sensors

MDPI

Review

Self-Driving Car Location Estimation Based on a Particle-Aided Unscented Kalman Filter

Ming Lin, Jaewoo Yoon and Byeongwoo Kim *

Department of Electrical Engineering, University of Ulsan, 93 Daehak-ro, Nam-gu, Ulsan 44610, Korea; flaaud159@naver.com (M.L.); jaewoo127@naver.com (J.Y.)

* Correspondence: bywokim@ulsan.ac.kr

Received: 27 March 2020; Accepted: 28 April 2020; Published: 29 April 2020

Abstract: Localization is one of the key components in the operation of self-driving cars. Owing to the noisy global positioning system (GPS) signal and multipath routing in urban environments, a novel, practical approach is needed. In this study, a sensor fusion approach for self-driving cars was developed. To localize the vehicle position, we propose a particle-aided unscented Kalman filter (PAUKF) algorithm. The unscented Kalman filter updates the vehicle state, which includes the vehicle motion model and non-Gaussian noise affection. The particle filter provides additional updated position measurement information based on an onboard sensor and a high definition (HD) map. The simulations showed that our method achieves better precision and comparable stability in localization performance compared to previous approaches.

Keywords: particle filter; sensor fusion; self-driving car; unscented Kalman filter; vehicle model; Monte Carlo localization

1. Introduction

In recent years, research on self-driving cars has gained much prominence. The ultimate goal of self-driving cars is to transport people from one place to another without any help from a driver. A self-driving system must control numerous parameters, including speed, orientation, acceleration, and maneuvering, in order to drive without any human assistance. All of these control parameters are controlled by the decision-making module, which handles all perception data from the vehicle and sensors. The perception module determines the relationship between the ego vehicle and the surrounding environment. One of the most important algorithm modules is vehicle localization because all the sensors sense the environment based on local vehicle coordinates [1]. The typical perception sensors of a self-driving car are the camera, radar, light detection and ranging (LIDAR), 2D laser scanner, global positioning system (GPS), and inertial measurement unit (IMU) [2]. GPS is the most commonly used navigation system in self-driving cars. However, because of issues with multipath routing and poor signal availability in cities, relying entirely on GPS is not suitable for localizing vehicles in urban environments. Although differential GPS systems can be used, the high cost and size of these systems limit their implementation [3]. Furthermore, GPS systems cannot be used in tunnels or indoor environments. Vision-based localization has been proposed as a method for localizing vehicles using a low-cost camera. However, the vision-based localization algorithm is easily affected by weather and light conditions, leading to insufficient accuracy and stability [4–8]. LIDAR-based map matching can yield highly precise results with the help of a high definition (HD) map. By contrast, matching requires the environment to be accurately mapped such that the point cloud of the environment does not change. Point cloud matching is expensive and requires considerable power and computation resources [9–16]. Thus, developing a low-cost localization method that can utilize the vehicle's sensors and road infrastructure to achieve precise and stable

location performance is necessary for furthering research on self-driving cars. One of the localization solutions that can be used in complex urban environments is vehicle localization based on local sensor systems and information from the HD map. Matching an entire point cloud with an HD map is inefficient; therefore, only the ground truth location map of infrastructures is used in this study, for computational efficiency. In previous research, vehicle localization based on vehicle-to-vehicle (V2V) and vehicular ad-hoc network (VANET) communication was proposed. The basic condition is that these algorithms require surrounding vehicles to be equipped with V2V communication equipment, which are then referred to for the infrastructures [17–23]. In this study, we only consider the case of a single vehicle. Moreover, because the vehicle data contain a large amount of noise, an efficient filtering algorithm is needed to obtain precise localization results.

The methods for vehicle localization have improved considerably over the years. The primary methodology that was used is the probabilistic approach. The Kalman filter (KF) is an optimal estimator that is designed for processing Gaussian noise with mean and variance, and it is an important component in several such approaches [24]. One of the assumptions of the KF is that the noise should be Gaussian. However, in practice, a function like the trigonometric filter renders the Gaussian noise non-Gaussian. Therefore, an extended Kalman filter (EKF), which uses a low order Taylor expansion to linearize the nonlinear (e.g., trigonometric) function, has been proposed. It uses a partial derivative to represent the rate of change of the nonlinear functions, which aims to keep the noise Gaussian. If the state is a vector, then the partial derivative parameters can be assembled into a new matrix, which is called a Jacobian matrix. Generally, in order to localize the vehicle's position, researchers derive the Jacobian matrix based on the transition and measurement models for handling the vehicle's noisy sensor data [25–30]. If an EKF based on a Jacobian matrix approximates a nonlinear function using a high order of Taylor series, it also works well in transforming nonlinear functions into linear ones. The critical problem, however, is that the Jacobian matrix is difficult to derive for complex dynamics. Therefore, a new, sample region-based Kalman filter, which is called the unscented Kalman filter (UKF), was proposed. The UKF performs better than the EKF and KF when the system model is highly nonlinear [31–33]. The UKF uses some key points, which are called sigma points, to approximate the non-Gaussian noise into Gaussian based on the unscented transform. In this way, it can properly capture the nonlinearity. Furthermore, because the UKF approximates the non-Gaussian noise with sigma points, it is easy to combine other information when selecting the sigma points and there is no need to calculate the Jacobian matrix.

The basic assumption of the Kalman filter family is that noise is Gaussian. In the real world, most noise does not have a Gaussian property. For processing non-Gaussian noise, a Monte Carlo-based localization approach, called particle filter (PF), has been proposed [34,35]. The particle filter uses several samples, referred to as particles, to approximate the non-Gaussian property. Because the particles are generated randomly, they can represent the properties of non-Gaussian noise precisely if there are sufficient numbers. However, a vehicle has limited computational resources; therefore, it cannot allow the particle filter to approximate the number of particles. Therefore, there is a trade-off between precision and computational resources when generating an effective particle-based system model. Thus, an extended Kalman filter-aided particle filter, called an extended particle filter (EPF), and an unscented particle filter (UPF), called the Kalman filter-aided particle filter, have been proposed [36–38]. Both the EPF and the UPF use system models to generate and update the particles. It should be noted that each particle should compute the sigma points or Jacobian matrix; therefore, both the EPF and UPF are computationally inefficient and difficult to implement [39–41].

In this study, we propose a new method, the particle-aided unscented Kalman filter (PAUKF), for vehicle localization. With the help of the particle filter, the unscented Kalman filter can estimate a system with high nonlinearity and various sources of nonlinear noise more precisely. Because each particle does not have to update the sigma points or share the prediction model, this method requires fewer computational resources. The computational burden and precision of PAUKF can be easily tuned by tuning the quantity of the sigma points and particles. The results of the simulation show that the PAUKF estimates the vehicle's position and state more accurately than other methods

that use a limited number of particles. Section 2 illustrates the methodology of the PAUKF. Section 3 details the simulation conditions, and Section 4 presents the analysis of the simulation results. Finally, Section 5 presents the conclusion of this paper.

