)), assessed RTMCL's performance across various hyperparameter sets. Fig. 6 illustrates localization outcomes on the 754-meter test track featuring curves and 42 pole-like landmarks simulating urban driving. Table 5 presents the UKF performance (RMSE) for detecting different object types, demonstrating its capability. Table 6 highlights the significant benefit of fusing Lidar and Radar, showing improved RMSE and NIS consistency compared to using either sensor alone.

The Particle Filter's convergence is analyzed in Table 7, showing diminishing error returns with increasing particle counts beyond 50, alongside increasing execution time. This highlights a critical computational trade-off: while 200 particles slightly reduce RMSE (e.g., x-error by ~4% compared to 50 particles), they nearly triple the PF execution time (from 0.74ms to 2.4ms). Therefore, M=50 was selected as optimal for this study, balancing sub-15cm accuracy with the demands of real-time operation (>100Hz). Table 8 demonstrates the system's robustness to landmark position uncertainty, maintaining <30cm localization error even with 1-meter standard deviations in the map. The overall real-time capability is confirmed in Table 9, detailing stage-wise execution times summing to 8.21ms (122Hz) on the test hardware.

Fig. 6, Fig. 7, and Fig. 8 are showing (a) Pose Trajectory, (b) Yaw Angle vs. Ground Truth, and (c) Speed/Yaw Rate Time Series, while Fig. 9 shows the typical error decay during the PF initialization phase, stabilizing quickly. Fig. 10 illustrates the particle weight distribution during a lap, indicating robust tracking as the best particle consistently maintains high importance. Fig. 11 displays the number of detected poles used by the PF, showing variability but typically staying within the effective range (4-12 noted in discussion) for stable operation.

An inverse relationship is observed between particle weights and the detected pole count. Equation (30) offers a more concise representation of Equation (22). The weights in Eq. (30) are calculated as the product of the likelihood function associated with observing each pole-like landmark. This likelihood function is modeled by a multivariate Gaussian probability density function. The optimal number of observed poles for smooth RTMCL operation is between 4 and 12, as shown in Fig. 11.

The RTMCL's real-time performance was extensively tested, proving its fast execution. Running on a standard Intel Core i5 CPU with 1.6 GHz and RAM of 8 GB, the RTMCL pipeline

achieves impressive real-time performance, exceeding the recommended range of 10Hz to 30Hz with a processing speed of 122Hz. Details on execution times for various stages are provided in Table 9.

Table 5. Valuation of the performance of the UKF

State Var	Cyclist	Vehicle	Pedestrian	Landmark
p_x	0.06481	0.18570	0.06521	0.03240
p_y	0.08093	0.18991	0.06052	0.04331
v_x	0.14521	0.47451	0.53323	0.00324
v_y	0.15922	0.50752	0.54422	0.00543
ψ	0.03923	0.25803	0.20751	0.00752

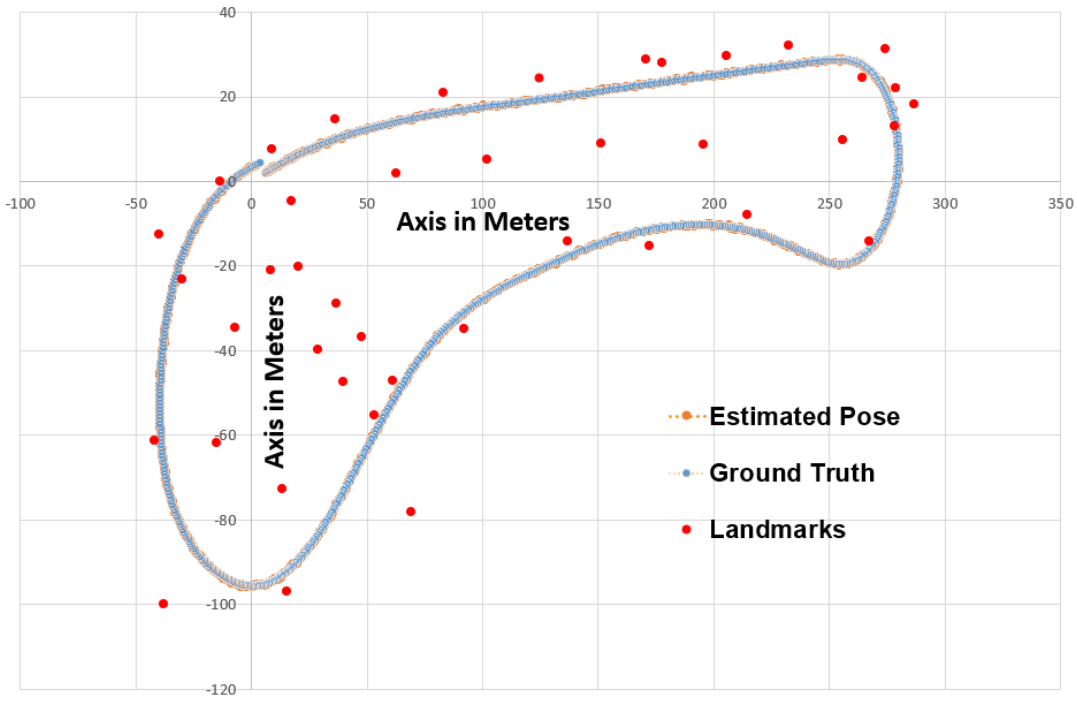

Fig. 6. Egocar localization outcomes in the shown test track

Table 6. Assessment of the performance of the UKF for the fusion of lidar and radar sensors

	Lidar+Radar	Lidar Only	Radar Only
p_x (RMSE)	0.06480	0.16122	0.20313
p_y (RMSE)	0.08091	0.14641	0.25392
v_x (RMSE)	0.14521	0.20822	0.19713
v_y (RMSE)	0.15922	0.21293	0.18711
ψ (RMSE)	0.03921	0.05403	0.04802
Average NIS	2.27973	1.69412	2.65762
Min NIS	0.00122	0.04873	0.11309
Max NIS	14.7491	12.9973	12.1832
Threshold: NIS > 95%	2.201 %	3.202 %	5.201 %

