

Kalman-filter-based sensor fusion applied to road-objects detection and tracking for autonomous vehicles

Proc IMechE Part I:
J Systems and Control Engineering
1–14

© IMechE 2020

Article reuse guidelines:

sagepub.com/journals-permissions

DOI: 10.1177/0959651820975523

journals.sagepub.com/home/pii



Wael Farag^{1,2}

Abstract

In this article, a real-time road-Object Detection and Tracking (LR_ODT) method for autonomous driving is proposed. This method is based on the fusion of lidar and radar measurement data, where they are installed on the ego car, and a customized Unscented Kalman Filter is employed for their data fusion. The merits of both devices are combined using the proposed fusion approach to precisely provide both pose and velocity information for objects moving in roads around the ego car. Unlike other detection and tracking approaches, the balanced treatment of both pose estimation accuracy and its real-time performance is the main contribution in this work. The proposed technique is implemented using the high-performance language C++ and utilizes highly optimized math and optimization libraries for best real-time performance. Simulation studies have been carried out to evaluate the performance of the LR_ODT for tracking bicycles, cars, and pedestrians. Moreover, the performance of the Unscented Kalman Filter fusion is compared to that of the Extended Kalman Filter fusion showing its superiority. The Unscented Kalman Filter has outperformed the Extended Kalman Filter on all test cases and all the state variable levels (–24% average Root Mean Squared Error). The employed fusion technique shows how outstanding is the improvement in tracking performance compared to the use of a single device (–29% Root Mean Squared Error with lidar and –38% Root Mean Squared Error with radar).

Keywords

Sensor fusion, Kalman filter, object detection and tracking, advanced driving assistance systems, autonomous driving

Date received: 26 June 2020; accepted: 24 October 2020

Introduction

Improving safety, lowering road accidents, boosting energy efficiency, enhancing comfort, and enriching driving experience are the most important driving forces behind equipping present-day cars with Advanced Driving Assistance Systems (ADAS).^{1–3} Many ADAS functions represent incremental steps toward a hypothetical future of safe fully autonomous cars.^{4–12}

A critical component of the various ADAS features that are also highly required in autonomous cars is the recognition and accurate assessment of the surroundings.^{5–8} This component depends on data observed from sensors mounted on the ego car.⁹ If there is an object close by, it is of interest to know where that object is, what the object's velocity is, and if the object can be described by a plain geometric shape.¹² Lidar and radar are one of the sought-after sensors for exploiting in ADAS and autonomous-car features.¹³ A lidar always returns many concentrated detection points (point-cloud) that describe each detected object.^{14,15} Likewise, a radar

often returns multiple detections per target but not as dense as a lidar.¹⁶ This means that it is necessary to group detections originating from the same target, that is, to cluster the detections, to obtain information about the surroundings.^{16–18}

The ego car equipped with a lidar and radar receives a collection of raw data of sensors measurements that include information of detected road objects. Then, the proposed LR_ODT method employs a two-step approach to find and identify these road objects within the received data. The first step is to coarse cluster the lidar/radar raw data separately to detect objects within

¹College of Engineering and Technology, American University of the Middle East, Al-Eqaila, Kuwait

²Department of Electrical Engineering, Cairo University, Cairo, Egypt

Corresponding author:

Wael Farag, Department of Electrical Engineering, Cairo University, Cairo, Giza 11612, Egypt.

Emails: wael.farag@aum.edu.kw, wael.farag@cu.edu.eg

using the Grid-Based Density-Based Spatial Clustering of Applications with Noise (GB-DBSCAN) algorithm.¹⁹ Accordingly, each object is then represented with its core point (centroid). The second step is the estimation of the object's cluster velocity as well as determining its corresponding geometrical shape, which is performed using the RANdom SAmple Consensus (RANSAC) iterative algorithm.²⁰

As mentioned before, it is mandatory to have continuous, precise, and accurate velocity and position information about the objects surrounding the ego car. In this article, this is accomplished by combining data from lidar and radar sensors.

Recent lidars have a large range (up to 200 m) and a wide field of view, and can thus track objects even at big distances (necessary at high speeds) as well as in curves (i.e. very accurate in position measurement).¹³ Their main drawback is that they completely lack dynamic information about the detected objects (velocity measurement). Radar sensors, on the other hand, have a relatively narrow field of view and reduced angular resolution (less accurate in position measurement), but use the Doppler effect to directly provide velocity information. The fusion of the data from both sensors can thus benefit from the combination of their merits.²¹

Accordingly, sensor fusion of lidar and radar that combines the strengths of both sensor types is a logical step. This step has been investigated earlier in the literature, with promising prospects in the automotive industry.²²

As an early endeavor, Gohring et al.²¹ apply lidar-radar fusion for the application of car following on highways based on the Kalman Filter (KF).²³ To test the performance of their fusion technique, the authors have formed the ground truth by computing the mean square errors of relative distances and velocities in a highway-tracking scenario using a least-squares polynomial approximation of sensors data.

Moreover, to improve the perceived model of the environment, Chavez-Garcia and Aycard²⁴ include the objects' classification from multiple sensors (lidar, radar, and camera) detections as a key component of the object's representation and the perception process that is based on a framework derived from evidence theory. The fusion approach is tested using real data from different driving scenarios and focusing on four objects of interest: pedestrian, bike, car, and truck.

For tracking multiple objects, Rangesh and Trivedi²⁵ propose a modular framework capable of accepting object proposals from different sensor modalities (cameras and lidars) and fuse them. The approach is tested on real-world highway data, showing its effectiveness to track objects through entire maneuvers around the ego-vehicle.

For obstacle detection, Hajri and Rahal²⁶ employ the global nearest neighbor (GNN) standard filter on the fused lidar/radar sensors data for associating new measurements with the underlying observed objects. The benefits of data fusion comparing with the use of a

single sensor are illustrated through several tracking scenarios (on a highway and a bend).

The emphasis of this article is on the data fusion between lidar and radar for road-objects' tracking. The proposed technique is the acronym: a Lidar/Radar-based road-Object Detection and Tracking technique (LR_ODT). According to the presented state of the art, two different general fusion methods can be distinguished and both were applied for lidar and radar: KF and evidence theory. However, approaches that apply any of these methods agree on the main steps. Meanwhile, the technique that has been adopted in this work is based on using KFs. Despite the previously mentioned works,²⁷ the literature still clearly lacks the investigation of employing the UKF^{28,29} for lidar/radar fusion while applied for road-object tracking for autonomous driving. Therefore, this article will mainly focus on the tailoring and implementation of UKF for tracking various road objects. The UKF design will be validated by tracking three road-objects: car, bicycle, and pedestrian. A quantitative comparison between the performance of the UKF versus that of the EKF,^{30,31} as well as a quantitative comparison between the performance of the LR_ODT with and without the employment of lidar/radar fusion, are carried out.

The contribution of this article can be enumerated as follows:

1. Tailoring the UKF as well as the EKF algorithms to fuse multiple radars and lidars data to achieve more accurate pose data for moving objects around the ego car, proving that the UKF-based method has better performance but the EKF-based method is less computationally demanding.
2. Employing a high-order-generic-object-motion model (five state variables that suits the most common road-objects: car, bicycle, and pedestrian) in the development of the UKF and EKF to generate more accurate estimates and improve the overall performance.
3. Carrying out a quantitative comparative study between the sensor fusion performance using EKF and UKF using the same use cases.
4. Evaluating the gain of fusion by testing the UKF on three different cases (lidar + radar, lidar only, and radar only).
5. Evaluating the real-time performance of both the EKF and UKF on a moderate computational platform.
6. Employing the GB-DBSCAN clustering algorithm to detect potential objects from the lidar/radar raw data, and finding their centroids.
7. Employing the RANSAC algorithm and object proposals for bicycle, car, and pedestrian for estimation of the detected object's cluster velocity as well as determining its corresponding geometrical shape.

This article is organized as follows: section "KF overview" gives an overview of the generic KF, section "The moving object model" presents the moving object

generic model that is used in the design of both the EKF and the UKF, section “Sensor fusion using EKF” details the sensor fusion using the EKF, section “Sensor fusion using UKF” details as well the sensor fusion using the UKF, section “Implementation of Kalman filters” is dedicated to describe the implementation issues of both the EKF and the UKF, section “Testing and evaluation results” presents the testing and evaluation results, and section “Conclusion” concludes the work done in this article and enumerates suggested improvements for future work.

KF overview

The KF²³ is a system of equations working together to form a predictor-update cyclic optimal estimator that minimizes the estimated error covariance. The KF estimates the state $x \in R^n$ given the measurement $z \in R^m$ of a discrete-time controlled process that is modeled by the following set of linear stochastic difference equations

$$\begin{aligned} x_k &= Fx_{k-1} + Bu_k + \nu_k \\ z_k &= Hx_k + \omega_k \\ \nu_k &\sim \mathcal{N}(0, Q_k) \\ \omega_k &\sim \mathcal{N}(0, R_k) \end{aligned} \quad (1)$$

where F is the process state transition model, B is the control-input model, u_k is the control input, H is the measurement model, ν_k is the process white noise which is the Gaussian distribution (\mathcal{N}) with zero mean and covariance matrix Q_k , and ω_k is the measurement white noise which is the Gaussian distribution (\mathcal{N}) with zero mean and covariance matrix R_k .