2. Particle-Aided Unscented Kalman Filter

This section describes the implementation of the PAUKF, including particle implementation and PAUKF implementation. Both the PF and UKF are Bayesian-based filters, and the environment is assumed to be Markov, which means that the PAUKF also has a Markov assumption.

2.1. Particle Filter Algorithm

The particle filter is a Monte Carlo-based method that can handle both Gaussian noise and non-Gaussian noise [42]. Because the vertical movement of the vehicle is small, we only consider the vehicle in a two-dimensional Cartesian space with the vehicle heading θ. A bicycle model is used in this study to represent the motion of the vehicle because a complex vehicle model aggravates the computational burden, and many parameters cause additional noise [43]. The state of the vehicle is represented by <x, y, and θ>, as shown in Figure 1.

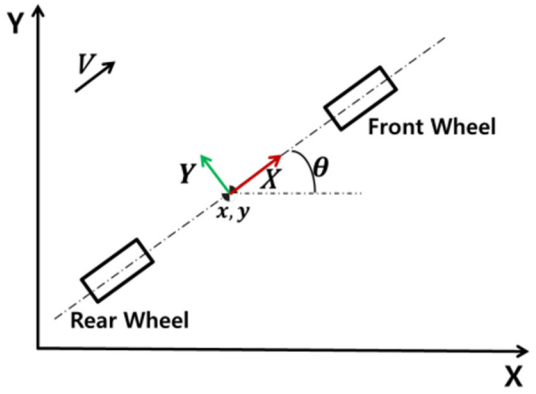

Figure 1. Vehicle state in two-dimensional Cartesian space.

The inputs that we use are the range sensor and the ground truth of the infrastructures in the HD map. The final position of the vehicle should be the best posterior belief based on past data and the current state. The particle filter is a nonparametric implementation of the Bayes filter, which uses a finite number of samples to approximate the posterior. Thus, the final belief bel(x) should be generated for each particle by using each important factor (weight), as shown in Equation (1). The $x_{[1,2\ldots N]}$ means the state vector of each particle and $w_{[1,2\ldots N]}$ is the weight of each particle. The size of each particle X was 3 × 1. W is a non-negative factor termed as the importance factor. In this study, we used 100 particles for simulation, which means that N is 100. The larger the importance factor, the more it affects the final estimation result.

$$\mathrm{bel}(x) = \sum_{i=1}^{N} x_N w_N. \quad (1)$$

The particle filter for localization can be divided into four parts, which are introduced in the following subsections.

2.1.1. Initialization

To localize the vehicle in global coordinates, it is essential to provide an initial location to the vehicle. Otherwise, the vehicle will search for its position over the entire world. Therefore, we use the GPS sensor for initialization. Even though the GPS signal is poor due to multipath and blocking issues, it still provides a limited area for the vehicle to localize. Because the particle filter is recursive, after initialization, the noisy GPS signal data are filtered recursively. When a particle filter receives GPS data, it generates N random particles for initialization.

2.1.2. Prediction

To obtain the prior belief, each particle at timestamp $k-1$ should predict the current state based on the system prediction model. The prediction model was constructed based on the vehicle model. Complex models, such as a dynamic model with tires, can also be included. However, complex vehicle models reduce computation efficiency. Such models also require detailed vehicle parameters, which are difficult to set. Incorrect parameters can cause noisy estimations. Considering the computational burden and precision, a kinematic model was used in this study [43]. We ignore the slip angle because vehicle travel in cities is typically not fast.

$$\bar{X}_{k+1} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{\theta} \end{bmatrix}_{k+1} = \begin{bmatrix} \frac{v_k}{\theta_k(\dot{\Delta t})}[\sin(\theta_k + \theta_k(\dot{\Delta t})) - \sin(\theta_k)] \\ \frac{v_k}{\theta_k(\dot{\Delta t})}[\cos(\theta_k) - \cos(\theta_k + \theta_k(\dot{\Delta t}))] \\ \theta_k(\dot{\Delta t}) \end{bmatrix} + \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_k. \quad (2)$$

As Equation (2) shows, the prediction contains several trigonometric functions, which correspond to a highly nonlinear prediction model. The theta angle of each particle is critical because it changes according to the local vehicle coordinates. The kinematic model incorporates several assumptions such as the value of $\dot{\theta}$ being equal to zero.

2.1.3. Weight Calculation

Weight is also an important factor that can heavily influence particle motion. Measurements from the range sensor were used to calculate the weight. We assume that the vehicle can receive all the data from the vehicle-to-everything (V2X), HD map, and range sensor. Thus, the vehicle can receive the distance data and orientation between the vehicle and every exterior infrastructure. Here, d_i is the measured distance between the vehicle and infrastructure i, ϵ_d is the distance measurement noise, and $\Delta\theta$ is the relative orientation angle of the vehicle and infrastructure i. As Equations (3) and (4) show, the measurement model is nonlinear in nature.

$$Z_{k+1} = f(\bar{X}_{k+1}) + \text{Noise}_{k+1}, \quad (3)$$

$$Z_{k+1} = \begin{bmatrix} d_{[i]} \\ \Delta\theta_{[i]} \end{bmatrix}_{k+1} = \begin{bmatrix} \sqrt{(\bar{x}_k - x_{b,i})^2 + (\bar{y}_k - y_{b,i})^2} \\ \arctan\left(\frac{\bar{x}_k - x_{b,i}}{\bar{y}_k - y_{b,i}}\right) \end{bmatrix}_{k+1} + \begin{bmatrix} \epsilon_d \\ -\theta_v + \epsilon_{\Delta\theta} \end{bmatrix}_{k+1}. \quad (4)$$

To evaluate the weight, a multivariable normal distribution function was used to assess the importance of each particle. Thus, a multivariable normal distribution function returns the weight of a particle based on the newest sensor measurement values and the predicted values from the model as

$$w_i = p(x_i, y_i) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{(x-\mu_x)^2}{2\sigma_{\bar{x}}^2} + \frac{(y-\mu_y)^2}{2\sigma_{\bar{y}}^2}\right)} \quad i = 1,2 \ldots N. \quad (5)$$

2.1.4. Resampling

After calculating the weight and prediction values, the particle filter should select the particle along with its corresponding weight, which is the resampling step.

The entire PF part of the algorithm's pseudo code is shown in Table 1.

Table 1. Pseudo code of the particle filter.