Table 7. PF convergence with various particle counts

No. of Particles	Error (x)	Error (y)	Error (Yaw)	Execution Time
15	122.3	33.00	1.596	0.2681 *msec*
25	0.138	0.124	0.005	0.4862 *msec*
50	0.114	0.115	0.004	0.7393 *msec*
100	0.115	0.107	0.004	1.2243 *msec*
150	0.110	0.106	0.004	2.0862 *msec*
200	0.110	0.104	0.004	2.4032 *msec*

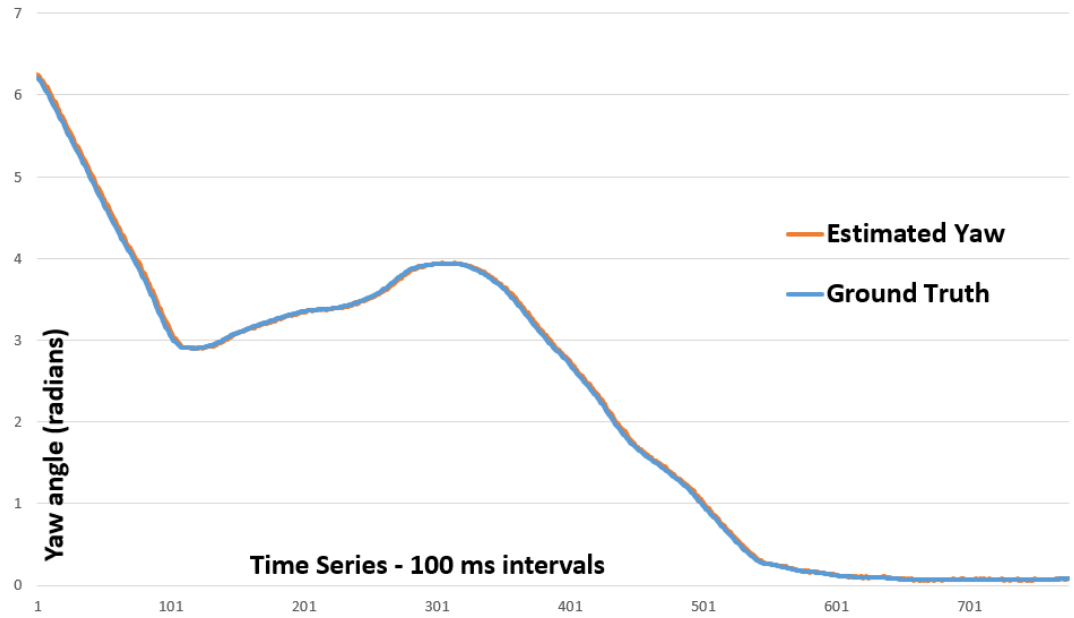

Fig. 7. The estimation of the orientation of the egocar (yaw angle) versus the ground truth in the experimenting track

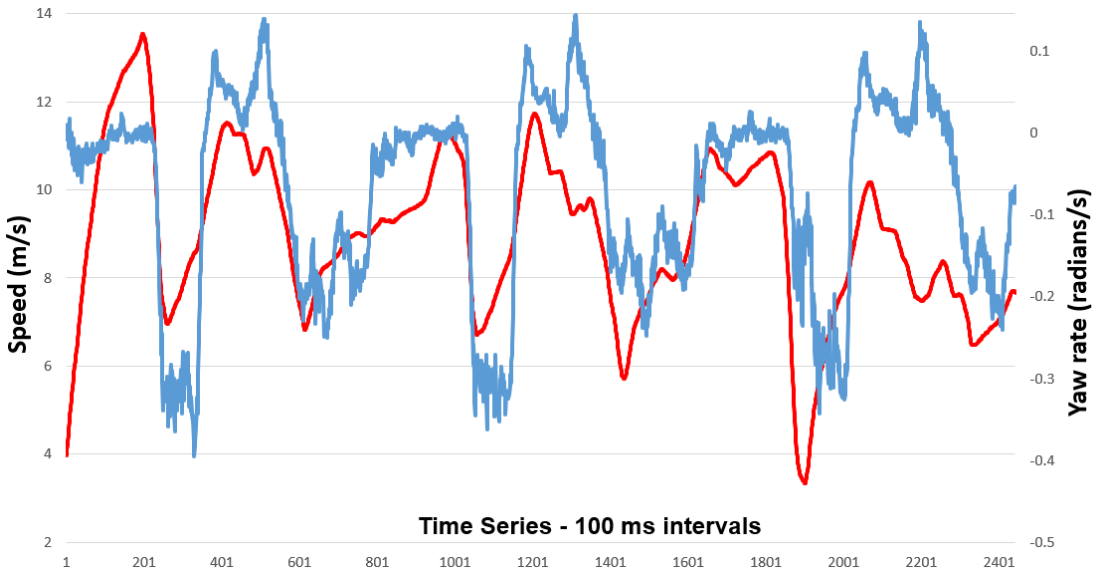

Fig. 8. A 3-lap tour on the experimenting track showing the performance of the egocar's yaw rate and speed

Table 8. Sensitivity to landmark positional variance

$\sigma_{x_{pole}}$	$\sigma_{y_{pole}}$	error (x)	error (y)	error (Yaw)
0.30	0.30	0.1144	0.1153	0.0041
0.50	0.50	0.1731	0.1632	0.0056
1.00	1.00	0.2927	0.2737	0.0099