The KF estimation process works in two steps:

1. *The predication step* estimates the next state as follows

$$\begin{aligned} \hat{x}_k &= Fx_{k-1} + Bu_k \\ \hat{P}_k &= FP_{k-1}F^T + Q_k \end{aligned} \quad (2)$$

2. *The measurement update step* works as follows

$$\begin{aligned} \hat{y}_k &= z_k - H\hat{x}_k \\ S_k &= H\hat{P}_kH^T + R_k \\ K_k &= \hat{P}_kH^TS_k^{-1} \\ x_k &= \hat{x}_k + K_k\hat{y}_k \\ P_k &= (I - KH)\hat{P}_k \end{aligned} \quad (3)$$

where P_k is the KF process estimate covariance, K_k is the KF gain, and S_k is the measurement covariance matrix.

However, the above equations are only limited to linear processes, and accordingly, it is not suitable to the radar measurement process which is inherently nonlinear. Therefore, the extended KF³⁰ is introduced. The EKF estimation process is represented by equation (4) instead of equation (1) as follows

$$\begin{aligned} \hat{x}_k &= f(x_{k-1}, u_k) + \nu_k \\ \hat{z}_k &= h(x_{k-1}) + \omega_k \end{aligned} \quad (4)$$

where $f(\cdot)$ and $h(\cdot)$ are nonlinear functions and can be linearized around an arbitrary operating point μ using the truncated Taylor series expansion as follows

$$\begin{aligned} f(x) &= f(\mu) + \frac{\partial f(\mu)}{\partial x}(x - \mu) \\ h(x) &= h(\mu) + \frac{\partial h(\mu)}{\partial x}(x - \mu) \end{aligned} \quad (5)$$

The derivatives of $f(\cdot)$ and $h(\cdot)$ with respect to x are called Jacobians. The F_j and H_j Jacobians are calculated as in equation (6) while taking the form of matrices of orders $n \times n$ and $m \times n$, respectively. These matrices contain all the partial derivatives with respect to each state variable

$$\begin{aligned} F_j &= \frac{\partial f(\mu)}{\partial x} \\ H_j &= \frac{\partial h(\mu)}{\partial x} \end{aligned} \quad (6)$$

Accordingly, the EKF process is represented as well by the prediction and update equations (2) and (3), respectively, after replacing F with $f(x_{k-1}, u_k)$, H with $h(x_{k-1})$ in equation (2), and F with F_j and H with H_j in equation (3).

Due to the EKF which is using only the first-order derivative in the linearization process and ignoring the higher order terms, errors are accumulated in the state and covariance estimation. The unscented KF is introduced²⁸ to overcome this limitation. The UKF is a derivative-free alternative to EKF that uses a deterministic sampling approach. The UKF utilizes as well the predict-update two-step process; however, they are now augmented with further steps like generation and prediction of sigma points as shown in Figure 1.

In the UKF process, the state Gaussian distribution is represented using a minimal set of carefully chosen sample points, called sigma points. $n_x = 2n+1$ sigma points are selected based on the following rule

$$X_k = \begin{bmatrix} x_k & x_k + \sqrt{(\lambda + n_x)P_k} & x_k - \sqrt{(\lambda + n_x)P_k} \end{bmatrix} \quad (7)$$

where X_k is the sigma-point matrix which includes n_x sigma-point vectors and λ is a design parameter that determines the spread of the generated sigma points and usually takes the form $\lambda = 3 - n_x$.

In the sigma-point prediction step, each generated sigma point is inserted in the UKF nonlinear process model given in equation (8) to produce the matrix of the predicted (estimated) sigma points, which has an $n \times n_x$ dimension

$$\hat{X}_{k+1} = f(X_k, \nu_k) \quad (8)$$

In the next step, the predicted state-mean and covariance matrices are calculated from the predicted sigma points as given in equation (9)

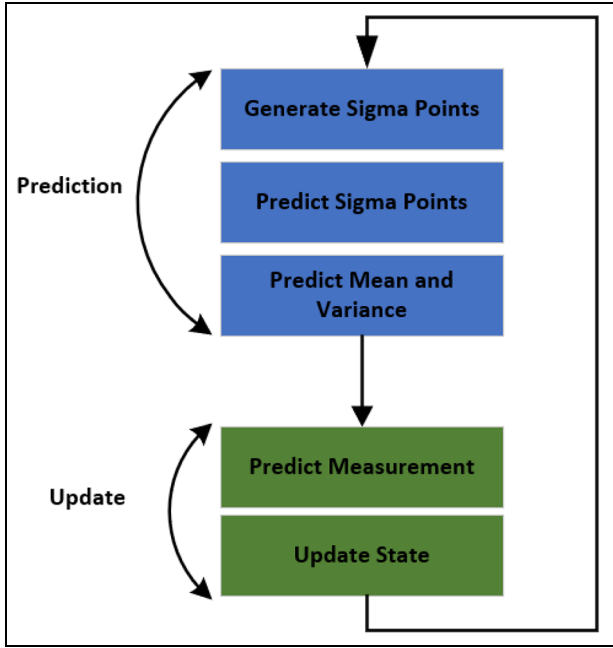


Figure 1. UKF roadmap.

$$\hat{x}_{k+1} = \sum_{i=0}^{n_x} w_i \hat{X}_{k+1,i} \quad (9)$$

$$\hat{P}_{k+1} = \sum_{i=0}^{2n_x} w_i (\hat{X}_{k+1,i} - \hat{x}_{k+1}) (\hat{X}_{k+1,i} - \hat{x}_{k+1})^T$$

where w_i 's are the sigma-point weights that are used here to invert the spreading of the sigma points. These weights are calculated as shown in equation (10)

$$w_i = \frac{\lambda}{\lambda + n_x}, i = 0 \quad (10)$$

$$w_i = \frac{1}{2(\lambda + n_x)}, i = 1, \dots, n_x$$

In the measurement prediction step, each generated sigma point is inserted in the UKF nonlinear measurement model given in equation (11) to produce the matrix of the predicted measurement sigma points, which has an $n \times n_x$ dimension

$$\hat{Z}_{k+1} = h(\hat{X}_{k+1}) \quad (11)$$

In the next step, the predicted measurement-mean-and-covariance matrices are calculated from the predicted sigma points as well as the measurement noise covariance matrix R as given in equation (12)

$$\hat{z}_{k+1} = \sum_{i=0}^{n_x} w_i \hat{Z}_{k+1,i} \quad (12)$$

$$S_{k+1} = \sum_{i=0}^{2n_x} w_i (\hat{Z}_{k+1,i} - \hat{z}_{k+1}) (\hat{Z}_{k+1,i} - \hat{z}_{k+1})^T + R$$

$$R = E\{\omega_k \omega_k^T\}$$

where w_i 's are the sigma-point weights that are determined using equation (10).

The final step is the UKF state update, where the UKF gain matrix (K) is calculated as in equation (13) using the calculated cross-correlation matrix (T) between the sigma points in the state space and the measurement space. The gain is used to update the UKF state vector (x) as well as the state covariance matrix (P)

$$T_{k+1} = \sum_{i=0}^{2n_x} w_i (\hat{X}_{k+1,i} - \hat{x}_{k+1}) (\hat{Z}_{k+1,i} - \hat{z}_{k+1})^T \quad (13)$$

$$K_{k+1} = T_{k+1} S_{k+1}^{-1}$$

$$x_{k+1} = \hat{x}_{k+1} + K_{k+1} (\hat{z}_{k+1} - z_{k+1})$$

$$P_{k+1} = \hat{P}_{k+1} - K_{k+1} S_{k+1} K_{k+1}^T$$

The moving object model

The state of the moving object³² is determined by the five variables grouped into the state vector x shown in equation (14), where p_x and p_y are the object position in the x - and y -axes, respectively, as shown in Figure 2; v is the magnitude of object velocity derived from its x and y components v_x and v_y , respectively. ψ is the yaw angle (object orientation) and $\dot{\psi}$ is rate of change of the object-yaw angle

$$x = \begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \end{bmatrix}, v = \sqrt{v_x^2 + v_y^2}, \psi = \tan^{-1} \frac{v_y}{v_x} \quad (14)$$

