Journal of ELECTRICAL ENGINEERING, VOL 71(2020), NO3, 138–149

sciendo

PAPERS

Road-objects tracking for autonomous driving using lidar and radar fusion

Wael Farag[1 2]

In this paper, based on the fusion of Lidar and Radar measurement data, a real-time road-Object Detection and Tracking (LR_ODT) method for autonomous driving is proposed. The lidar and radar devices are installed on the ego car, and a customized Unscented Kalman Filter (UKF) is used for their data fusion. Lidars are accurate in determining objects' positions but significantly less accurate on measuring their velocities. However, Radars are more accurate on measuring objects velocities but less accurate on determining their positions as they have a lower spatial resolution. Therefore, the merits of both sensors are combined using the proposed fusion approach to provide both pose and velocity data for objects moving in roads precisely. The Grid-Based Density-Based Spatial Clustering of Applications with Noise (GB-DBSCAN) clustering algorithm is used to detect objects and estimate their centroids from the lidar and radar raw data. Then, the estimation of the object's velocity as well as determining its corresponding geometrical shape is performed by the RANdom SAmple Consensus (RANSAC) algorithm. The proposed technique is implemented using the high-performance language C++ and utilizes highly optimized math and optimization libraries for best real-time performance. The performance of the UKF fusion is compared to that of the Extended Kalman Filter fusion (EKF) showing its superiority. Simulation studies have been carried out to evaluate the performance of the LR_ODT for tracking bicycles, cars, and pedestrians.

K e y w o r d s: sensor fusion, kalman filter, object detection, object tracking, adas, autonomous driving

1 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 ones 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, *ie* 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 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 objects 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 paper, 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 (*ie* 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].

[1] College of Engineering & Technology, American University of the Middle East, Kuwait, [2] Electrical Engineering Department, Cairo University, Egypt, wael.farag@aum.edu.kw, wael.farag@cu.edu.eg

DOI:10.2478/jee-2020–0021, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

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-Garci *et al* [24] 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 *et al* [25] 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 *et al* [26] employ the global nearest neighbor standard filter (GNN) 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 paper 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: Kalman filter 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 Kalman filters. Despite the previously mentioned works [26], 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 paper 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 paper 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 (5 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.

2 Kalman filter overview

The Kalman filter [23] 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

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

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: The predication step – estimates the next state as follows

$$\begin{aligned} \hat{x}_k &= Fx_{k-1} + Bu_k, \\ \hat{P}_k &= FP_{k-1}F^\top + Q_k. \end{aligned} \tag{2}$$

The measurement update step – works as follow

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

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 Kalman filter [30] is introduced. The EKF estimation process is, instead of (1), represented by

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

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

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

The derivatives of $f(.)$ and $h(.)$ with respect to x are called Jacobians. The F_j and H_j Jacobians are calculated as in (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

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

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

Due to the EKF 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 Kalman filter is introduced [28] 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 Fig. 1.

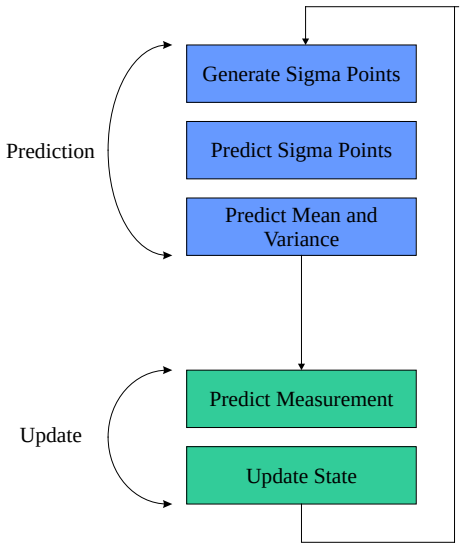

Fig. 1. UKF roadmap

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

$$X_k = [x_k x_k + \sqrt{\lambda + n_x)P_k} x_k - \sqrt{(\lambda + n_x)P_k}], \tag{7}$$