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

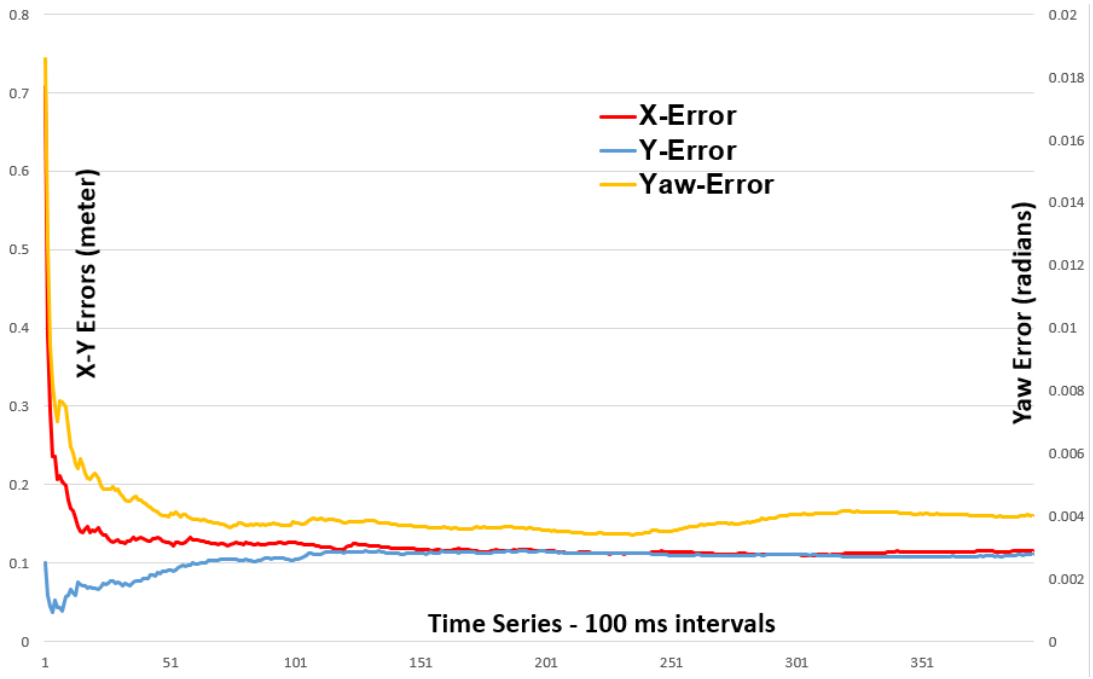

Fig. 9. The initiation phase of the PF shows the decay of errors

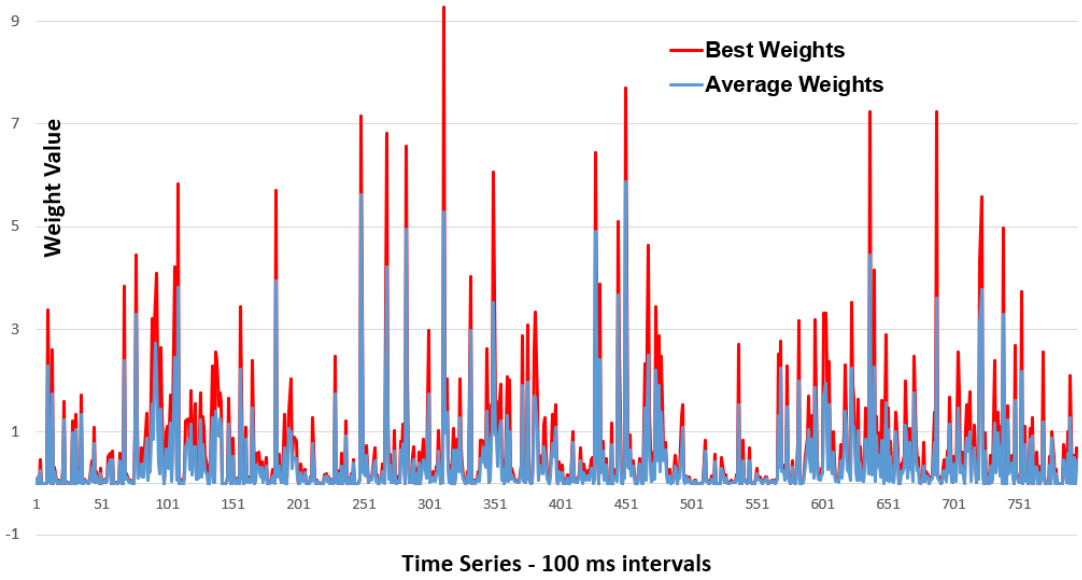

Fig. 10. The egocar's single-lap tour and the particles' weights distribution

Table 9. RTMCL processing time for each pose estimation

Task	Exec. Time (µs)
- 12-poles UKF-based state estimation:	12×439
- GB-DBSCAN + RANSAC + ICP → Clustering using + data association:	835
- Particle-Filter-based Pose estimation:	739
- 20% estimated overhead control tasks:	1368
- Execution Times Summation	8210

The RTMCL can accommodate more than 50 pole detections while still meeting the 30Hz measurement rate requirement for lidar and radar. This analysis confirms that the RTMCL's real-time performance is suitable, allowing for improved robustness by increasing the number of particles or detected poles.