Order	Process
1	Start one sample time iteration
2	Initialization $X_{1,2...N}$ particles
3	For 1 to N do
4	$\overline{X}_{k+1} = \text{prediction model}(X_k, u_t)$
5	End for
6	$Z_{k+1} = \text{measurent input}$
7	$w_{[1,2...N]} = \text{multivariable normal distribution}(\overline{X}_{k+1}, Z_{k+1}, \sigma_{distance}, \sigma_{orientation})\overline{X}_{k+1}$
8	Return $\widehat{X}_{PF} = f(X_{k+1}^{[1,2...N]}, w_{[1,2...N]})$
9	End one sample time iteration

2.2. Particle-Aided Unscented Kalman Filter Algorithm

The particle filter algorithm is introduced in Section 2.1. The particle estimates the position of the vehicle by using the range sensor. The final results for each particle contain information about the surrounding infrastructures and the position of the ego vehicle. Therefore, it can be concluded that this sensor provides more accurate results. When the UKF estimates the state, the results from the particle filter will be the measurement value of the vehicle. Subsequently, the PAUKF can extract a more precise result based on the particle filter estimation results. A flowchart of the PAUKF is shown in Figure 2.

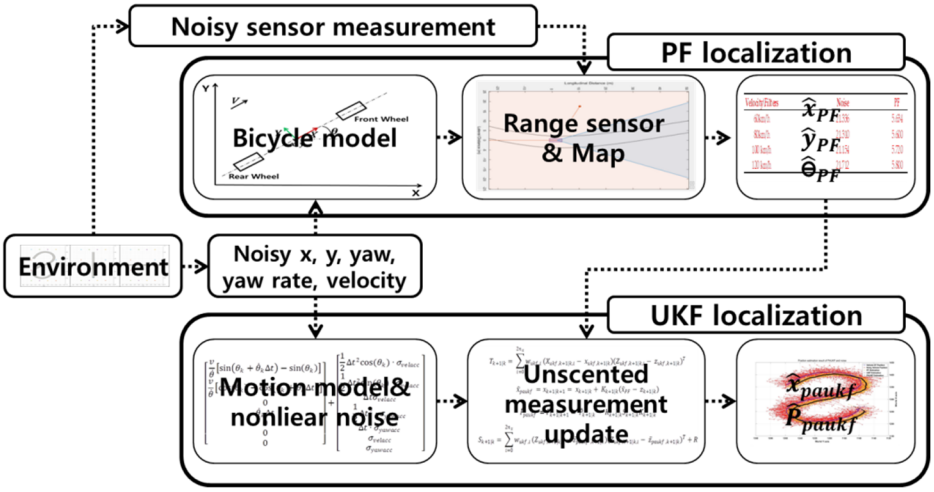

Figure 2. Particle-aided unscented Kalman filter algorithm flowchart.

The UKF is a Bayesian filter that has better performance than the EKF when estimating the state of a discrete-time nonlinear dynamic system. Because it is based on a Kalman filter, the framework of a UKF is almost the same as that of a KF. The difference is that a UKF performs stochastic linearization by using the weighted statistical linear regression process, known as an unscented transform. Instead of using a Taylor expansion, a UKF deterministically extracts the mean and covariance using the sigma points.

The sigma points are predefined by an empirical parameter λ that is calculated using Equation (6). The sigma point is a symmetrical region around the mean value. $P_{k|k}$ is the covariance matrix of the state, which updates at every iteration. The state vector of the vehicle is x_k, which is a 5 × 1 vector, as shown in Equation (7). The state vector value is the mean of the sigma matrix. n_x is the quantity of the state vector with a size of 5. Then, the sigma points are generated using Equation (8).

$$\lambda = 3 - n_x, \tag{6}$$

$$x_{paukf,k} = [x \quad y \quad v \quad \theta \quad \dot{\theta}]^T, \tag{7}$$

$$X_{paukf,k} = (\mu_k,\ \mu_k + \sqrt{(\lambda + n_x)P_k},\ \mu_k - \sqrt{(\lambda + n_x)P_k}). \quad (8)$$

After it generates the sigma points, a UKF needs a prediction model to determine the prior probability of the state. To obtain a more precise position regarding position, a UKF considers more states of the vehicle and the effect of nonlinear noise on the states.

The constant turn rate and velocity magnitude (CTRV) vehicle motion model was used in this study [44]. Based on the CTRV model, the discrete state transition model is derived by integrating the differential equation of the state, which considers the nonlinear process noise vector. Considering the underlying structure of the vehicle for a real-world test, the acceleration and yaw acceleration are considered to be noise. In particular, the acceleration and yaw acceleration noise effects are nonlinear. Therefore, the process noise cannot be handled by addition alone. In order to handle nonlinear noise, it is considered to be a state, as shown in Equation (9). This means that the size of x_{ukf} becomes $x_{k,aug}$, which is a 7 × 1 vector.

$$x_{paukf,k,aug} = \begin{bmatrix} x & y & v & \theta & \dot{\theta} & w_{velacc} & w_{yawacc} \end{bmatrix}^T. \quad (9)$$

The process noises w_{velacc} and w_{yawacc} are set as normal Gaussian distributions with variances of ${\sigma_{velacc}}^2$ and ${\sigma_{yawacc}}^2$, respectively, as shown in Equations (10) and (11).

$$w_{velacc} \sim N(0,\ {\sigma_{velacc}}^2), \quad (10)$$

$$w_{yawacc} \sim N(0, {\sigma_{yawacc}}^2). \quad (11)$$

The covariance matrix P_k is also augmented into $P_{k,aug}$, which has a size of 7 × 7, as shown in Equation (12).

$$P_{k,aug} = \begin{bmatrix} P_k & 0 & 0 \\ 0 & {\sigma_{velacc}}^2 & 0 \\ 0 & 0 & {\sigma_{yawacc}}^2 \end{bmatrix}. \quad (12)$$

The process model is derived based on the CTRV assumption and noise, as shown in Equation (13). The model that we derive has highly nonlinear properties, as indicated in Equation (14).

$$x_{paukf,k+1} = x_{paukf,k} + \int_k^{k+1} \dot{x}_{paukf,k}\, dt, \quad (13)$$

$$x_{paukf,k+1,aug} = f(x_{paukf,k,aug}, \sigma_{velacc}, \sigma_{yawacc}) = x_{paukf,k,aug} + \begin{bmatrix} \frac{v}{\dot{\theta}}\left[\sin(\theta_k + \dot{\theta}_k\Delta t) - \sin(\theta_k)\right] \\ \frac{v}{\dot{\theta}}\left[\cos(\theta_k) - \cos(\theta_k + \dot{\theta}_k\Delta t)\right] \\ 0 \\ \dot{\theta}_k\Delta t \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{v}{\dot{\theta}}\left[\sin(\theta_k + \dot{\theta}_k\Delta t) - \sin(\theta_k)\right] \\ \frac{v}{\dot{\theta}}\left[\cos(\theta_k) - \cos(\theta_k + \dot{\theta}_k\Delta t)\right] \\ 0 \\ \dot{\theta}_k\Delta t \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2\cos(\theta_k)\cdot\sigma_{velacc} \\ \frac{1}{2}\Delta t^2\sin(\theta_k)\cdot\sigma_{velacc} \\ \Delta t\sigma_{velacc} \\ \frac{1}{2}\Delta t^2\cdot\sigma_{yawacc} \\ \Delta t\cdot\sigma_{yawacc} \\ \sigma_{velacc} \\ \sigma_{yawacc} \end{bmatrix}. \quad (14)$$