The nonlinear $x_{k+1} = f(x_k, v_k)$ difference equation that describes the motion model of the object is derived based on the state vector x and presented in equations (15) and (16)

$$x_{k+1} = x_k + \begin{bmatrix} \frac{v_k}{\psi_k} (\sin(\psi_k + \dot{\psi}_k \Delta t) - \sin(\psi_k)) \\ \frac{v_k}{\psi_k} (-\cos(\psi_k + \dot{\psi}_k \Delta t) + \cos(\psi_k)) \\ 0 \\ \Delta t \\ 0 \end{bmatrix} + v_k \quad (15)$$

$$v_k = \begin{bmatrix} \frac{1}{2} (\Delta t)^2 \cos(\psi_k) \cdot v_{a,k} \\ \frac{1}{2} (\Delta t)^2 \sin(\psi_k) \cdot v_{a,k} \\ \Delta t \cdot v_{a,k} \\ \frac{1}{2} (\Delta t)^2 \cdot v_{\ddot{\psi},k} \\ \Delta t \cdot v_{\ddot{\psi},k} \end{bmatrix} \quad (16)$$

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

$$v_{a,k} \sim \mathcal{N}(0, \sigma_a^2) \quad (17)$$

$$v_{\ddot{\psi},k} \sim \mathcal{N}(0, \sigma_{\ddot{\psi}}^2)$$

where Δt is the time difference between two consecutive samples, $\ddot{\psi}$ is the yaw acceleration, a is the longitudinal acceleration, $v_{a,k}$ is the longitudinal acceleration noise at sample k with a standard deviation σ_a^2 , and $v_{\ddot{\psi},k}$ is

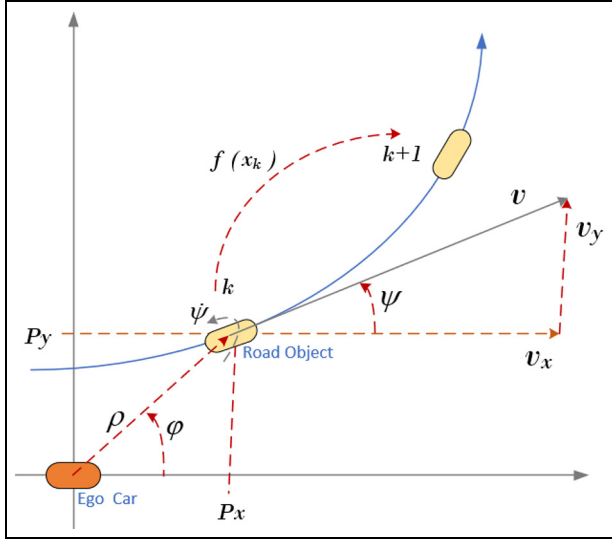


Figure 2. An object motion model.

the yaw acceleration noise at sample k with a standard deviation σ_{ψ}^2 .

If the ψ is zero, and to avoid dividing by zero in equation (15), the following approximation is used to calculate the prediction of p_x , and p_y

$$\begin{aligned} p_{x_{k+1}} &= p_{x_k} + v_k \cos(\psi_k) \Delta t \\ p_{y_{k+1}} &= p_{y_k} + v_k \sin(\psi_k) \Delta t \end{aligned} \quad (18)$$

Sensor fusion using EKF

Figure 3 presents the lidar and radar data fusion technique employing the EKF. The received sensor raw

data (either lidar or radar) are getting processed before being supplied to the EKF.

The processing is performed using clustering and association algorithms. DBSCAN³³ is an unsupervised clustering algorithm that groups together data points if the density of the points is high enough. It requires two parameters to determine the density. The first parameter is ϵ describing the radial distance from a point p that is being evaluated. The second parameter is $minPts$, which is the least number of detections that have to be within a distance ϵ from p , including p itself, to form a cluster. By choosing ϵ and $minPts$, it is then possible to decide the necessary density for a group of points to form a cluster; however, these fixed parameters are not convenient if various types and topologies of road objects need to be detected. As an improvement to this algorithm, GB-DBSCAN is introduced.¹⁹ It works in the same manner as DBSCAN but does not have fixed parameters. Instead, a polar grid is created according to the radial and angular resolution of the sensor. Instead of looking at a circular search area with a fixed radius, GB-DBSCAN is able to use a more dynamic, elliptic, search area.

While GB-DBSCAN is used for coarse clustering, the RANSAC²⁰ is used to fine-tune the clustering and associate geometrical shape proposals to potential coarse clusters. The output of the RANSAC is a fine-tuned object centroid (p_x and p_y) and its type (bicycle, car, or pedestrian).

The processed lidar measurement vector includes the moving object centroid position (p_x and p_y) in Cartesian coordinates, while the radar measurement vector includes the moving object centroid position (ρ , ϕ) and radian velocity ($\dot{\rho}$) in polar coordinates as

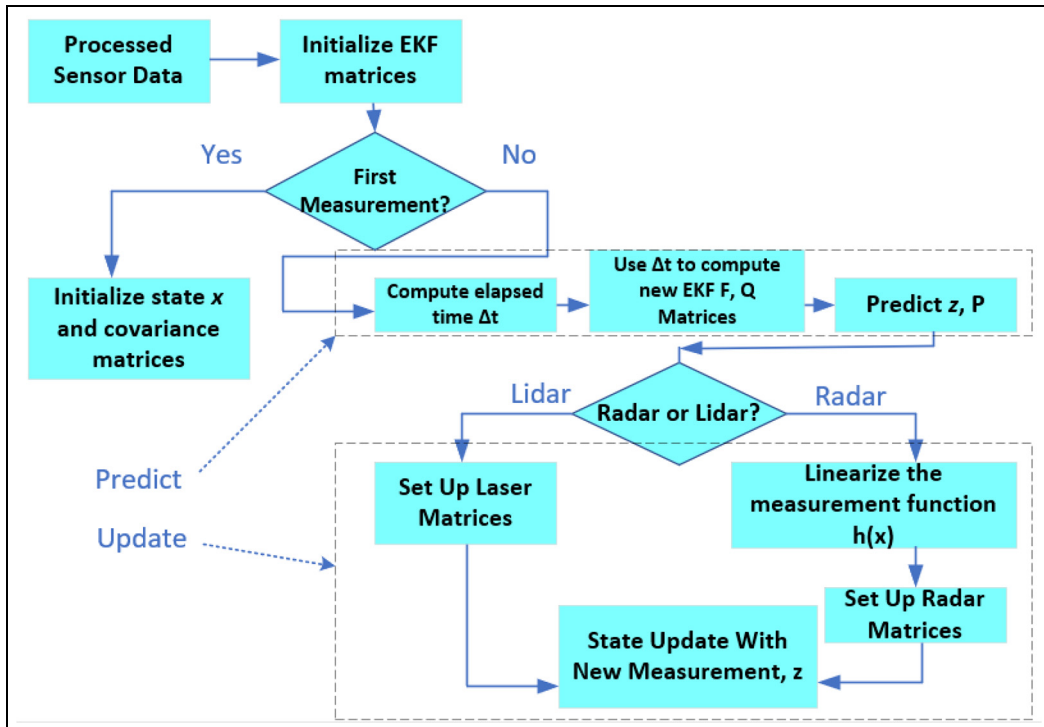


Figure 3. Lidar and radar data fusion using EKF.

represented by equation (19). The mapping function that specifies how lidar Cartesian coordinates got mapped to the radar polar coordinates are given as well in equation (20)

$$z_{lidar} = \begin{pmatrix} p_x \\ p_y \end{pmatrix}, z_{radar} = \begin{pmatrix} \rho \\ \varphi \\ \dot{\rho} \end{pmatrix} \quad (19)$$

$$H_{jradar} = \begin{bmatrix} \frac{p_x}{\sqrt{p_x^2 + p_y^2}} & \frac{p_y}{\sqrt{p_x^2 + p_y^2}} & 0 & 0 & 0 \\ -\frac{p_y}{p_x^2 + p_y^2} & \frac{p_x}{p_x^2 + p_y^2} & 0 & 0 & 0 \\ \frac{p_y v (p_y \cos(\psi) - p_x \sin(\psi))}{(p_x^2 + p_y^2)^{3/2}} & \frac{p_x v (p_x \sin(\psi) - p_y \cos(\psi))}{(p_x^2 + p_y^2)^{3/2}} & \frac{p_x \cos(\psi) + p_y \sin(\psi)}{\sqrt{p_x^2 + p_y^2}} & \frac{v (p_y \cos(\psi) - p_x \sin(\psi))}{\sqrt{p_x^2 + p_y^2}} & 0 \end{bmatrix} \quad (25)$$

$$h(x) = \begin{pmatrix} \rho \\ \varphi \\ \dot{\rho} \end{pmatrix} = \begin{pmatrix} \sqrt{p_x^2 + p_y^2} \\ \arctan\left(\frac{p_y}{p_x}\right) \\ \frac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}} \end{pmatrix} \quad (20)$$