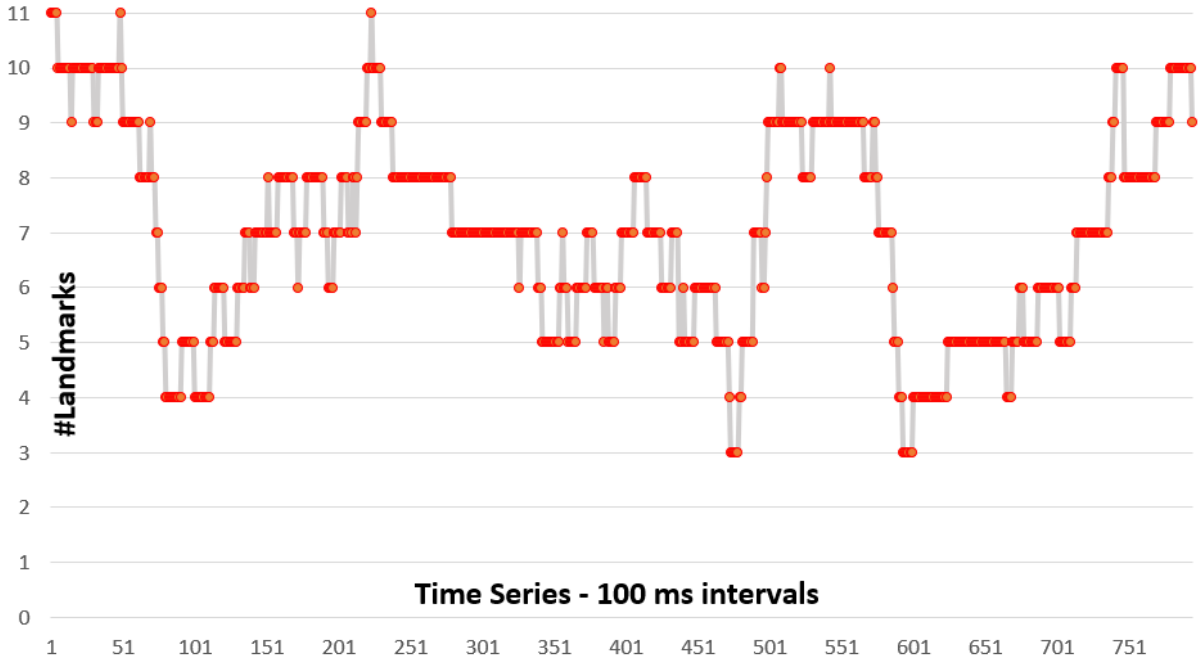

Fig. 11. The egocar's single-lap tour and the detected poles distribution

3.2. Limitations of Experimental Validation

It is crucial to acknowledge the limitations of the validation performed. The results presented were obtained using a simulated environment (CARLA [70]) with a predefined test track and accurately mapped pole-like landmarks. This controlled setting allowed for focused evaluation of the core RTMCL algorithm's performance and parameter sensitivity. However, the study did not include validation on real-world datasets or in unstructured, dynamically changing environments. Consequently, the system's performance under challenging real-world conditions such as:

- Adverse Weather: Heavy rain, dense fog, or snow simultaneously impacting both lidar and radar.
- Dense Urban Environments: Significant occlusions, multi-path reflections affecting radar, or GPS signal degradation/denial.
- Varying Landmark Densities: Scenarios with very sparse or highly cluttered/ambiguous landmarks.
- Sensor Imperfections: Unmodeled noise, calibration drift, or temporary sensor dropouts.

While the methodology is designed to be generalizable, its effectiveness in these complex scenarios would require further investigation and potentially environment-specific tuning. The current results demonstrate the potential of the RTMCL approach under idealized conditions but should be interpreted with these constraints in mind.

4. Discussion

The RTMCL system presented demonstrates a promising approach for real-time autonomous vehicle localization, achieving a mean error of approximately 11 cm in both lateral and longitudinal positioning within the simulated test environment. This performance is attributed to the synergistic combination of UKF-based radar-lidar fusion for robust landmark detection, the use of a probabilistic landmark map to handle uncertainties, and a tailored particle filter for state estimation. The system also exhibits robustness to significant landmark position uncertainties (Table 8) and confirms real-time capability (122Hz on standard CPU hardware, Table 9).

4.1. Comparison with Prior Localization Methods

RTMCL's precision compares favorably with previously reported pole-based localization techniques. Our 11 cm mean error represents a significant improvement over the 20 cm error reported by Spangenberg et al. [72] (using stereo cameras and particle/Kalman filters), the 16.4 cm mean / 99.6 cm max error by Weng et al. [47] (using lidar, IMU, GPS via Bayesian filter), and the considerably larger errors of 95 cm (longitudinal) / 49 cm (lateral) reported by Suhr et al. [54] (using maps, GPS, IMU, camera).

While these comparisons highlight RTMCL's potential accuracy advantage, it's important to consider methodological differences (sensors used, validation environments). Furthermore, contemporary research explores Deep Learning (DL) approaches for tasks relevant to autonomy [73]-[75]. While DL excels in areas like perception [73] and prediction [74], often leveraging powerful GPUs (e.g., NVIDIA Jetson/Titan X), RTMCL focuses on precise real-time *localization* using established probabilistic filtering methods. RTMCL offers advantages in interpretability and demonstrated efficiency on standard CPU hardware (Intel Core i5), making it potentially more suitable for integration into systems with constrained computational resources compared to complex end-to-end DL localization models.

4.2. Computational Trade-Offs and Practical Considerations

The real-time performance of RTMCL hinges on balancing accuracy and computational load. As shown in Table 7 and discussed previously, the choice of 50 particles offered the best compromise for the tested scenarios, achieving high accuracy without overburdening the CPU. Increasing particles improves accuracy marginally but significantly impacts latency, potentially violating real-time constraints. The UKF, GB-DBSCAN, and ICP steps also contribute to the overall 8.21ms cycle time (Table 9). While efficient, the costs of clustering and ICP can scale with data density and landmark count, respectively, potentially requiring optimization in highly complex environments. The achieved 11 cm accuracy running at 122Hz on a standard Intel i5 CPU underscores the method's practicality without reliance on specialized hardware accelerators.

4.3. Limitations and Future Directions

Despite promising results, RTMCL has limitations, primarily stemming from the simulation-based validation, as detailed in Section 3.2. The system's performance in real-world adverse weather, areas with sparse/ambiguous landmarks, or during prolonged GPS outages needs thorough investigation. The sensitivity to increased sensor noise levels or map inaccuracies beyond those simulated also warrants further study. The reliance on accurately generated offline maps and robust time synchronization between sensors are practical prerequisites.