In the augmented prediction step, n_x, should be the quantity of the augmented state $n_{x,aug}$ with a size of 7, and λ also needs to be calculated as in Equation (15). Subsequently, the sigma points are predicted using Equation (16).

$$\lambda = 3 - n_{x,aug}, \tag{15}$$

$$X_{paukf,k,aug} = (\mu_{paukf,k,aug}, \mu_{paukf,k,aug} + \sqrt{(\lambda + n_{x,aug})P_{paukf,k,aug}}, \mu_{paukf,k,aug} - \sqrt{(\lambda + n_{x,aug})P_{paukf,k,aug}}). \tag{16}$$

The weight of each sigma point is calculated based on Equations (17) and (18). The predicted mean and covariance were calculated using Equations (19) and (20). The predicted value is the prior of the Bayesian distribution model. These predicted values should be updated when the measurement data are incoming.

$$w_{paukf,i} = \frac{\lambda}{\lambda + n_{x,aug}}, \text{ when } i = 0, \tag{17}$$

$$w_{paukf,i} = \frac{1}{2(\lambda + n_{x,aug})}, \text{ when } i = 1 \ldots n_{x,aug}, \tag{18}$$

$$\bar{x}_{paukf,k+1|k} = \sum_{i=0}^{n_a} w_{paukf,i} X_{paukf,k+1|k,i}, \tag{19}$$

$$\bar{P}_{k+1|k} = \sum_{i=0}^{2n_a} w_{paukf,i} (X_{paukf,k+1|k,i} - \bar{x}_{paukf,k+1|k}) (X_{paukf,k+1|k,i} - \bar{x}_{paukf,k+1|k})^T. \tag{20}$$

After predicting the new mean and covariance matrix based on the augmented sigma point, the algorithm no longer needs to consider noise as the acceleration noise information is already included in the state. Therefore, the augmented state changes back to a normal state with a size of 5. Until now, the prior probability was calculated based on sigma points. When using a Bayesian filter, measurement prediction can be implemented. Instead of using the original range sensor with noise from the vehicle, $\hat{x}_{PF}$ is used in this step. This means that $\hat{x}_{PF}$ becomes a virtual sensor, which is more precise than the original sensor. Since $\hat{x}_{PF}$ already includes sensor information based on the particle filter, it optimally provides a more precise belief of the state. The measurement vector of the sensor is shown in Equation (21).

$$z = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}. \tag{21}$$

The measurement model is shown in Equations (22) and (23). The particle filter measurement provides x, y, and yaw data. This means that the number of rows in matrix A is 3. Because the augmented state information is included in the state, the state of the UKF recovers to 5. This means that the number of columns in matrix A is 5.

$$Z_{paukf,k+1|k,i} = AX_{paukf,k+1|k,i} + \omega_{paukf,k+1}, \tag{22}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{23}$$

The predicted measurement mean is calculated based on the weight of each measurement's sigma points, as shown in Equation (24).

$$\bar{z}_{paukf,k+1|k} = \sum_{i=1}^{n_x} w_{paukf,i} Z_{paukf,k+1|k,i}. \tag{24}$$

The predicted measurement covariance is calculated using Equation (25). R is the measurement noise covariance, as shown in Equation (26). The covariance is tuned according to the particle filter estimation results.

$$S_{k+1|k} = \sum_{i=0}^{2n_x} w_{paukf,i} (Z_{paukf,k+1|k,i} - \bar{z}_{paukf,k+1|k})(Z_{paukf,k+1|k,i} - \bar{z}_{paukf,k+1|k})^T + R, \tag{25}$$

$$R = \begin{bmatrix} {\sigma_{x_{PF}}}^2 & 0 & 0 \\ 0 & {\sigma_{y_{PF}}}^2 & 0 \\ 0 & 0 & {\sigma_{\theta_{PF}}}^2 \end{bmatrix}. \quad (26)$$

At this time, a measurement value is needed to calculate the posterior probability. The update step is similar to that of the Kalman filter. The only difference is that the UKF needs to calculate the cross-correlation value, according to Equation (27), between the sigma points in the state space and the measurement space.

$$T_{k+1|k} = \sum_{i=0}^{2n_x} w_{paukf,i} (X_{paukf,k+1|k,i} - x_{paukf,k+1|k})(Z_{paukf,k+1|k,i} - z_{paukf,k+1|k})^T. \quad (27)$$

Based on the cross-correlation matrix and the measurement covariance, the Kalman gain is then calculated as

$$K_{k+1|k} = T_{k+1|k} S_{k+1|k}^{-1}. \quad (28)$$

The state is updated using the measurement value $\hat{x}_{PF}$, which is obtained from the particle filter estimation as

$$\hat{x}_{paukf} = x_{k+1|k+1} = x_{k+1|k} + K_{k+1|k}(\hat{x}_{PF} - z_{k+1|k}). \quad (29)$$

The covariance matrix is then updated based on the updated Kalman gain and the measurement covariance matrix as

$$\hat{P}_{paukf} = P_{k+1|k+1} = \bar{P}_{k+1|k} - K_{k+1|k} S_{k+1|k} K_{k+1|k}^T. \quad (30)$$

The terms $x_{k+1|k+1}$ and $P_{k+1|k+1}$ are the final estimation results of the PAUKF, which combines the bicycle model, CTRV motion model, Monte Carlo-based estimation, and unscented Kalman filter-based estimation. The complete pseudo code of the PAUKF algorithm is shown in Table 2.

Table 2. Pseudocode of the particle-aided unscented Kalman filter (PAUKF).

Order	Process
1	Start one sample time iteration
2	Initialization $X_{1,2...N}$ particles
3	For 1 to N do
4	$\bar{X}_{k+1} = \text{prediction model}(X_k, u_t)$
5	End for
6	$\hat{x}_{PF} = f(\bar{X}_{k+1}, w_{[1,2...N]})$
7	For 1 to n_{aug} do
8	X_{k+1} = unscented transform (λ, x_k, σ)
9	$\bar{x}_{paukf,k+1|k}$ = CTRV model(based state prediction
10	$\bar{z}_{paukf,k+1|k} = A(\bar{x}_{paukf,k+1|k})$ for measurement prediction
11	$\hat{x}_{paukf}, \hat{P}_{paukf} = \text{state update}(T_{k+1|k}, S_{k+1|k}, \bar{z}_{paukf,k+1|k}, \bar{x}_{paukf,k+1|k}, \hat{x}_{PF}, R)$
12	End one sample time iteration

3. Simulation Environment

The simulation of the PAUKF algorithm was performed using MATLAB. The autonomous driving toolbox is used for constructing the infrastructure, road structure, and vehicle kinematic model. The update frequency of the GPS is 10 Hz, and the frequency of the range sensor and gyroscope is 100 Hz. The noise of the GPS and the range sensor is simulated using the ground truth data appended with Gaussian noise and non-Gaussian noise. Gaussian noise is generated by using the normrnd function in MATLAB, and non-Gaussian noise is generated using a sinusoidal function (we assume the noise affected by the sinusoidal function) and a random number generator, as shown in Table 3 [45,46]. The seed of the random number is set as 50, and the sample time is set as 0.01 s. The variances fed into the PAUKF should be carefully tuned when applied to specific

cases. It should be noted that even if the random seed is the same, the random number is generated depending on the number of times that it has been called. This means that any change in parameter changes the result because the input value changes.