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

$$X_{k+1} = f(X_k, \nu_k). \tag{8}$$

In the next step, the predicted state-mean and covariance matrices are calculated from the predicted sigma points as

$$\begin{aligned} \hat{x}_{k+1} &= \sum_{i=0}^{n_x} w_i \hat{X}_{k+1,i}, \\ \hat{P}_{k+1} &= \sum_{i=0}^{2n_x} w_i (\hat{X}_{k+1,i} - \hat{x}_{k+1})(\hat{X}_{k+1,i} - \hat{x}_{k+1})^\top, \end{aligned} \tag{9}$$

where w_i' are the sigma-point weights that are used here to invert the spreading of the sigma points. These weights are calculated as

$$\begin{aligned} w_i &= \frac{\lambda}{\lambda + n_x}, \quad i = 0, \\ w_i &= \frac{\lambda}{2(\lambda + n_x)}, \quad i = 1, \ldots, n_x. \end{aligned} \tag{10}$$

In the measurement prediction step, each generated sigma point is inserted in the UKF nonlinear measurement model

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

to produce the matrix of the predicted measurement sigma points, which has an $n \times n_x$ dimension.

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 is

$$\begin{aligned}\hat{z}_{k+1} &= \sum_{i=0}^{n_x} w_i \hat{Z}_{k+1,i},\\ \hat{S}_{k+1} &= \sum_{i=0}^{2n_x} w_i (\hat{Z}_{k+1,i} - \hat{z}_{k+1})(\hat{Z}_{k+1,i} - \hat{z}_{k+1})^\top + R,\end{aligned} \tag{12}$$

where w_i' are the sigma-point weights determined by (10).

The final step is the UKF state update, where the UKF gain matrix (K) is calculated as

$$\begin{aligned}T_{k+1} &= \sum_{i=0}^{2n_x} w_i \big(\hat{X}_{k+1,i} - \hat{x}_{k+1}\big)\big(\hat{Z}_{k+1,i} - \hat{z}_{k+1}\big)^\top,\\ K_{k+1} &= T_{k+1} S_{k+1}^{-1},\\ x_{k+1} &= \hat{x}_{k+1} + K_{k+1}(\hat{z}_{k+1} - z_{k+1}),\\ P_{k+1} &= \hat{P}_{k+1} - K_{k+1} S_{k+1} K_{k+1}^\top,\end{aligned} \tag{13}$$

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).

3 The moving object model

The state of the moving object is determined by the five variables grouped into the state vector x shown in (14), where p_x, and p_y are the object position in the x and y -axis respectively as shown in Fig. 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 the rate of change of the object-yaw angle.

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

The nonlinear $x_{k+1} = f(x_k, \nu_k)$ difference equation that describes the motion model of the object is derived based on the state vector x

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

$$\nu_k = \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \cos(\psi_k)\nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \sin(\psi_k)\nu_{a,k} \\ \Delta t \nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \nu_{\ddot{\psi},k} \\ \Delta t \nu_{\ddot{\psi},k} \end{bmatrix}, \tag{16}$$

$$\begin{aligned}\Delta_t &= t_{k+1} - t_k,\\ \nu_{a,k} &\approx \mathcal{N}, (0, \sigma_a^2),\\ \nu_{\ddot{\psi},k} &\approx \mathcal{N}(0, (\sigma_{\ddot{\psi}}^2),\end{aligned} \tag{17}$$

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

If $\dot{\psi}$ is zero, to avoid dividing by zero (15), the following approximation is used to calculate the prediction of p_x, and p_y

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

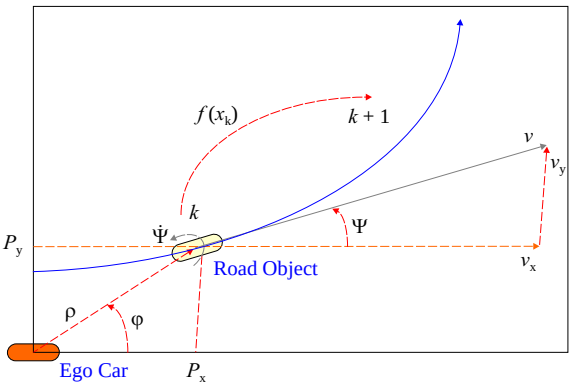

Fig. 2. An object motion model

4 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) is getting processed before being supplied to the EKF.