Future work should prioritize validation on diverse real-world datasets to assess generalization and robustness across varied conditions [76]-[79]. Addressing the limitation of sparse pole landmarks could involve integrating additional features (e.g., curbs, building corners) or sensor modalities (e.g., cameras for visual landmarks, potentially leveraging techniques from [54]) to provide redundancy. Exploring adaptive tuning of noise parameters based on environmental conditions or sensor health diagnostics could further enhance robustness. Finally, implementing fault detection and mitigation strategies for handling sensor dropouts or inconsistent measurements would be crucial for safety-critical deployment.

5. Conclusions

The RTMCL system offers a novel and effective pipeline for real-time, high-accuracy self-driving car localization, achieving an impressive mean error of approximately 11 cm for both lateral and longitudinal positioning in simulation. This performance is realized through a multi-stage process integrating initial GPS/IMU pose estimation, refined state tracking via a tailored UKF fusing radar and lidar sensor data, extraction of pole-like landmarks using clustering, and final high-precision pose generation using a customized particle filter incorporating probabilistic landmark information.

Critically, RTMCL distinguishes itself from alternative approaches. Unlike computationally intensive SLAM methods or dense HD maps requiring massive storage and frequent updates, RTMCL achieves centimeter-level accuracy using lightweight, sparse landmark maps, offering a compelling balance between high precision and computational tractability. Its core innovations lie in robust radar-lidar landmark detection via UKF, which mitigates weather sensitivity often encountered in lidar-only systems [47], [49], and the explicit probabilistic modeling of landmark map uncertainties, handled effectively within the particle filter framework for superior robustness compared to deterministic matching.

The practical impact of this approach is significant. RTMCL demonstrates its capability operating effectively at over 100Hz (exceeding the typical 30Hz requirement) on standard Intel Core i5 CPU hardware. This highlights its potential viability for integration into mass-market autonomous vehicles, contrasting with some deep learning-based localization methods that often necessitate more powerful and energy-intensive GPU-based computing platforms [73].

However, several limitations must be acknowledged to provide a balanced perspective. The system's performance fundamentally relies on the presence, density, and visibility of pole-like landmarks; its effectiveness may degrade in environments where such features are sparse or consistently occluded (e.g., open highways, dense urban traffic). The validation was primarily conducted in simulation, and real-world robustness under extreme weather conditions (simultaneously affecting both radar and lidar), significant sensor noise, or during prolonged GPS/IMU signal denial (affecting initialization) requires rigorous empirical verification. The current implementation also lacks explicit fault detection and contingency plans for handling sensor failures or highly inconsistent data streams. While the 50-particle configuration proved efficient, scaling to highly complex scenarios might necessitate more particles, impacting the computational load, although the current performance suggests headroom on the tested hardware.

Future advancements for RTMCL should prioritize validation on diverse real-world datasets to thoroughly assess generalization and robustness. Enhancing performance in sparse-landmark environments could involve integrating the detection and mapping of additional static features (e.g., curbs, building corners, traffic signs) perhaps using complementary sensors like cameras or range finders. Machine learning integration holds promise for further refinement; for instance, ML techniques could potentially be explored to develop adaptive models for dynamically tuning UKF or PF noise parameters based on real-time environmental assessment or sensor confidence levels, or to improve the robustness of landmark feature extraction and data association, though this introduces considerations of training data needs, model complexity, and interpretability.

In conclusion, despite the acknowledged limitations and areas for future work, RTMCL presents a significant step towards reliable, accurate, and computationally efficient localization for autonomous vehicles, offering a strong and practical foundation built upon robust probabilistic principles.

Supplementary Materials: Availability of data and material: available upon request.

Author Contribution: WAF: Writing – original draft, Methodology, Formal analysis, Data curation, Conceptualization. MOF: Writing review & editing.

Funding: This research received no external funding.

Acknowledgment: NA.

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

References

[1] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, “A Survey of Autonomous Driving: Common Practices and Emerging Technologies,” *IEEE Access*, vol. 8, pp. 58443-58469, 2020,

https://doi.org/10.1109/ACCESS.2020.2983149.

[2] W. Farag, "A lightweight vehicle detection and tracking technique for advanced driving assistance systems," *Journal of Intelligent & Fuzzy Systems*, vol. 39, no. 3, pp. 2693–2710, 2020, https://doi.org/10.3233/JIFS-190634.

[3] W. Farag and Z. Saleh, "An advanced vehicle detection and tracking scheme for self-driving cars," *2nd Smart Cities Symposium (SCS 2019)*, pp. 1-6, 2019, https://doi.org/10.1049/cp.2019.0222.

[4] W. Farag, "Multiple Road-Objects Detection and Tracking for Autonomous Driving," *Journal of Engineering Research*, vol. 10, no. 1A, pp. 237-262, 2021, https://doi.org/10.36909/jer.10993.

[5] W. Farag, "Lidar and radar fusion for real-time road-objects detection and tracking," *Intelligent Decision Technologies*, vol. 15, no. 2, pp. 291-304, 2021, https://doi.org/10.3233/IDT-200106.

[6] W. Farag, "A Comprehensive Real-Time Road-Lanes Tracking Technique for Autonomous Driving," *International Journal of Computing and Digital Systems*, vol. 9, no. 3, pp. 349-362, 2020, https://doi.org/10.12785/ijcds/090302.

[7] W. Farag, "Real-Time Detection of Road Lane-Lines for Autonomous Driving," *Recent Advances in Computer Science and Communications*, vol. 13, no. 2, pp. 265-274, 2020, http://dx.doi.org/10.2174/2213275912666190126095547.

[8] W. Farag, "Traffic signs classification by deep learning for advanced driving assistance systems," *Intelligent Decision Technologies*, vol. 13, no. 3, pp. 305-314, 2019, https://doi.org/10.3233/IDT-180064.