$$p_x = \rho \cos(\varphi), p_y = \rho \sin(\varphi) \quad (21)$$

The EKF state nonlinear model is the moving object model that is described in detail in section “The moving object model” and mathematically represented by equations (15)–(17). The Jacobian of this state model (F_j) is presented in equation (22), while the associated state covariance matrix (Q) is given by equation (23)

$$F_j = \begin{bmatrix} 1 & 0 & \frac{1}{\psi} (-\sin(\psi) + \sin(\Delta t \dot{\psi} + \psi)) & \frac{v}{\psi} (-\cos(\psi) + \cos(\Delta t \dot{\psi} + \psi)) & \frac{v \Delta t}{\psi} \cos(\Delta t \dot{\psi} + \psi) - \frac{v}{\psi} (-\sin(\psi) + \sin(\Delta t \dot{\psi} + \psi)) \\ 0 & 1 & \frac{1}{\psi} (\cos(\psi) - \cos(\Delta t \dot{\psi} + \psi)) & \frac{v}{\psi} (-\sin(\psi) + \sin(\Delta t \dot{\psi} + \psi)) & \frac{v \Delta t}{\psi} \sin(\Delta t \dot{\psi} + \psi) - \frac{v}{\psi} (\cos(\psi) - \cos(\Delta t \dot{\psi} + \psi)) \\ 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (22)$$

$$Q = \begin{bmatrix} \frac{\Delta t^4}{4} \sigma_{a_x}^2 & 0 & \frac{\Delta t^3}{2} \sigma_{a_x}^2 & 0 & 0 \\ 0 & \frac{\Delta t^4}{4} \sigma_{a_y}^2 & \frac{\Delta t^3}{2} \sigma_{a_y}^2 & 0 & 0 \\ \frac{\Delta t^3}{2} \sigma_{a_x}^2 & \frac{\Delta t^3}{2} \sigma_{a_y}^2 & \Delta t^2 \sigma_a^2 & 0 & 0 \\ 0 & 0 & 0 & \Delta t^2 \sigma_{\psi}^2 & 0 \\ 0 & 0 & 0 & 0 & \Delta t^2 \sigma_{\psi}^2 \end{bmatrix} \quad (23)$$

Equation (24) presents the lidar measurement model (H_{lidar}) as well as the measurement noise covariance matrix (R_{lidar}) based on the state vector in equation (14). These matrices are required for the update step of the EKF that is represented by equation (3)

$$H_{lidar} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (24)$$

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

where σ_{p_x} and σ_{p_y} are the noise standard deviations for the object x and y positions, respectively.

In addition, equation (25) shows the derived Jacobian matrix (H_{jradar}) which represents the radar measurement model that will be used to estimate the measurement vector that is given in equation (19). The estimated measurement will be used in the update step of the EKF as illustrated in equation (3). The measurement noise covariance matrix (R_{radar}) is derived and is shown in equation (26). This matrix is required as well for the update step of the EKF

$$R_{radar} = \begin{bmatrix} \sigma_{\rho}^2 & 0 & 0 \\ 0 & \sigma_{\varphi}^2 & 0 \\ 0 & 0 & \sigma_{\dot{\rho}}^2 \end{bmatrix} \quad (26)$$

where σ_{ρ} is the noise standard deviation of the object radial distance, σ_{φ} is the noise standard deviation of the object heading (bearing), and $\sigma_{\dot{\rho}}$ is the noise standard deviation of the object yaw rate.

As per the above presentation, each sensor has its own prediction update scheme; however, both sensors share the same state prediction scheme. The belief about the object's position and velocity is updated asynchronously each time the measurement is received regardless of the source sensor.

Both distinct update schemes are getting updated by any received measurement data (either from lidar or from radar). The state vector (x) is getting updated after receiving a lidar measurement vector (z_{lidar}) in equation (19) by inserting (F_j , Q , H_{lidar} , and R_{lidar}) in equations (2) and (3). Likewise, the state vector (x) is getting updated after receiving a radar measurement vector (z_{radar}) in equation (19) by inserting (F_j , Q , H_{jradar} , and R_{radar}) in equations (2) and (3). Accordingly, the state vector (x) is the product of the fusion between lidar and radar measurement data.

Sensor fusion using UKF

Figure 4 presents the lidar and radar data fusion technique employing the UKF. After computing the elapsed time between consecutive sensor reading (Δt), the sigma points (X_k) are generated using equation (7), and then,

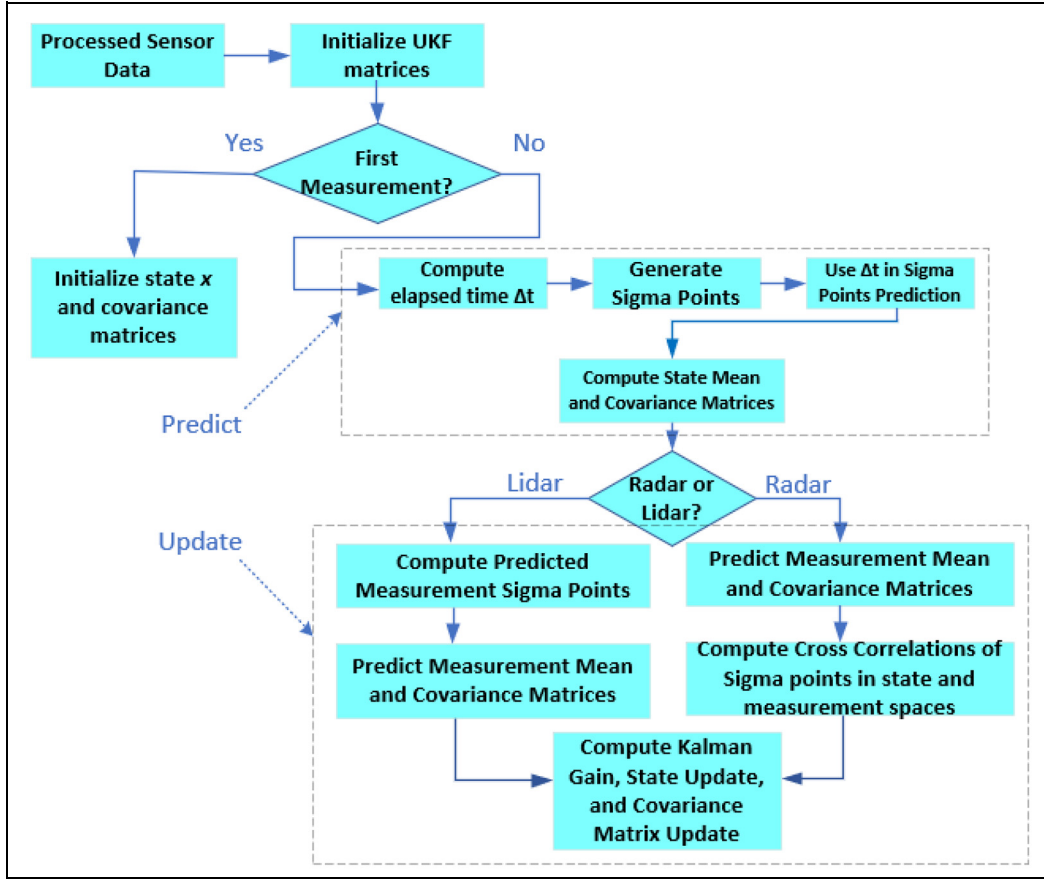


Figure 4. Lidar and radar data fusion using UKF.

a next-time-step prediction for sigma points (\hat{X}_{k+1}) is carried out using equation (8) while employing the moving object nonlinear motion model given in equation (15). The resulted predicted sigma points are then used to compute the state mean (\hat{x}_{k+1}) and covariance (\hat{P}_{k+1}) matrices using equation (9).

Then, the fusion technique thus branches into two directions based on the source of the last sensor data measurement. If the source is a radar, and employing the nonlinear radar measurement model (equation (20)), the predicted measurement sigma points (\hat{Z}_{k+1}) are calculated from the predicted state sigma points (\hat{X}_{k+1}) using equation (11). Then, the predicted measurements (\hat{z}_{k+1}) and their covariance matrix (S_{k+1}) are calculated based on equation (12) using the measurement noise covariance matrix R_{radar} given in equation (26). Then, \hat{x}_{k+1} and \hat{z}_{k+1} are used to compute the cross-correlation matrix (T_{k+1}) between the sigma points in the state space (\hat{X}_{k+1}) and the measurement space (\hat{Z}_{k+1}) as in equation (13). Based on this cross-correlation matrix, the KF gain (K_{k+1}) is then calculated and used to compute the updated object's state vector (x_{k+1}) and covariance matrix (P_{k+1}) as shown by equation (13).