The processing is performed using clustering and association algorithms. DBSCAN [33] 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 $\min Pts$, which is the least number of detections that must be within a distance ε from p, including p itself, to form a cluster. By choosing ϵ and $\min Pts$, it is then possible to decide the necessary density for a group of points to form a cluster, however, these fixed parameters not convenient if various types and topologies of road objects need to

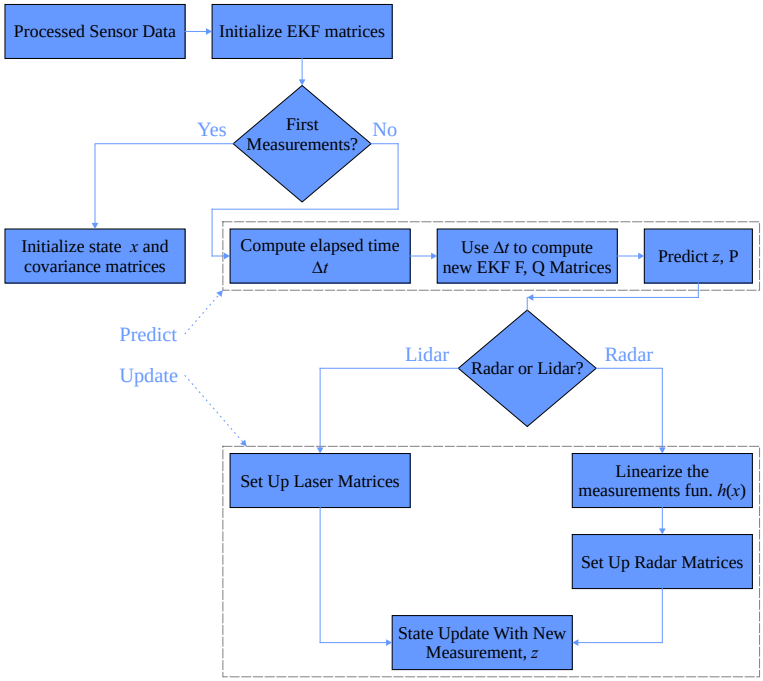

Fig. 3. Lidar and radar data fusion using EKF

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 can use a more dynamic, elliptic, search area.

While GB-DBSCAN is used for coarse clustering, the RANSAC [20] 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

$$z_{\text{lidar}} = \begin{pmatrix} p_x \\ p_y \end{pmatrix}, \quad z_{\text{radar}} = \begin{pmatrix} \rho \\ \varphi \\ \dot{\rho} \end{pmatrix}. \tag{19}$$

The mapping function that specifies how lidar Cartesian coordinates got mapped to the radar polar coordinates are

$$h(x) = \begin{pmatrix} \rho \\ \varphi \\ \dot{\rho} \end{pmatrix} = \begin{pmatrix} \sqrt{p_x^2 + p_y^2} \\ \arctan \dfrac{p_y}{p_x} \\ \dfrac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}} \end{pmatrix}, \tag{20}$$

$$p_x = \rho \cos(\varphi), \quad p_y = \rho \sin(\varphi). \tag{21}$$

The EKF state nonlinear model is the moving object model that is described in detail in Section 3 and mathematically represented by (15-17). The Jacobian of this state model is

$$F_j = \begin{bmatrix} 1 & 0 & \frac{1}{\dot{\psi}} S(\psi, \dot{\psi}) & -\frac{v}{\dot{\psi}} C(\psi, \dot{\psi}) & F(\psi, \dot{\psi}) \\ 0 & 1 & \frac{1}{\dot{\psi}} C(\psi, \dot{\psi}) & \frac{v}{\dot{\psi}} (\psi, \dot{\psi}) & G(\psi, \dot{\psi}) \\ 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{22}$$