Table 3. Simulated noisy environment setting.

Noise Name	Generate Method
GPS x error (Gaussian)	$\sim N(9.65, 12.2)$ [46]
GPS x error (Non_Gaussian)	$\sim 15\sin(N(0,1)) + N(9.65, 12.2) + 5)$
GPS y error (Gaussian)	$\sim N(8.34, 12.33)$ [46]
GPS y error (Non_Gaussian)	$\sim 15\sin(N(0,1)) + N(8.34, 12.33) + 5)$
Velocity error	$\sim \sin(N(0,1))$
Yaw error	$\sim \sin(N(0,10))$
Yaw rate error	$\sim \sin(N(0,10))$
Random seed	50

The road simulated with three geometries is shown in Figure 3. There are S-shaped roads and a straight road in the X direction. An S-shaped road is used to verify the performance of each filter on a curved road. The straight-line road in the X direction is used in order to verify the performance of each filter on a straight road. There are 12 infrastructures around the road, and their positions are fixed, even when the map changes. In order to prevent the position of the infrastructures from affecting the performance of the filters, all the infrastructures are symmetrical. The velocity is set to a constant value, and the values are 60, 80, 100, and 120 km/h.

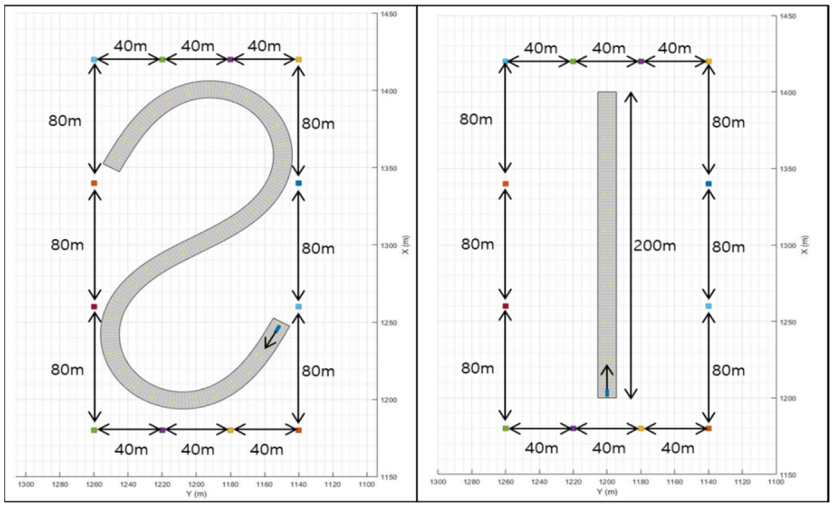

Figure 3. Simulation model.

4. Analysis of Simulation Results

The simulation results are compared to evaluate the performance of the PAUKF. The evaluation parameter is based on the Root Mean Square Error (RMSE) as Equation (31) shows. We choose RMSE as an assessment parameter because the estimation performance of the filter can be compared intuitively by the numerical value of RMSE alone. In Equation (31), N indicates the number of data points. The trajectory of the estimated results and the ground truth of the vehicle's trajectory are compared to verify the algorithm. The effect of the yaw angle is considered for both the x and y directions; therefore, there is no additional comparison of the yaw angle. The unit for all position parameters is meters.

$$\begin{bmatrix} \mathrm{RMSE}_{\mathrm{est}} \\ \mathrm{RMSE}_{\mathrm{noise}} \end{bmatrix} = \begin{bmatrix} \sqrt{[\sum_{i=1}^{N}(\mathrm{Position}_{\mathrm{est}_i} - \mathrm{Position}_{\mathrm{mean\_est}_i})^2]/N} \\ \sqrt{[\sum_{i=1}^{N}(\mathrm{Position}_{\mathrm{noise}_i} - \mathrm{Position}_{\mathrm{mean\_noise}_i})^2]/N} \end{bmatrix}. \quad (31)$$

4.1. Filter Performance on the S-Shaped Road

Figure 4 shows the trajectory results of the PF, UKF, and PAUKF, and noise in the S-shaped road. As the legend shows, the green line with a green circle is the ground truth trajectory, the dashed line with a red upward-pointing triangle is the noisy vehicle trajectory, the black dashed line with a black square is the PF estimated trajectory, the blue dashed line with a blue square is the UKF estimated trajectory, and the yellow dashed line with the yellow star marker is the PAUKF estimated trajectory. The data in Figure 4 are generated when the vehicle velocity is 60 km/h, and the noise is Gaussian, as shown in Table 3. The PF estimated trajectory is near the ground truth trajectory. However, the PF-estimated trajectory is not smooth, and the error is still large. This is because the PF localizes the vehicle position with noisy relative distance to each infrastructure and noisy vehicle data. Since there is no other measurement, it must be considered that the measurement is correct. Compared to that with the PF, the UKF-estimated trajectory is relatively smooth; however, it cannot filter the noise of the GPS data. Because the GPS measurement of the UKF has high variance and the UKF does not use range sensor data, the UKF believes the vehicle model more than the measurement. The noisy measurement also makes the UKF less sensitive to the changes in the position and yaw. Compared to that with the PF and UKF, the trajectory estimated by the PAUKF is more accurate and smoother. As it combines the smoothness of the UKF and the accuracy of the PF, the PAUKF reacts more quickly and precisely when the position and yaw change. Moreover, the PAUKF does not depend completely on either of the filters, trades off the filters, and generates even better results.

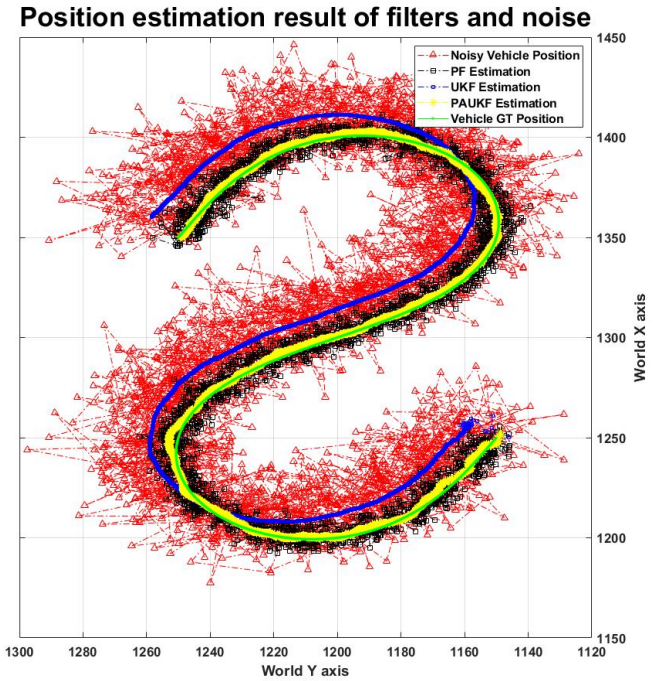

Figure 4. Position estimation result of the filters in the S curve road.