If the measurement data source is a lidar, and employing the linear lidar measurement model (H_{lidar}) shown in equation (23), the predicted measurement sigma points (\hat{Z}_{k+1}) is directly calculated from (\hat{X}_{k+1}). Then, the predicted measurements (\hat{z}_{k+1}) and their

covariance matrix (S_{k+1}) are calculated based on equation (12) using the measurement noise covariance matrix R_{lidar} given in equation (24). Then, \hat{x}_{k+1} and \hat{z}_{k+1} are used to compute the cross-correlation matrix (T_{k+1}) between the sigma points in the state space (\hat{X}_{k+1}) and the measurement space (\hat{Z}_{k+1}) as in equation (13). Based on this cross-correlation matrix, the KF gain (K_{k+1}) is then calculated and used to compute the updated object's state vector (x_{k+1}) and covariance matrix (P_{k+1}) as shown by equation (13).

Implementation of KFs

Both KFs (EKF and UKF) are implemented using the high-performance language GCC C++³⁴ on Ubuntu Linux operating system.³⁵ This combination is fitting for the required real-time performance.³⁶ A C++ numerical solver, matrix and vector operations package "Eigen,"³⁷ is used to numerically calculate the Jacobians, object model, and effectively performing the predict and update steps.

In the following sections, several important matters that have a crucial effect on the KFs design process will be highlighted and discussed.

Noise parameters setting

The object motion model described by equations (14)–(17) includes several noise parameters that need to be

Table 1. The Kalman filter noise parameters.

Parameter	EKF	UKF
σ_a m/s ²	3.0	1.0
σ_{a_x} m/s ²	3.0	—
σ_{a_y} m/s ²	3.0	—
$\sigma_{\dot{\psi}}$ rad/s ²	0.6	0.6
$\sigma_{\ddot{\psi}}$ rad/s ²	0.06	0.06
σ_{p_x} (lidar) m	0.15	0.15
σ_{p_y} (lidar) m	0.15	0.15
σ_{ρ} (radar) m	0.3	0.3
σ_{φ} (radar) rad	0.03	0.03
$\sigma_{\dot{\rho}}$ (radar) m/s	0.3	0.3

EKF: Extended Kalman Filter; UKF: Unscented Kalman Filter.

carefully set. To set the two process noise parameters as an example, the longitudinal acceleration noise standard deviation σ_a and the yaw acceleration noise $\sigma_{\ddot{\psi}}$, one has to approximate that the expected top acceleration road objects can exhibit both longitudinal and angular as an initial guess and then fine-tune these values through trial-and-error iterations. After, exhaustive tuning process, the most appropriate valuation for σ_a and $\sigma_{\ddot{\psi}}$ is found to be 3 m/s² and 0.6 rad/s². Similarly, the other parameters can be set and accordingly, Table 1 presents the fine-tuned parameters for both EKF and UKF.

σ_a is set to 3 m/s² because $\{-6, 6\}$ m/s² is the expected boundary of the longitudinal acceleration (e.g. road vehicles) as statistically 95% of the time the acceleration will stay within the $\{-2\sigma_a, 2\sigma_a\}$ range. The same principle applies to $\sigma_{\ddot{\psi}}$ where the appraisal of how fast a vehicle can accelerate or decelerate while completing a circular path is guiding the selection of the set value of 0.6 rad/s².

Consistency measures for KFs

The KF design is considered consistent if the estimation error (\hat{y}_k in equation (3)) is unbiased, that is, has zero-mean, and that the actual mean square error of the filter matches the filter-calculated state covariance. As a measure of filter consistency, the time-average Normalized Innovation Squared (NIS)³⁸ can be used to finetune the noise parameters. The NIS follows a Chi-square distribution (χ^2), and based on the number of the measurement-vector dimension, it is used to assess whether KF's noise assumptions are consistent with the realized measurements. The metric, described by equation (27), is used to calculate the NIS value at each sample k , and then, averaging these values ($NIS_{Average}$) over a moving window of measurements of length N

$$NIS_k = (z_{k+1} - \hat{z}_k)^T S_k^{-1} (z_{k+1} - \hat{z}_k) \quad (27)$$

$$NIS_{Average} = \frac{1}{N} \sum_{k=1}^N NIS_k$$

Table 2. Initialization of Kalman filter states.

Parameter	EKF	UKF
p_x m	First raw x-reading	First raw x-reading
p_y m	First raw y-reading	First raw y-reading
v m/s	0.0	0.0
ψ rad	0.0	0.0
$\dot{\psi}$ rad/s	0.0	0.0
$\sigma_{\hat{p}_x}$ m	1.0	1.0
$\sigma_{\hat{p}_y}$ m	1.0	1.0
$\sigma_{\dot{v}}$ m/s	$\sqrt{1000}$	$\sqrt{1000}$
$\sigma_{\dot{\psi}}$ rad	$\sqrt{1000}$	$\sqrt{1000}$
$\sigma_{\ddot{\psi}}$ m/s ²	$\sqrt{1000}$	$\sqrt{1000}$

EKF: Extended Kalman Filter; UKF: Unscented Kalman Filter.

The radar's measurement is a three-dimensional vector (3 degrees of freedom); therefore, for a consistent EKF or UKF design, the values of NIS_k should be less than "7.815" in 95% of the time-steps. Likewise, the lidar's measurement is a two-dimensional vector (2 degrees of freedom); therefore, for a consistent EKF or UKF design, the values of NIS_k should be less than "5.991" in 95% of the time steps.

In both sensors, the $NIS_{Average}$'s value should not be too low as it means the process uncertainty is highly overestimated or too high as it means the process uncertainty is highly underestimated. It actually should be around the middle or the two-third of their corresponding threshold ranges.

Initialization of KFs

The proper initialization of the KF is very crucial to its subsequent performance.³⁹ The main initialized variables are the estimate state vector (x) and its estimate covariance matrix (P).

The first two terms of the state vector x given by equation (14) are p_x and p_y , which are simply initialized using the first received raw sensor measurement. For the other three terms of the state vector, intuition augmented with some trial-and-error is used to initialize these variables as listed in Table 2.

The state covariance matrix is initialized as a diagonal matrix that contains the covariance of each variable estimate (equation (28)). The initialization logic works as follows: little or almost no correlation among the state variables (independent variables) is assumed; therefore, the off-diagonal terms (covariances between variables) are initialized to zeros. Each diagonal term represents the variance (confidence) of each state element estimate as shown in equation (28). The variance of each element is initialized depends on the a priori information about this element. Since the first two elements of the state vector (p_x and p_y) are initialized using the first raw reading of the sensors, then both $\sigma_{\hat{p}_x}^2$ and $\sigma_{\hat{p}_y}^2$ are set to small values. However, little a priori information is known about the other three terms ($v, \psi, \dot{\psi}$);

therefore, they have been initialized to large values as listed in Table 2. Note that radar velocity measurement ($\dot{\rho}$) cannot directly be used to initialize the state vector velocity (object velocity v) as they are not the same

$$P = \begin{bmatrix} \sigma_{\hat{p}_x}^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{\hat{p}_y}^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{\hat{v}}^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\hat{\psi}}^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\hat{\dot{\psi}}}^2 \end{bmatrix} \quad (28)$$

Performance measures for KFs

To check the performance of the KF, in terms of how far the estimated results from the true results (ground truth). There are many evaluation metrics,⁴⁰ but perhaps the most common one is the Root Mean Squared Error (RMSE) given in equation (29). The metric is calculated over a moving window of measurements of length N .⁴¹ x_k^{est} is the estimated state vector of the KF given in equation (4), and x_k^{true} is the true state vector supplied by the simulator or given as training data during the KF design phase

$$RMSE = \sqrt{\frac{1}{N} \sum_{k=1}^N (x_k^{est} - x_k^{true})^2} \quad (29)$$

Testing and evaluation results

Extensive trials-and-errors attempts are used to tune the many hyper-parameters of the *LR_ODT*. However, to be more consistent and accurate, numerical Key Performance Indicators (KPIs) are constructed and coded as in equations (27) and (29) to evaluate the performance of the fusion technique under the given set of hyper-parameters.

Several test tracks have been used to evaluate the performance of the *LR_ODT* under different sets of hyper-parameters in an iterative tuning process. Examples of these test tracks are shown in Figures 5–7. These tracks are representing three different moving objects, bicycle, car, and pedestrian, respectively, to emulate various motion profiles and velocities.