[9] D. C. Guastella and G. Muscato, "Learning-Based Methods of Perception and Navigation for Ground Vehicles in Unstructured Environments: A Review," *Sensors*, vol. 21, no. 1, p. 73, 2020, https://doi.org/10.3390/s21010073.

[10] S.-J. Babak, S. A. Hussain, B. Karakas, and S. Cetin, "Control of autonomous ground vehicles: a brief technical review," *IOP Conference Series: Materials Science and Engineering*, vol. 224, no. 1, p. 012029, 2017, https://doi.org/10.1088/1757-899X/224/1/012029.

[11] W. Farag and Z. Saleh, "Traffic signs identification by deep learning for autonomous driving," *Smart Cities Symposium 2018*, pp. 1-6, 2018, https://doi.org/10.1049/cp.2018.1382.

[12] W. A. Farag and W. H. F. Aly, "Computer Vision-Based Road Vehicle Tracking For Self-Driving Car Systems," *Journal of Southwest Jiaotong University*, vol. 58, no. 3, pp. 785-802, 2023, https://doi.org/10.35741/issn.0258-2724.58.3.66.

[13] W. Farag, "Road-objects tracking for autonomous driving using lidar and radar fusion," *Journal of Electrical Engineering*, vol. 71, no. 3, pp. 138–149, 2020, https://sciendo.com/article/10.2478/jee-2020-0021.

[14] A. Woo, B. Fidan, and W. W. Melek, "Localization for Autonomous Driving," *Handbook of Position Location: Theory, Practice, and Advances, Second Edition, Second Edition*, 2019, https://doi.org/10.1002/9781119434610.ch29.

[15] R. Zekavat and R. M. Buehrer, "Localization for Autonomous Driving," *Handbook of Position Location*, pp. 1051-1087, 2018, https://doi.org/10.1002/9781119434610.ch29.

[16] W. Farag, "Kalman-filter-based sensor fusion applied to road-objects detection and tracking for autonomous vehicles," *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 235, no. 7, pp. 1125-1138, 2021, https://doi.org/10.1177/0959651820975523.

[17] H. Ma, W. Pei, and Q. Zhang, "Research on Path Planning Algorithm for Driverless Vehicles," *Mathematics*, vol. 10, no. 15, p. 2555, 2022, https://doi.org/10.3390/math10152555.

[18] W. Farag, "Recognition of traffic signs by convolutional neural nets for self-driving vehicles," *International Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 22, no. 3, pp. 205-214, 2018, https://doi.org/10.3233/KES-180385.

[19] W. Farag, M. Abouelela, and M. Helal, "Finding and Tracking Automobiles on Roads for Self-Driving Car Systems," *International Journal of Robotics and Control Systems*, vol. 3, no. 4, pp. 704-727, 2023, https://doi.org/10.31763/ijrcs.v3i4.1022.

[20] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough and A. Mouzakitis, "A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 829-846, 2018, https://doi.org/10.1109/JIOT.2018.2812300.

[21] W. Farag, "Complex track maneuvering using real-time MPC control for autonomous driving," *International Journal of Computing and Digital Systems*, vol. 90, no. 5, 2020, https://doi.org/10.12785/ijcds/090511.

[22] W. Farag, "Complex-Track Following in Real-Time Using Model-Based Predictive Control," *International Journal of Intelligent Transportation Systems Research*, vol. 19, pp. 112-127, 2021, https://doi.org/10.1007/s13177-020-00226-1.

[23] W. Farag, "Track maneuvering using PID control for self-driving cars," *Recent Advances in Electrical and Electronic Engineering*, vol. 13, no. 1, pp. 91-100, 2020, http://dx.doi.org/10.2174/2352096512666190118161122.

[24] W. Farag and Z. Saleh, "Tuning of PID Track Followers for Autonomous Driving," *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 1-7, 2018, https://doi.org/10.1109/3ICT.2018.8855773.

[25] W. Farag and Z. Saleh, "Road Lane-Lines Detection in Real-Time for Advanced Driving Assistance Systems," *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 1-8, 2018, https://doi.org/10.1109/3ICT.2018.8855797.

[26] Y. Liu, X. Pei, X. Guo, C. Chen, and H. Zhou, "An integration planning and control method of intelligent vehicles based on the iterative linear quadratic regulator," *Journal of the Franklin Institute*, vol. 361, no. 1, pp. 265-282, 2024, https://doi.org/10.1016/j.jfranklin.2023.11.046.

[27] W. Farag, "Multi-Agent Reinforcement Learning using the Deep Distributed Distributional Deterministic Policy Gradients Algorithm," *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, pp. 1-6, 2020, https://doi.org/10.1109/3ICT51146.2020.9311945.

[28] A. Schaefer, D. Büscher, J. Vertens, L. Luft and W. Burgard, "Long-Term Urban Vehicle Localization Using Pole Landmarks Extracted from 3-D Lidar Scans," *2019 European Conference on Mobile Robots (ECMR)*, pp. 1-7, 2019, https://doi.org/10.1109/ECMR.2019.8870928.

[29] J. Levinson, M. Montemerlo, and S. Thrun, "Map-Based Precision Vehicle Localization in Urban Environments," *Robotics: Science and Systems III*, 2024, https://www.roboticsproceedings.org/rss03/p16.pdf.

[30] F. Lu, G. Chen, J. Dong, X. Yuan, S. Gu and A. Knoll, "Pole-based Localization for Autonomous Vehicles in Urban Scenarios Using Local Grid Map-based Method," *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 640-645, 2020, https://doi.org/10.1109/ICARM49381.2020.9195330.

[31] L. de Paula Veronese *et al*., "Evaluating the Limits of a LiDAR for an Autonomous Driving Localization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1449-1458, 2021, https://doi.org/10.1109/TITS.2020.2971054.

[32] T. Zhou, M. Yang, K. Jiang, H. Wong, and D. Yang, "MMW Radar-Based Technologies in Autonomous Driving: A Review," *Sensors*, vol. 20, no. 24, p. 7283, 2020, https://doi.org/10.3390/s20247283.