The filter performance results are shown in Table 4. Since the UKF does not use range sensor information, it is not appropriate to compare it with the PF and PAUF. Thus, there are no RMSEs for the UKF in Table 4. To compare with other literature, we calculate the mean value of estimation. The mean of estimation error for the PAUKF is 1.08 m and the variance is 0.7147 m, which is more

precise than the mean of 1.69 m and variance of 1.63 m obtained by GANGNAM for similar noise [47]. In order to determine the performance of the filters in an extreme environment, the algorithm is tested under different velocity and noise environments. As mentioned in Section 3, even if the random seed is the same, the random number still changes depending on the number of times it has been called. Therefore, we analyzed the trend of every filter. It can be observed that the PF and PAUKF estimation errors increase slightly when the velocity increases. However, if we consider the magnitude of the RMSE of the changes in noise from 21.336 to 21.712 m, it can be found that the RMSE of the estimation error does not change even when the velocity increases from 60 to 120 km/h. Compared to the Gaussian noise, the non-Gaussian noise generated a larger mean value. Even so, the precision of the PF does not change even when the noise increases, and the precision is almost the same as the RMSE range of 5.489–5.959 m. The PAUKF has an RMSE range of 1.440–1.772 m, even when the noise increases and velocity increases. This is because PAUKF takes the PF estimation results as input and trades off the measurement and predicted value from the UKF. The trade-off is done using the cross-correlation function in Equation (27). Therefore, the PAUKF combines the recursiveness of the UKF and the location information of the infrastructures based on the PF. The PAUKF improves the accuracy by 4.028–4.049 m compared to the PF.

Table 4. Total Root Mean Square Error (RMSE) of filters in different conditions (unit: m).

	With Gaussian Noise			**With Non-Gaussian Noise**		
Velocity	**Noise**	**PF**	**PAUKF**	**Noise**	**PF**	**PAUKF**
60 km/h	21.336	5.634	1.451	29.796	5.959	1.655
80 km/h	21.310	5.600	1.651	29.730	5.579	1.440
100 km/h	21.154	5.720	1.501	29.430	5.631	1.616
120 km/h	21.712	5.800	1.772	29.934	5.489	1.454

4.2. Filter Performance on a Straight Road in the X Direction

Figure 5 shows the trajectory results of the PF, UKF, and PAUKF, and the noise on a straight road in the X direction. The data in Figure 5 are generated when the vehicle velocity is 60 km/h, and the noise is Gaussian. The algorithm for a straight line is used to determine the performance when the vehicle only moves in the X direction. From Figure 5, it can be seen that the PAUKF converges to the ground truth better than the PF and UKF. Because the vehicle only moves in the X direction, there is no information about the movement in the Y direction. Therefore, even though the PAUKF estimation is better than that of the UKF, in order to improve the response time, the PAUKF tends to believe more about noise in the Y direction. As a result, the PAUKF is not sufficiently precise in the Y direction, as Figure 5 shows.

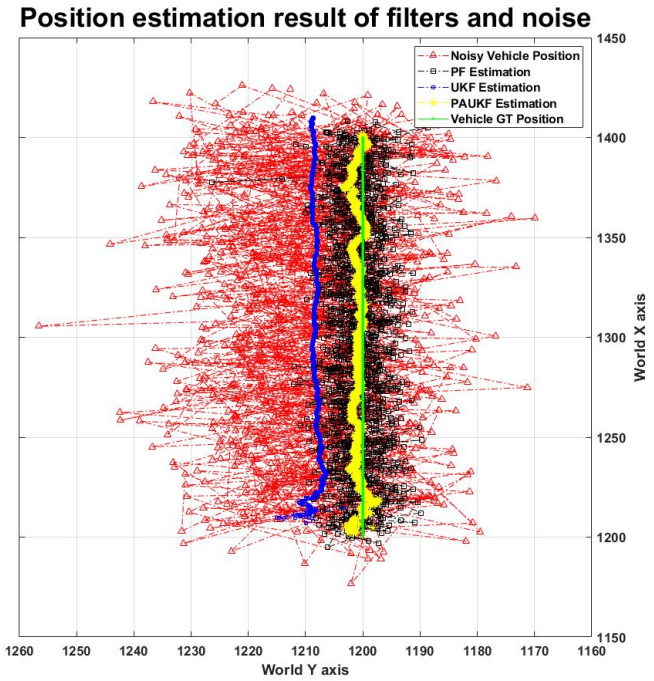

Figure 5. Position estimation result of the filters.

The results for filter performance are shown in Table 5 when the vehicle moves in the X direction. It can be found that the PF and PAUKF estimation properties are the same as those of the vehicle when it runs on an S-shaped road. The estimation results show that the performance of the algorithm does not change even when the map changes. The RMSE of the PF is 5.384–5.692 m and the PAUKF has an RMSE of 1.312–1.800 m even when the noise increases and the velocity increases. The PAUKF improves the accuracy by 3.892–4.072 m compared to the PF.

Table 5. RMSE of filters in different conditions (unit: m).

	With Gaussian Noise			With Non-Gaussian Noise		
Velocity	**Noise**	**PF**	**PAUKF**	**Noise**	**PF**	**PAUKF**
60 km/h	21.411	5.655	1.486	29.376	5.658	1.659
80 km/h	21.518	5.546	1.312	28.987	5.491	1.526
100 km/h	21.848	5.692	1.800	29.274	5.615	1.482
120 km/h	21.363	5.383	1.710	30.078	5.617	1.719

5. Conclusions

In this work, we propose a novel approach for a vehicle estimation algorithm, called the PAUKF, which combines the advantages of the PF and the UKF. The PAUKF combines the unscented transform property of a UKF with a sample-based PF to handle the localization problem in a bad GPS environment by using the range sensor and ground truth data of the infrastructure in an HD map. Owing to properties of the UKF, the PAUKF becomes more robust and precise compared to the original PF, given the same quantity of particles. The performance of the algorithm is stable and accurate (minimum RMSE: 1.44 m) when vehicles move along an S curve or any straight road at speeds of 60 to 120 km/h. The results of the simulation showed that the PAUKF has a significantly higher precision and stability than the PF and in previous research. In future work, we will try to implement the PAUKF in a real vehicle and incorporate the 3D range sensor data to upgrade the algorithm in the real world.