Table 3 presents the testing results of the *LR_ODT* fusion algorithm that uses the UKF and compares it with that uses the EKF. The performance evaluation is carried out on the three test tracks. The RMSE KPI (equation (29)) is used to compare both UKF and EKF performances on the five state variables p_x , p_y , v_x , v_y , and ψ . The KPI is comparing each estimated state variable to its ground-truth value and finding the error. The lower the value of the KPI, the better the performance. Moreover, the bicycle tracking results using the UKF are depicted in the form of x - y position (shown in Figure 8), longitudinal velocity (shown in Figure 9), yaw angle (shown in Figure 11), and yaw rate (shown in

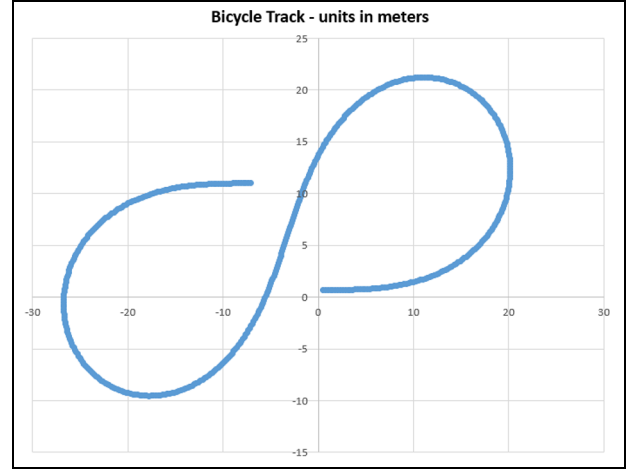


Figure 5. Test track for a moving bicycle.

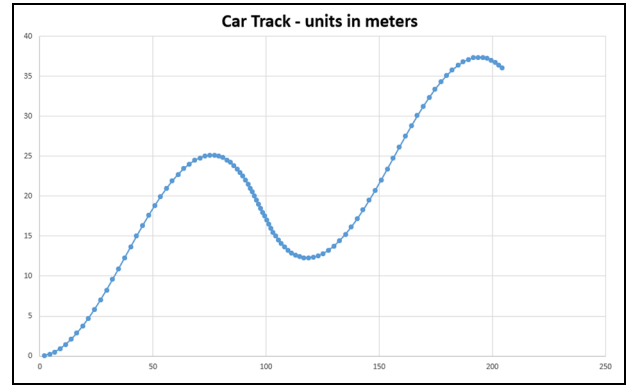


Figure 6. Test track for a moving car.

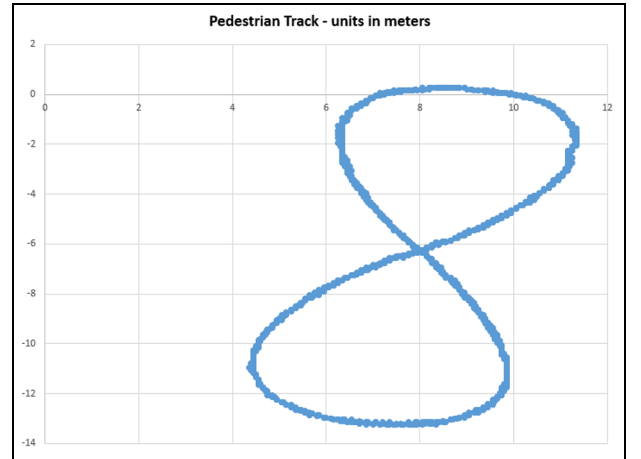


Figure 7. Test track for a walking pedestrian.

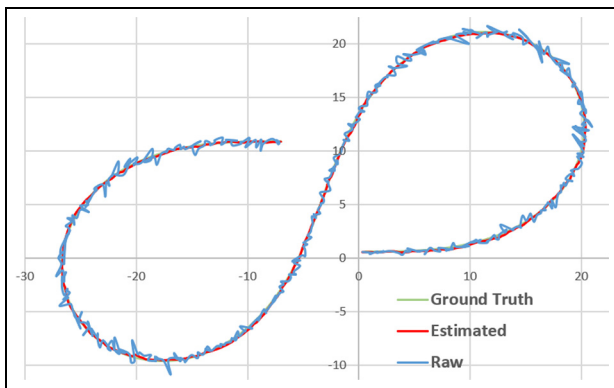
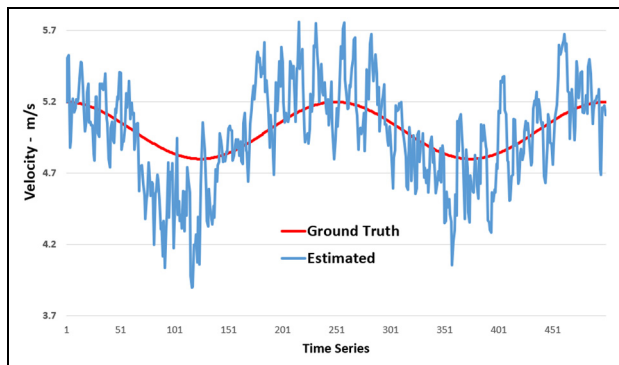
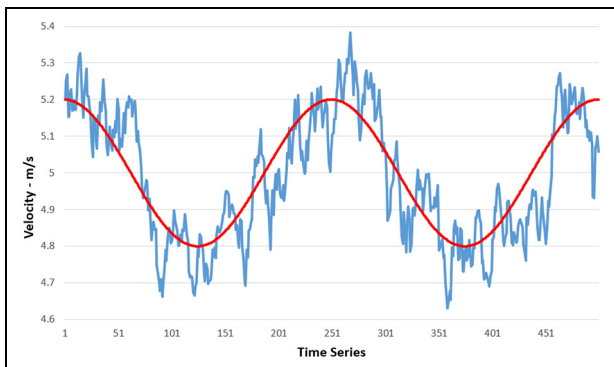
Figure 12). The bicycle tracking results using the EKF are reported in the third column of Table 3 and the estimated longitudinal velocity profile is depicted in Figure 10.

The UKF design-consistency indicator (NIS) results are reported in Table 3. The table reports values for the

Table 3. Performance evaluation of Kalman filters.

Test case	State variable	RMSE EKF	RMSE UKF	Percentage improvement
Bicycle	p_x	0.0959	0.0648	-32.43
	p_y	0.0931	0.0809	-13.10
	v_x	0.2953	0.1452	-50.83
	v_y	0.3750	0.1592	-57.55
	ψ	0.0728	0.0392	-46.15
Car	p_x	0.1946	0.1857	-4.57
	p_y	0.1903	0.1899	-0.21
	v_x	0.5190	0.4745	-8.57
	v_y	0.8111	0.5075	-37.43
	ψ	0.4037	0.2580	-36.09
Pedestrian	p_x	0.0758	0.0652	-13.98
	p_y	0.0842	0.0605	-28.15
	v_x	0.6323	0.5332	-15.67
	v_y	0.5807	0.5442	-6.29
	ψ	0.2301	0.2075	-9.82

RMSE: Root Mean Squared Error; EKF: Extended Kalman Filter; UKF: Unscented Kalman Filter.

**Figure 8.** Performance of the UKF position estimation on the bicycle track.**Figure 10.** EKF-velocity-estimation performance on the bicycle track.**Figure 9.** UKF-velocity-estimation performance on the bicycle track.

NIS by taking into consideration the estimated measurements by lidar alone, the estimated measurements by radar alone, and after combining both lidar and radar measurements. The values reported in the fifth column of the table shows a very consistent filter design. All the values are significantly lower than 5%.

To assess the significance of the fusion between lidar and radar in tracking, the UKF is tested in one time with measurements from lidar alone and another time with measurements from radar alone. The results reported in Table 4 show how fusion makes the difference and substantially improves accuracy. The estimation of all state variables is spectacularly improved. For example, the RMSE of x -position (p_x) estimation is reduced by 60% compared to “lidar-alone” and 60% compared to “radar-alone” estimations. Moreover, the RMSE of x -velocity (v_x) estimation is reduced by 30% compared to “lidar-alone” and 26% compared to “radar-alone” estimations. The NIS values are calculated as well for “lidar-alone” and “radar-alone” cases to test the consistency of the UKF in their cases. The reported values in Table 5 and the performance depicted in Figures 13 and 14 show that fusion significantly improves the consistency. The NIS values that exceed the 95%-threshold have been reduced by 31% compared to the “lidar-alone” and 38.5% compared to the “radar-alone” ones.

Comparing UKF performance with that of the EKF, Table 3 details the results. It is obvious that

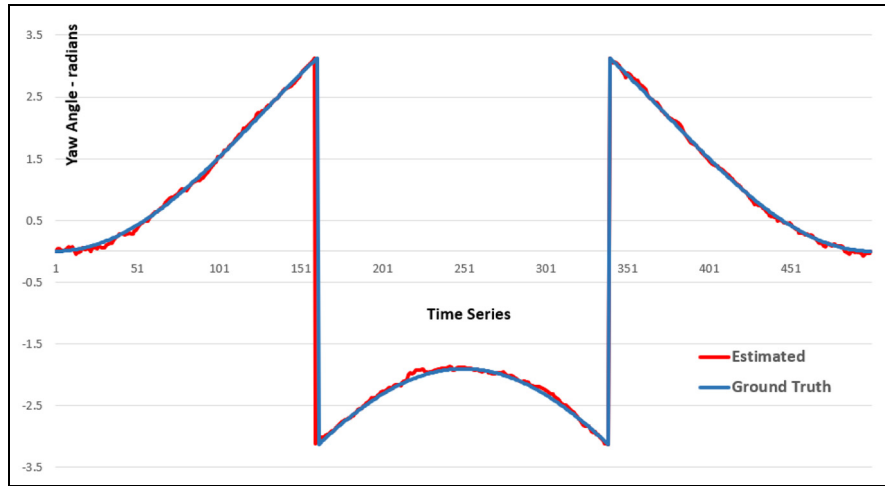


Figure 11. UKF-yaw-angle estimation performance on the bicycle track.