[33] A. Takanose, Y. Atsumi, K. Takikawa, and J. Meguro, "Improvement of Reliability Determination Performance of Real-Time Kinematic Solutions Using Height Trajectory," *Sensors*, vol. 21, no. 2, p. 657, 2021, https://doi.org/10.3390/s21020657.

[34] S. S. Rathour, A. Boyali, L. Zheming, Seiichi. Mita, and V. John, "A Map-based Lateral and Longitudinal DGPS/DR Bias Estimation Method for Autonomous Driving," *International Journal of Machine Learning and Computing*, vol. 7, no. 4, pp. 67-71, 2017, https://www.ijml.org/vol7/622-S78.pdf.

[35] N. Carlevaris-Bianco, A. Ushani, and R. M. Eustice, "NCLT Dataset: The University of Michigan North Campus Long-Term Vision and LIDAR Dataset," *The University of Michigan*, 2016, https://robots.engin.umich.edu/nclt/.

[36] M. Modsching, R. Kramer, and K. Hagen, "Field trial on GPS accuracy in a medium-size city: the influence of built-up," *3rd Workshop on Positioning, Navigation, and Communication*, pp. 209-218, 2006, http://www.modsching.com/papers/FieldtrialonGPSAccuracy10pages2006-02-10_06crcit.pdf.

[37] A. Rakhmanov and Y. Wiseman, "Compression of GNSS Data with the Aim of Speeding up Communication to Autonomous Vehicles," *Remote Sensing*, vol. 15, no. 8, p. 2165, 2023, https://doi.org/10.3390/rs15082165.

[38] W. Li, Z. Li, W. Jiang, Q. Chen, G. Zhu, and J. Wang, "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 Sensing*, vol. 14, no. 5, p. 1295, 2022, https://doi.org/10.3390/rs14051295.

[39] W. Farag, "Robot arm navigation using deep deterministic policy gradient algorithms," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 35, no. 5, pp. 617-627, 2023, https://doi.org/10.1080/0952813X.2021.1960640.

[40] W. Farag, "Real-time lidar and radar fusion for road-objects detection and tracking," *International Journal of Computational Science and Engineering*, vol. 24, no. 5, p. 517, 2021, https://doi.org/10.1504/IJCSE.2021.118100.

[41] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," *2010 IEEE International Conference on Robotics and Automation*, pp. 4372-4378, 2010, https://doi.org/10.1109/ROBOT.2010.5509700.

[42] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni and J. B. Tenenbaum, "Synthesizing 3D Shapes via Modeling Multi-view Depth Maps and Silhouettes with Deep Generative Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2511-2519, 2017, https://doi.org/10.1109/CVPR.2017.269.

[43] X. Qin, Y. Luo, N. Tang, G. Li, "Making data visualization more efficient and effective: a survey," *The VLDB Journal*, vol. 29, no. 1, pp. 93-117, 2020, https://doi.org/10.1007/s00778-019-00588-3.

[44] William E. Lorensen, Harvey E. Cline, "Marching cubes: A high-resolution 3D surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163-169, 1987, https://doi.org/10.1145/37402.37422.

[45] K. Wong, Y. Gu and S. Kamijo, "Mapping for Autonomous Driving: Opportunities and Challenges," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 1, pp. 91-106, 2021, https://doi.org/10.1109/MITS.2020.3014152.

[46] J. Kümmerle, M. Sons, F. Poggenhans, T. Kühner, M. Lauer and C. Stiller, "Accurate and Efficient Self-Localization on Roads using Basic Geometric Primitives," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5965-5971, 2019, https://doi.org/10.1109/ICRA.2019.8793497.

[47] L. Weng, M. Yang, L. Guo, B. Wang and C. Wang, "Pole-Based Real-Time Localization for Autonomous Driving in Congested Urban Scenarios," *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pp. 96-101, 2018, https://doi.org/10.1109/RCAR.2018.8621688.

[48] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981, https://doi.org/10.1145/358669.358692.

[49] M. Sefati, M. Daum, B. Sondermann, K. D. Kreisköther and A. Kampker, "Improving vehicle localization using semantic and pole-like landmarks," *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 13-19, 2017, https://doi.org/10.1109/IVS.2017.7995692.

[50] W. Farag, "Navigation of Robotic-Arms using Policy Gradient Reinforcement Learning," *International Journal of Computing and Digital Systems*, vol. 12, no. 1, 2022, https://doi.org/10.12785/ijcds/120171.

[51] A. Schaefer, D. Büscher, J. Vertens, L. Luft, and W. Burgard, "Long-term vehicle localization in urban environments based on pole landmarks extracted from 3-D lidar scans," *Robotics and Autonomous Systems*, vol. 136, p. 103709, 2021, https://doi.org/10.1016/j.robot.2020.103709.

[52] K. Chiang, Y. Chiu, S. Srinara, and M. Tsai, "Performance of LiDAR-SLAM-based PNT with initial poses based on NDT scan matching algorithm," *Satellite Navigation*, vol. 4, no. 1, p. 3, 2023, https://doi.org/10.1186/s43020-022-00092-0.

[53] H. Taheri and Z. C. Xia, "SLAM; definition and evolution," *Engineering Applications of Artificial Intelligence*, vol. 97, p. 104032, 2021, https://doi.org/10.1016/j.engappai.2020.104032.

[54] J. K. Suhr, J. Jang, D. Min and H. G. Jung, "Sensor Fusion-Based Low-Cost Vehicle Localization System for Complex Urban Environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1078-1086, 2017, https://doi.org/10.1109/TITS.2016.2595618.

[55] F. Lu and E. Milios, "Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans," *Journal of Intelligent and Robotic Systems*, vol. 18, no. 3, pp. 249-275, 1997, https://doi.org/10.1023/A:1007957421070.