where: $S(\psi, \dot{\psi}) = -\sin(\psi) + \sin(\Delta t \dot{\psi} + \psi)$, $C(\psi, \dot{\psi}) = \cos(\psi) - \cos(\Delta t \dot{\psi} + \psi)$, and $F(\psi, \dot{\psi}) = \frac{v \Delta t}{\dot{\psi}} \cos(\Delta t \dot{\psi} + \psi) - \frac{v}{\dot{\psi}^2} S(\psi, \dot{\psi})$, while $G(\psi, \dot{\psi}) = \frac{v \Delta t}{\dot{\psi}} \sin(\Delta t \dot{\psi} + \psi) - \frac{v}{\dot{\psi}^2} C(\psi, \dot{\psi})$.

Then, the associated state covariance matrix (Q) is given by

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

The lidar measurement model and the measurement noise covariance matrix based on the state vector in (14) are

$$H_{\text{lidar}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad R_{\text{lidar}} = E[\omega \, \omega^\top] = \begin{bmatrix} \sigma_{p_x}^2 & 0 \\ 0 & \sigma_{p_y}^2 \end{bmatrix}, \tag{24}$$

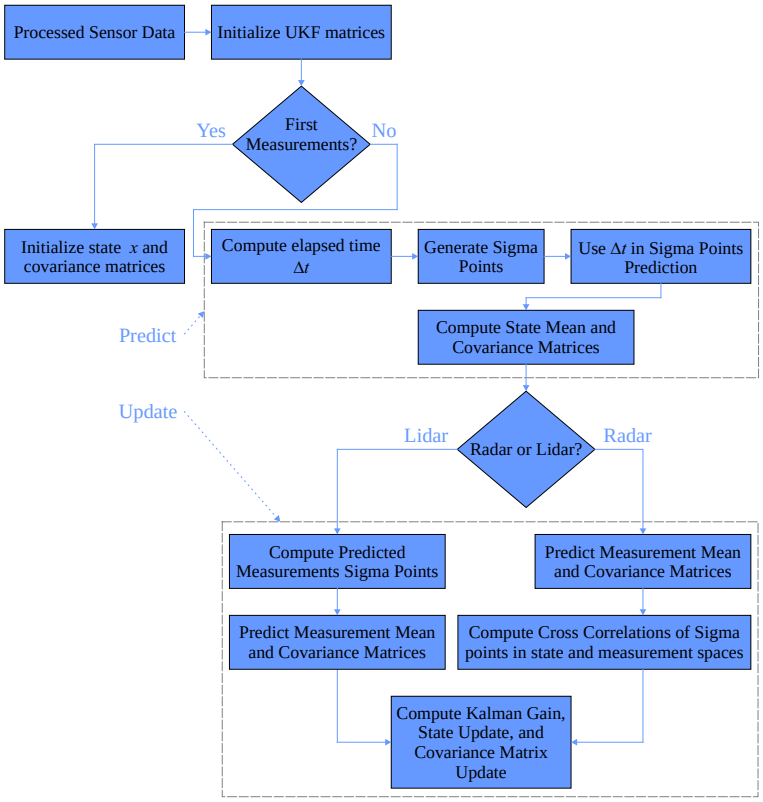

Fig. 4. Lidar and radar data fusion using UKF

where σ_{p_x} and σ_{p_y} are the noise standard deviations for the object x and y positions respectively. These matrices are required for the update step of the EKF (3).