Author Contributions: Conceptualization, M.L. and J.Y.; Software, M.L.; Methodology, M.L. and J.Y.; Validation, M.L.; Writing—original draft preparation M.L.; B.K. Writing—review & editing. All authors have read and agreed to the published version of the manuscript

Funding: This work was supported by KoreaHydro & Nuclear Power company through the project "Nuclear Innovation Center for Haeoleum Alliance" and Technology Innovation Program 2020 funded By the Ministry of Trade, industry & Energy(MI, Korea).

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

Nomenclature

$bel(x)$	Belief of state
$x_{[1,2,3,...i]}$	States of particle1, Particle 2, … Particle i
$w_{[1,2,3,...i]}$	Weights of particle1, particle 2, … particle i
$\overline{X}_{k+1}$	State at sample time k + 1
K	Timestamp
x_k, y_k	Vehicle position in the x, y dimension at time k
v_k	Vehicles position in the x dimension at time k
θ_k	Yaw angle at time k
$\theta_k(\dot{dt})$, $\dot{\theta}$	Yaw rate of vehicle at time k
Δt, dt	Sample time
Z_{k+1}	Measurement vector at time k + 1
$d_{[i]}$	Distance of ego vehicle to ith beacon
$\Delta\theta_{[i]}$	Relative angle of vehicle orientation and ith beacon
$x_{b,i}$, $y_{b,i}$	Relative distance of vehicle and ith beacon
ϵ_d	Noise distance measurement
$\epsilon_{\Delta\theta}$	Noise of angle measurement
$p(x_i, y_i)$	Multivariable normal distribution
σ_x, σ_y	Covariance of sensor range noise in the x- and y-directions
$x_{paukf,k,aug}$	State of PAUKF
w_{velacc}	Noise of vehicle acceleration
w_{yawacc}	Noise of vehicle yaw acceleration
σ_{velacc}	Variance of noise of vehicle acceleration
σ_{velacc}	Variance of noise in vehicle yaw acceleration
$P_{k,aug}$	Variance matrix of PAUKF.

$X_{paukf,k+1,aug}$	Augmented state with sigma points of PAUKF at time k + 1
$\mu_{paukf,k,aug}$	Mean value of augmented state of PAUKF at time k
$n_{x,aug}$	Number of augmented states
$w_{paukf,i}$	Weight of ith sigma point
λ	Sigma point design parameter
$\bar{x}_{paukf,k+1\|k}$	Predicted state based on the weight of sigma points and states
$\bar{P}_{k+1\|k}$	Predicted variance based on sigma points and predicted state mean
$\omega_{paukf,k+1}$	Measurement noise of PAUKF.
$Z_{paukf,k+1\|k,i}$	Measurement prediction based on sigma points.
$X_{paukf,k+1\|k,i}$	Sigma points of state
A	Measurement transition model.
$z_{paukf,k+1\|k}$	Predicted measurement based on sigma points and weights
$S_{k+1\|k}$	Predicted measurement covariance matrix.
R	Variance matrix of the measurement noise.
$\sigma_{x_{pf}}$	Covariance of PF estimation in the x dimension
$\sigma_{y_{pf}}$	Covariance of PF estimation in the y-dimension
$T_{k+1\|k}$	Cross-correlation matrix of PAUKF
$K_{k+1\|k}$	Kalman gain of PAUKF
$\hat{x}_{PAUFK}$	Final state estimation of PAUKF.
$\hat{P}_{PAUFK}$	Final state variance matrix of PAUKF

References

1. Levinson, J.; Montemerlo, M.; Thrun, S. Map-based precision vehicle localization in urban environments. In Proceedings of the Robotics: Science and Systems, Cambridge, MA, USA, 27–30 June 2007; Volume 4, p. 1.
2. Samyeul, N.; An, K.; Wooyong, H. High-Level Data Fusion based Probabilistic Situation Assessment for Highly Automated Driving. In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Las Palmas, Spain, 15–18 September 2015; pp. 1587–1594.
3. Motilal, A.; Kurt, K. Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS. In Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), Hong Kong, China, 20 August 2006; pp. 1063–1068.
4. Mattern, N.; Wanielik, G. Vehicle localization in urban environments using feature maps and aerial images. In Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), Washington, DC, USA, 2011; pp. 1027–1032.
5. Li, C.; Dai, B.; Wu, T. Vision-based precision vehicle localization in urban environments. In Proceedings of the 2013 Chinese Automation Congress, Changsha, China, 7–8 November 2013; pp. 599–604.

6. Parra, I.; Sotelo, M.; Llorca, D.; Ocaña, M.; Robust visual odometry for vehicle localization in urban environments. *Robotica* **2010**, *28*, 441–452.
7. KAMIJO, S.; Gu, Y.; Hsu, L.; Autonomous vehicle technologies: Localization and mapping. *IEICE Fundam.* **2015**, *9*, 131–141.
8. Vivacqua, R.; Vassallo, R.; Martins, F.; A Low Cost Sensors Approach for Accurate Vehicle Localization and Autonomous Driving Application. *Sensors*. **2017,** 17, 2359.
9. Yu, Y.; Li, J.; Guan. H.; Jia. F.; Wang. C. Three-dimensional object matching in mobile laser scanning point clouds. *IEEE Geosci. Remote Sens. Lett.* **2014**, *12*, 492–496.
10. Hata, A.Y.; Wolf. D.F. Feature detection for vehicle localization in urban environments using a multilayer LIDAR. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 420–429.
11. Levinson, J.; Thrun, S. Robust vehicle localization in urban environments using probabilistic maps. In Proceedings of the IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010.
12. Castorena, J.; Agarwal, S. Ground-edge-based LIDAR localization without a reflectivity calibration for autonomous driving. *IEEE Robot. Autom. Lett.* **2017**, *3*, 344–351.
13. Kim, H.; Liu, B.; Goh, C.Y.; Lee, S.; Myung, H. Robust vehicle localization using entropy-weighted particle filter-based data fusion of vertical and road intensity information for a large scale urban area. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1518–1524.
14. Wolcott, R.W.; Eustice, R.M. Robust LIDAR localization using multiresolution Gaussian mixture maps for autonomous driving. *Int. J. Robot. Res.* **2017**, *36*, 292–319.
15. Wolcott, R.W.; Eustice, R.M. Visual localization within lidar maps for automated urban driving. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 176–183.
16. Sampo, K.; Saber, F.; Konstantinos, K.; Mehrdad, D.; Francis, M.; Alexandros, M. 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.
17. Fujii, S.; Fujita, A.; Umedu, T.; Kaneda, S.; Yamaguchi, H.; Higashino, T.; Takai, M. Cooperative vehicle positioning via V2V communications and onboard sensors. In Proceedings of the 2011 IEEE Vehicular Technology Conference (VTC Fall), San Francisco, CA, USA, 5–8 September 2011; pp. 1–5.
18. Rohani, M.; Gingras, D.; Vigneron, V.; Gruyer, D. A new decentralized Bayesian approach for cooperative vehicle localization based on fusion of GPS and VANET based inter-vehicle distance measurement. *IEEE Intell. Transp. Syst. Mag.* **2015**, *7*, 85–95.
19. Mattern, N.; Obst, M.; Schubert, R.; Wanielik, G.; Co-operative vehicle localization algorithm—Evaluation of the CoVeL approach. In Proceedings of the International Multi-Conference on Systems, Signals & Devices, Chemnitz, Germany, 20–23 March 2012; pp. 1–5.
20. Ordóñez-Hurtado, R.H.; Griggs, W.M.; Crisostomi, E.; Shorten, R.N. Cooperative positioning in vehicular ad-hoc networks supported by stationary vehicles. Avoilable online: https://www.hamilton.ie/smarttransport/publications/arXivCooperativePositioningFeb2015.pdf.
21. Golestan, K.; Seifzadeh, S.; Kamel, M.; Karray, F.; Sattar, F. Vehicle localization in VANETs using data fusion and V2V communication. In Proceedings of the second ACM international symposium on Design and analysis of intelligent vehicular networks and applications, Paphos, Cyprus, 21–22, October 2012; pp. 123–130.
22. Lina, A.; Imad, M.; Monika, R. Weighted localization in Vehicular Ad Hoc Networks using vehicle-to-vehicle communication. In Proceedings of the 2014 Global Information Infrastructure and Networking Symposium (GIIS), Montreal, QC, Canada, 15–19 September 2014; pp. 1–5.
23. Altoaimy, L.; Mahgoub, I. Fuzzy logic based localization for vehicular ad hoc networks. In Proceedings of the 2014 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS), Orlando, FL, USA, 9–12 December 2014; pp. 121–128.
24. Kalman, R.E. A new approach to linear filtering and prediction problems. *J. Basic Eng.* **1960**, *82*, 35–45.
25. Andrew. H.J. *Stochastic Processes and Filtering Theory*; Academic Press: San Diego, CA, USA, 1970.
26. Hamzah, A.; Toru. N. Extended Kalman filter-based mobile robot localization with intermittent measurements. *Syst. Sci. Control Eng. An Open Access J.* **2013**, *1*, 113–126.
27. Xu, Y.; Shmaliy, Y.S.; Ahn, C.K.; Tian, G.; Chen, X. Robust and accurate UWB-based indoor robot localisation using integrated EKF/EFIR filtering. *IET Radar Sonar Navig.* **2018**, *12*, 750–756.