[56] S. Thrun, "Particle Filters in Robotics," *Proceedings of Uncertainty in AI (UAI) 2002*, 2002, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e6604ff9a3cceb37329096df80eb9cec54a385b9.

[57] B. Chen, L. Dang, N. Zheng, J. C. Principe, "Kalman Filtering Under Information Theoretic Criteria," *Kalman Filtering Under Information Theoretic Criteria*, pp. 89-126, 2023, https://doi.org/10.1007/978-3-031-33764-2_4.

[58] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pp. 153-158, 2000, https://doi.org/10.1109/ASSPCC.2000.882463.

[59] G. A. Einicke and L. B. White, "Robust extended Kalman filtering," *IEEE Transactions on Signal Processing*, vol. 47, no. 9, pp. 2596-2599, 1999, https://doi.org/10.1109/78.782219.

[60] W. Farag, "Synthesis of intelligent hybrid systems for modeling and control," *University of Waterloo*, 1998, https://uwspace.uwaterloo.ca/bitstream/handle/10012/238/NQ30606.pdf;sequence=1.

[61] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226-231, 1996, http://file.biolab.si/papers/1996-DBSCAN-KDD.pdf.

[62] D. Kellner, J. Klappstein and K. Dietmayer, "Grid-based DBSCAN for clustering extended objects in radar data," *2012 IEEE Intelligent Vehicles Symposium*, pp. 365-370, 2012, https://doi.org/10.1109/IVS.2012.6232167.

[63] R. A. Brown, "Building a Balanced k-d Tree in O(kn log n) Time," *Journal of Computer Graphics Techniques (JCGT)*, vol. 4, no. 1, pp. 50-68, 2015, http://jcgt.org/published/0004/01/03/.

[64] W. A. Farag, J. M. H. Barakat, "Utilizing Probabilistic Maps and Unscented-Kalman-Filtering-Based Sensor Fusion for Real-Time Monte Carlo Localization," *World Electric Vehicle Journal*, vol. 15, no. 1, p. 5, 2024, https://doi.org/10.3390/wevj15010005.

[65] M. Nagiub and W. Farag, "Automatic selection of compiler options using genetic techniques for embedded software design," *2013 IEEE 14th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 69-74, 2013, https://doi.org/10.1109/CINTI.2013.6705166.

[66] K. H. A. Faraj, K. H. Ahmed, T. N. A. A. Attar, W. M. Hameed, A. B. Kanbar, "Response time analysis for XAMPP server based on different versions of linux operating system," *The Scientific Journal of Cihan University–Sulaimaniya*, vol. 4, no. 2, pp. 102-114, 2020, https://doi.org/10.25098/4.2.23.

[67] A. Chalvatzaras, I. Pratikakis and A. A. Amanatiadis, "A Survey on Map-Based Localization Techniques for Autonomous Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1574-1596, 2023, https://doi.org/10.1109/TIV.2022.3192102.

[68] S. Zhao and B. Huang, "On initialization of the Kalman filter," *2017 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, pp. 565-570, 2017, https://doi.org/10.1109/ADCONIP.2017.7983842.

[69] R. Piché, "Online tests of Kalman filter consistency," *International Journal of Adaptive Control and Signal Processing*, vol. 30, no. 1, pp. 115-124, 2016, https://doi.org/10.1002/acs.2571.

[70] D. R. Niranjan, B. C. VinayKarthik, "Deep learning based object detection model for autonomous driving research using carla simulator," *2021 2nd international conference on smart electronics and*

communication (ICOSEC), pp. 1251-1258, 2021, https://doi.org/10.1109/ICOSEC51865.2021.9591747.

[71] W. Farag, "Real-Time Autonomous Vehicle Localization Based on Particle and Unscented Kalman Filters," *Journal of Control, Automation and Electrical Systems*, vol. 32, no. 2, pp. 309-325, 2021, https://doi.org/10.1007/s40313-020-00666-w.

[72] R. Spangenberg, D. Goehring and R. Rojas, "Pole-based localization for autonomous vehicles in urban scenarios," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2161-2166, 2016, https://doi.org/10.1109/IROS.2016.7759339.

[73] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao and D. Li, "Object Classification Using CNN-Based Fusion of Vision and LIDAR in Autonomous Vehicle Environment," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4224-4231, 2018, https://doi.org/10.1109/TII.2018.2822828.

[74] H. Gao *et al*., "Trajectory prediction of cyclist based on dynamic Bayesian network and long short-term memory model at unsignalized intersections," *Science China Information Sciences*, vol. 64, no. 7, p. 172207, 2021, https://doi.org/10.1007/s11432-020-3071-8.

[75] W. A. Farag, V. H. Quintana and G. Lambert-Torres, "Neuro-fuzzy modeling of complex systems using genetic algorithms," *Proceedings of International Conference on Neural Networks (ICNN'97)*, vol. 1, pp. 444-449, 1997, https://doi.org/10.1109/ICNN.1997.611709.

[76] W. Farag, "Real-time NMPC path tracker for autonomous vehicles," *Asian Journal of Control*, vol. 23, no. 4, pp. 1952-1965, 2021, https://doi.org/10.1002/asjc.2335.

[77] W. A. Farag, M. Fayed, "Advancing vehicle detection for autonomous driving: integrating computer vision and machine learning techniques for real-world deployment," *Journal of Control and Decision, Taylor & Francis*, 2025, https://doi.org/10.1080/23307706.2025.2469893.

[78] W. A. Farag, M. Helal, "Real-time localization with probabilistic maps and unscented kalman filtering: a dynamic sensor fusion approach," *Journal of Control and Decision*, 2024, https://doi.org/10.1080/23307706.2024.2417218.

[79] Y. Kawai, J. Padron, Y. Yokokura, K. Ohishi and T. Miyazaki, "Equivalent Disturbance Compensator and Friction Compensation for Back-Forward Drivability Improvement," *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1-6, 2021, https://doi.org/10.1109/IECON48115.2021.9589468.