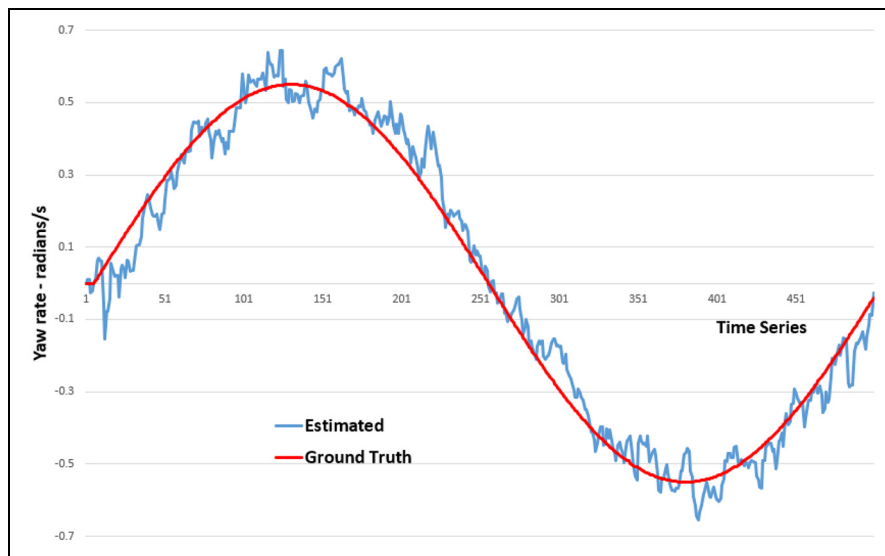


Figure 12. UKF-yaw-rate estimation performance on the bicycle track.

Table 4. Sensor fusion evaluation of the UKF (Bicycle Track).

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

RMSE: Root Mean Squared Error; NIS: Normalized Innovation Squared.

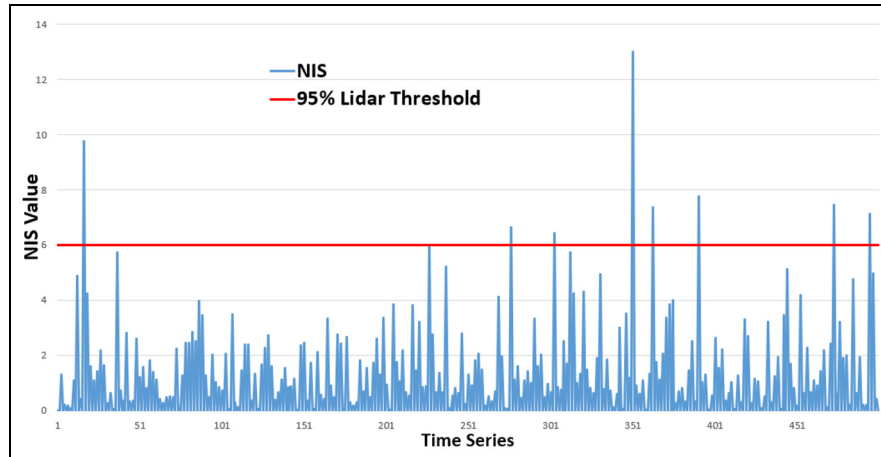
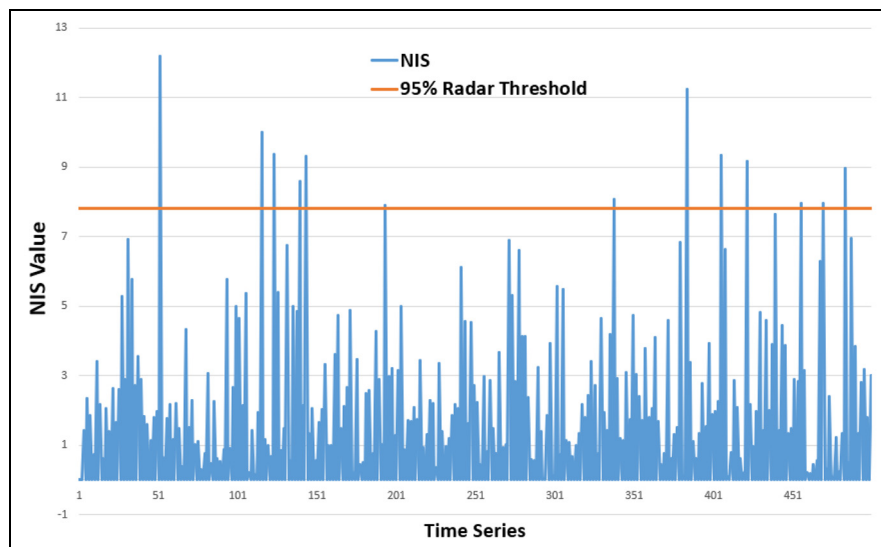
UKF outperforms EKF at all velocity and motion profiles. The accuracy of all the estimated states is sustainably higher using the UKF. States that got affected more with non-linearities in the object model

(e.g. v_x , v_y , and ψ) have seen more accuracy improvement. Comparing the velocity profiles in both Figure 9 and Figure 10 gives some insight into the achieved improvement.

Table 5. Consistency evaluation of Kalman filters.

	NIS average	NIS Min	NIS Max	NIS above 95% threshold
Radar	2.8054	0.05806	11.124	3.6%
Lidar	2.7903	0.00116	14.749	1.6%
Lidar + Radar	2.2797	0.00116	14.749	2.2%

NIS: Normalized Innovation Squared.

**Figure 13.** Lidar NIS values for UKF on the bicycle track.**Figure 14.** Radar NIS values for UKF on the bicycle track.

The *LR_ODT* proved to be well enough fast in execution to be used in real time. Using an Intel Core i5 with 1.6 GHz and 8 GB RAM which is a very moderate computational platform, the following measurements (Table 6) are collected for both EKF and UKF.

Table 6 shows how the EKF is twice faster than UKF. By considering that the lidar/radar measurements are collected at approximately 30 fps rate, the measurement cycle is 33.3 ms, which is large enough to

Table 6. Sensor fusion execution time of EKF and UKF.

Phase	EKF (μ s)	UKF (μ s)
Predict	4.827	27.32
Update	16.40	14.20
Total	21.23	41.52

EKF: Extended Kalman Filter; UKF: Unscented Kalman Filter.

Table 7. LR_ODT execution time for single object.

	EKF (ms)	UKF (ms)
State estimation execution time	0.637	1.246
Clustering and object association	0.427	0.835
Control code overhead—20%	0.213	0.416
Total—30 fps	1.276	2.496

EKF: Extended Kalman Filter; UKF: Unscented Kalman Filter.

be utilized for tracking 25 objects using EKF or 13 objects using UKF according to the data in Table 7.

The proposed *LR_ODT* technique is fit for deployment in any kind of autonomous vehicles, like gas-powered vehicles or electric vehicles.⁴² In autonomous electric vehicles, it will support functions like coordinated wheel control,⁴³ lateral control,⁴⁴ lane change maneuver,⁴⁵ and path tracking.⁴⁶

Conclusion

In this article, a real-time road-object detection and tracking method (*LR_ODT*) for autonomous cars is proposed, implemented, and described in detail. The method uses a tailored unscented KF to perform data fusion for the mounted lidar and radar devices on the ego car. The raw data of the lidar/radar are getting clustered using both GB-DBSCAN and RANSAC algorithms to produce the raw object's pose and to determine its geometrical shape. The *LR_ODT* method is fully implemented using GCC C++ in addition to advanced math libraries to optimize its real-time performance. The design steps, initialization, and tuning of both the EKF and the UKF are described in detail. The consistency evaluation of both filters has been explained as well.

The validation results show that the proposed method is reliably able to detect and track three types of street objects: bicycle, car, and pedestrians on three different tracks and speed profiles. The employed generic object motion model is comprehensive and is described using five state variables. The UKF has outperformed the EKF on all test cases and all the state variable levels (−24% average RMSE), despite its considerably more complex design and higher execution time.

Comparing the validation results of the UKF applied to a single sensor to the one employing the fusion of multiple sensors shows how outstanding is the improvement in tracking performance using the later (−29% RMSE with lidar and −38% RMSE with radar).

The measured throughput (execution time) using an affordable CPU proved that the *LR_ODT* method is very suitable for real-time multi-object detection and tracking.