28. Luo, Y.H.; Jiang, P.; Hu, W.W.; Xie, Y.Q. Application of EKF in Laser/Inertial Sensors Localization System. In Proceedings of the 2010 International Conference on Electrical and Control Engineering, Wuhan, China, 25–27 June 2010; pp. 3369–3372.
29. Jaewoo, Y.; Byeongwoo, K. Vehicle position estimation using nonlinear tire model for autonomous vehicle. *J. Mech. Sci. Technol.* **2016**, *30*, 3461–3468.
30. Zhao, S.; Gu, J.; Ou, Y.; Zhang, W.; Pu, J.; Peng, H. IRobot self-localization using EKF. In Proceedings of the 2016 IEEE International Conference on Information and Automation (ICIA), Ningbo, China, 1–3 August 2016; pp. 801–806.
31. Xiao, Y.; Ou, Y.; Feng, W. Localization of indoor robot based on particle filter with EKF proposal distribution. In Proceedings of the 2017 IEEE international conference on cybernetics and intelligent systems (CIS) and IEEE conference on robotics, automation and mechatronics (RAM), Ningbo, China, 19–21 November 2017; pp. 568–571.
32. Julier, S.J.; Uhlmann, J.K. Unscented filtering and nonlinear estimation. *Proc. IEEE* **2004**, *92*, 401–422.
33. Wan, E.A.; Van Der Merwe, R. The unscented Kalman filter for nonlinear estimation. In Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium, Lake Louise, AB, Canada, 4 October 2000; pp. 153–158.
34. Julier, S.J.; Uhlmann, J.K. New extension of the Kalman filter to nonlinear systems. In Proceedings of the Signal processing, sensor fusion, and target recognition VI, Orlando, FL, USA, 28 July 1997; Volume 3068, pp. 182–193.
35. Doucet, A.; Johansen, A.M. A tutorial on particle filtering and smoothing: Fifteen years later. *Handb. Nonlinear Filter.* **2009**, *12*, 3.
36. Thrun, S. Particle filters in robotics. In Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence, Edmonton, AB, Canada, 1–4, August 2002; pp. 511–518.
37. Dan, C. Particle filters- A theoretical perspective. *Sequential Monte Carlo Methods in Practice*, 1st ed.; Doucet, A., De Freitas, N., Gordon, N., Eds.; Springer: New York, NY, USA, 2001; pp. 17 – 38.
38. Aggarwal, P.; Syed, Z.; El-Sheimy, N. Hybrid extended particle filter (HEPF) for integrated inertial navigation and global positioning systems. *Meas. Sci. Technol.* **2009**, *20*, 055203–055212.
39. Aggarwal, P.; Gu, D.; Nassar, S.; Syed, Z.; El-Sheimy, N. Extended particle filter (EPF) for INS/GPS land vehicle navigation applications. In Proceedings of the Institute of Navigation Satellite Division Technical Meeting (ION GNSS 2007), Fort Worth, TX, USA, 25 – 28, September 2007; pp. 2619–2626.
40. Wan, Y.; Wang, S.; Qin, X. IMM Iterated ExtendedH∞ Particle Filter Algorithm. *Hindawi Publ. Corp. Math. Probl. Eng.* **2013**, *2013*, 8.
41. Van, D.M.R.; Doucet, A.; De, F.N.; Wan, E.A. The unscented particle filter. In Proceedings of the Advances in neural information processing systems, Denver, CO, USA, 27- 30, November 2000; pp. 584–590.
42. Yu, W.; Peng, J.; Zhang, X.; Li, S.; Liu, W. An adaptive unscented particle filter algorithm through relative entropy for mobile robot self-localization. *Math. Probl. Eng.* **2013**, doi:10.1155/2013/567373.
43. Pak, J.M.; Ahn, C.K.; Shmaliy, Y.S.; Shi, P.; Lim, M.T. Accurate and reliable human localization using composite particle/FIR filtering. *IEEE Trans. Hum. Mach. Syst.* **2016**, *47*, 332–342.
44. Rajamani, R. *Vehicle Dynamics and Control*; Springer Science & Business Media: Boston, MA, USA, 2011; pp. 20–26.
45. Sebastian, T.; Wolfram, B.; Dieter, F. *Probabilistic Robotics*; MIT Press: Cambridge, MA, USA, pp. 136 – 164.
46. Sun, L.; Shen, M.; Xu, W.; Li, Z.; Beadle, P. Model for Generating Non-gaussian Noise Sequences Having Specified Probability Distribution and Spectrum. In *Parallel and Distributed Computing: Applications and Technologies*; Liew, K.M., Shen, H., See, S., Cai, W., Fan, P., Horiguchi, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 795–798.
47. 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.* **2016**, *18*, 1078–1086.