The derived Jacobian matrix

$$H_{j_{\text{radar}}} = \frac{1}{|p|^2}\begin{bmatrix} p_x|p| & p_y|p| & 0 & 0 \\ -p_y & p_x & 0 & 0 \\ \dfrac{p_y v p_1}{|p|} & -\dfrac{p_x v p_1}{|p|} & p_2|p| & v p_1|p| \end{bmatrix}, \quad (25)$$

where: $|p| = \sqrt{p_x^2 + p_y^2}$, $p_1 = p_y \cos(\psi) - p_x \sin(\psi)$, $p_2 = p_x \cos(\psi) + p_y \sin(\psi)$. This represents the radar measurement model that will be used to estimate the measurement vector (19). The estimated measurement can be used in the update step of the EKF as illustrated in (3).

The measurement noise covariance matrix

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

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), $\sigma_{\dot{\rho}}$ is the noise standard deviation of the object yaw rate. This matrix is also required in the update step of the EKF.

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 (19) by inserting $(F_j, Q, H_{j_{\text{lidar}}}$, and $R_{\text{lidar}})$ in (2-3). Likewise, the state vector (x) is getting updated after receiving a radar measurement vector (z_{radar}) in (19) by inserting $(F_j, Q, H_{j_{\text{radar}}}$, and $R_{\text{radar}})$ in (2-3). Accordingly, the state vector (x) is the product of the fusion of then lidar and radar measurement data.

5 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 (7), and then a next-time-step prediction for sigma points $(\hat{X}_{k+1})$ is carried out using (8) while employing the moving object nonlinear motion model given in (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 (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 (20), the predicted measurement sigma points $(\hat{Z}_{k+1})$ are calculated

from the predicted state sigma points ($\hat{X}_{k+1}$) using (11). Then, the predicted measurements ($\hat{z}_{k+1}$) and their covariance matrix (S_{k+1}) are calculated based on (12) using the measurement noise covariance matrix R_{radar} , (26). Then, $\hat{x}_{k+1}$ and $\hat{z}_{k+1}$ and 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 ($\hat{Z}_{k+1}$), (13). Based on this cross-correlation matrix, the Kalman filter 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 (13).

If the measurement data source is a lidar, and employing the linear lidar measurement model (H_{lidar}) shown in (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 (12) using the measurement noise covariance matrix R_{lidar} given in (24). Then, $\hat{x}_{k+1}$ and $\hat{z}_{k+1}$ and 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 $\hat{Z}_{k+1}$), (13). Based on this cross-correlation matrix, the Kalman filter 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}), (13).

6 Implemantation of Kalman filter

Both Kalman filters (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 Jacobeans, object model and effectively performing the predict and update steps.

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

6.1 Noise parameters setting

The object motion model described by (14-17) includes several noise parameters that need to be 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 the expected top acceleration road objects can exhibit both longitudinal and angular as an initial guess, then fine-tune these values through trial-and-error iterations. After, exhaustive tuning process, the most appropriate valuation for σ_a and $\sigma_{\ddot{\psi}}$ are found to be 3 m/s^2 and 0.6 rad/s^2. Similarly, the other parameters can be set and accordingly, Table 1 presents the fine-tuned parameters for both EKF and UKF.

Hence, σ_a is set to 3 m/s^2 as $\{-6, 6\}$ m/s^2 is the expected interval of the longitudinal acceleration (*eg* 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/s^2.

Table 1. The Kalman filters 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	-
$\sigma_{\ddot{\psi}}$ rad/s^2	0.6	0.6
$\sigma_{\dot{\psi}}$ rad/s	0.06	0.06
σ_{p_x} (lidar) m	0.15	0.15
σ_{p_y} (lidar) m	0.15	0.15
σ_p (radar) m	0.3	0.3
σ_φ (radar) rad	0.03	0.03
$\sigma_{\dot{\rho}}$ (radar) m/s	0.3	0.3

6.2 Consistency measures of Kalman filters

The KF design is considered consistent if the estimation error, $\hat{y}_k$ in (3), is unbiased, *ie* 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 fine-tuned the noise parameters. The NIS follows a Chi-squared distribution (χ^2) and based on the number of the measurement-vector dimension; is used to assess whether KF's noise assumptions are consistent with the realized measurements. The metric, described by

$$\begin{aligned} NIS_k &= (z_{k+1} - \hat{z}_k)^\top S_k^{(} - 1)(z_{k+1} - \hat{z}_k), \\ NIS_{\text{Average}} &= \tfrac{1}{N} \sum_{k=1}^{k=N} NIS_k. \end{aligned} \tag{27}$$

It was 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.