In the future, it is intended to add a front camera accompanied by a feature-extraction deep learning technique to the presented fusion technique and further investigate the benefits it will add to the overall

tracking performance. Furthermore, the *LR_ODT* will be augmented with other road objects like guardrails, trucks, and animals.


Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Wael Farag  <https://orcid.org/0000-0002-9191-1824>

References

1. Farag W. Traffic signs classification by deep learning for advanced driving assistance systems. *Intel Decis Tech* 2019; 13(3): 215–231.
2. Farag W and Saleh Z. Road lane-lines detection in real-time for advanced driving assistance systems. In: *International conference on innovation and intelligence for informatics, computing, and technologies (3ICT)*, Bahrain18–20 November 2018. New York: IEEE.
3. Farag W. A comprehensive real-time road-lanes tracking technique for autonomous driving. *Int J Comput Digital Syst* 2020; 9(3): 349–362.
4. Farag W and Saleh Z. Behavior cloning for autonomous driving using convolutional neural networks. In: *International conference on innovation and intelligence for informatics, computing, and technologies (3ICT)*, Bahrain18–20 November 2018. New York: IEEE.
5. Farag W. Recognition of traffic signs by convolutional neural nets for self-driving vehicles. *Int J Knowle Based Intel Eng Syst* 2018; 22(3): 205–214.
6. Farag W and Saleh Z. Tuning of PID track followers for autonomous driving. In: *International conference on innovation and intelligence for informatics, computing, and technologies (3ICT)*, Bahrain18–20 November 2018. New York: IEEE.
7. Farag W. Complex trajectory tracking using PID control for autonomous driving. *Int J Intel Transp Syst Res* 2020; 18: 356–366.
8. Farag W. Complex-track following in real-time using model-based predictive control. *Int J Intel Transp Syst Res*. Epub ahead of print 12 June 2020. DOI: 10.1007/s13177-020-00226-1.
9. Farag W and Saleh Z. An advanced road-lanes finding scheme for self-driving cars. In: *2nd smart cities symposium (SCS'19)*, Bahrain, 24–26 March 2019. New York: IEEE.
10. Farag W. Safe-driving cloning by deep learning for autonomous cars. *Int J Adv Mechatron Syst* 2019; 7(6): 390–397.
11. Farag W. Cloning safe driving behavior for self-driving cars using convolutional neural networks. *Recent Patent Comput Sci* 2019; 12: 120–127.
12. Farag W and Saleh Z. An advanced vehicle detection and tracking scheme for self-driving cars. In: *2nd smart cities*

- symposium (SCS'19)*, Bahrain, 24–26 March 2019. New York: IEEE.
13. Yurtsever E, Lambert J, Carballo A, et al. A survey of autonomous driving: common practices and emerging technologies, 2020, <https://arxiv.org/abs/1906.05113>
 14. Che E, Jung J and Olsen MJ. Object recognition, segmentation, and classification of mobile laser scanning point clouds: a state of the art review. *Sensors* 2019; 19: 810.
 15. Xie Y, Tian J and Zhu X. A review of point cloud semantic segmentation, 2019, <https://arxiv.org/pdf/1908.08854.pdf>
 16. Lidman P and Luu S. *Clustering, shape extraction and velocity estimation applied to radar detections*. M.sc. Thesis, Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, 2018.
 17. Farag W. Real-time detection of road lane-lines for autonomous driving. *Recent Adv Comput Sci Commun* 2020; 13: 265–274.
 18. Farag W. A comprehensive vehicle-detection-and-tracking technique for autonomous driving. *Int J Comput Digital Syst* 2020; 9(4): 567–580.
 19. Dietmayer K, Kellner D and Klappstein J. Grid-based DBSCAN for clustering extended objects in radar data. In: *IEEE intelligent vehicles symposium*, Alcalá De Henares, 3–7 June 2012. New York: IEEE.
 20. Fischler M and Bolles R. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Comm ACM* 1981; 24(6): 381–395.
 21. Gohring D, Wang M, Schnurmacher M, et al. Radar/lidar sensor fusion for car-following on highways. In: *5th international conference on automation, robotics and applications*, Wellington, 6–8 December 2011, pp.407–412. New York: IEEE.
 22. Kaempchen N, Fuerstenberg KC, Skibicki AG, et al. Sensor fusion for multiple automotive active safety and comfort applications. In: Valldorf J and Gessner W (eds) *Advanced microsystems for automotive applications*, vol. 2. New York: Springer, 2004, pp.137–163.
 23. Zarchan P and Musoff H. Fundamentals of Kalman filtering: a practical approach. *American Institute of Aeronautics and Astronautics*, 2013, <https://arc.aiaa.org/doi/book/10.2514/4.102776#:~:text=Fundamentals%20of%20Kalman%20Filtering%2C%20Fourth,Kalman%20filters%20can%20be%20designed>
 24. Chavez-Garcia RO and Aycard O. Multiple sensor fusion and classification for moving object detection and tracking. *IEEE T Intel Transp Syst* 2015; 17(2): 252–534.
 25. Rangesh A and Trivedi MM. No blind spots: full-surround multi-object tracking for autonomous vehicles using cameras and LiDARs. *IEEE T Intel Veh* 2019; 4: 588–589.
 26. Hajri H and Rahal MC. Real-time lidar and radar high-level fusion for obstacle detection and tracking with evaluation on a ground truth, 2019, <https://arxiv.org/abs/1807.11264>
 27. Jahromi BS, Tulabandhula T and Cetin S. Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles. *Sensors* 2019; 19(20): 4357.
 28. Wan EA and Van Der Merwe R. The unscented Kalman filter for nonlinear estimation. In: *Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium (Cat. No.00EX373)*, Lake Louise, AB, Canada, 4 October 2000. New York: IEEE.
 29. Julier SJ and Uhlmann JK. Unscented filtering and nonlinear estimation. *Proc IEEE* 2004; 92(3): 401–422.
 30. Einicke GA and White LB. Robust extended Kalman filtering. *IEEE T Signal Pr* 1999; 47(9): 2596–2599.
 31. Best MC and Bogdanski K. Extending the Kalman filter for structured identification of linear and nonlinear systems. *Int J Model Ident Control* 2017; 27(2): 114–124.
 32. Schubert R, Richter E and Wanielik G. Comparison and evaluation of advanced motion models for vehicle tracking. In: *11th international conference on information fusion*, Cologne, 30 June–3 July 2008. New York: IEEE.
 33. Sander J, Xu X, Ester M, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the 2nd international conference on knowledge discovery and data mining*, Portland, OR, 2–4 August 1996, pp.226–231. New York: IEEE.
 34. GCCC++, <https://gcc.gnu.org/> (accessed on 11 March 2020).
 35. Ubuntu Linux, <https://www.ubuntu.com/> (accessed on 11 March 2020).
 36. Naguib M and Farag W. Automatic selection of compiler options using genetic techniques for embedded software design. In: *2013 IEEE 14th international symposium on computational intelligence and informatics (CINTI)*, Budapest, Hungary, 19–21 November 2013. New York: IEEE.
 37. Eigen, http://eigen.tuxfamily.org/index.php?title=Main_Page (accessed 11 March 2020).
 38. Piché R. Online tests of Kalman filter consistency. *Int J Adapt Control Signal Pr* 2016; 30(1): 115–124.
 39. Zhao S and Huang B. On initialization of the Kalman filter. In: *2017 6th international symposium on advanced control of industrial processes (AdCONIP)*, Taipei, Taiwan, 28–31 May 2017. New York: IEEE.
 40. Saho K. Kalman filter for moving object tracking: performance analysis and filter design. *Intechopen*, 2018, <https://www.intechopen.com/books/kalman-filters-theory-for-advanced-applications/kalman-filter-for-moving-object-tracking-performance-analysis-and-filter-design>
 41. Farag W. *Synthesis of intelligent hybrid systems for modeling and control*. PhD Thesis, University of Waterloo, ON, Canada, 1998.
 42. Farag W. A lightweight vehicle detection and tracking technique for advanced driving assistance systems. *J Intel Fuzzy Syst* 2020; 39(3): 2693–2710.
 43. Farag W. Complex track maneuvering using real-time MPC control for autonomous driving. *Int J Comput Digital Syst* 2020; 9(5): 909–920.
 44. Norouzi A, Masoumi M, Barari A, et al. Lateral control of an autonomous vehicle using integrated backstepping and sliding mode controller. *Proc IMechE, Part K: J Multi-body Dynamics* 2019; 233(1): 141–151.
 45. Norouzi A, Kazemi R and Azadi S. Vehicle lateral control in the presence of uncertainty for lane change maneuver using adaptive sliding mode control with fuzzy boundary layer. *Proc IMechE, Part I: J Systems and Control Engineering* 2018; 232(1): 12–28.
 46. Hang P and Chen X. Integrated chassis control algorithm design for path tracking based on four-wheel steering and direct yaw-moment control. *Proc Inst Mech Eng, Part I: J Systems and Control Engineering* 2019; 233(6): 625–641.