The radar's measurement is a three-dimensional vector (three 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 (two 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} 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 2/3rd of their corresponding threshold ranges.

Table 2. Initialization of Kalman filter states

Parameter	EKF	UKF
p_x m	1st raw x-reading	1st raw x-reading
p_y m	1st raw y-reading	1st 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
$\sigma_{\hat{v}}$ m/s	$\sqrt{1000}$	$\sqrt{1000}$
$\sigma_{\hat{\psi}}$ rad	$\sqrt{1000}$	$\sqrt{1000}$
$\sigma_{\hat{\dot{\psi}}}$ m/s^2	$\sqrt{1000}$	$\sqrt{1000}$

Table 3. Performance of Kalman filter

Test case	State variable	RMSE EKF	RMSE UKF	Improvement (%)
Biccicle	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

6.3 Initialization of Kalman filters

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

$$P = \begin{bmatrix} \sigma^2_{\hat{p}_x} & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{\hat{p}_y} & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\hat{p}_v} & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\hat{p}_\psi} & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\hat{p}_{\dot{\psi}}} \end{bmatrix}. \tag{28}$$

The first two terms of the state vector x given by (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 Tab. 2.

The state covariance matrix is initialized as a diagonal matrix that contains the covariance of each variable estimate (Eq. (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 (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 $\sigma^2_{p_x}$ and $\sigma^2_{p_y}$ 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 Tab. 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.

6.4 Performance measures of Kalman filters

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

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{k=N}\left(x_k^{\text{est}} - x_k^{\text{true}}\right)^2}. \tag{29}$$

The metric is calculated over a moving window of measurements of length N, [41]. Here, x_k^{est} is the estimated state vector of the KF given in (4), and x_k^{true} is the true state vector supplied by the simulator or given as training data during the KF design phase.

7 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 (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 Fig. 5 to Fig. 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 (29)) is

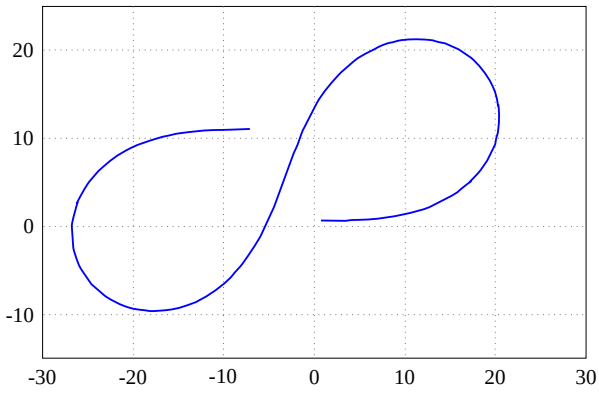

Fig. 5. Test track of a moving bicycle

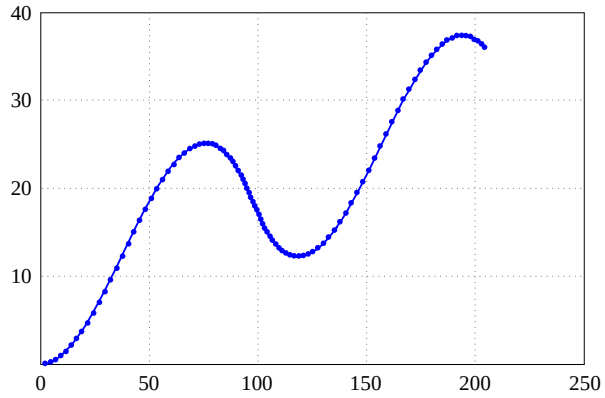

Fig. 6. Test track of a moving carbicycle

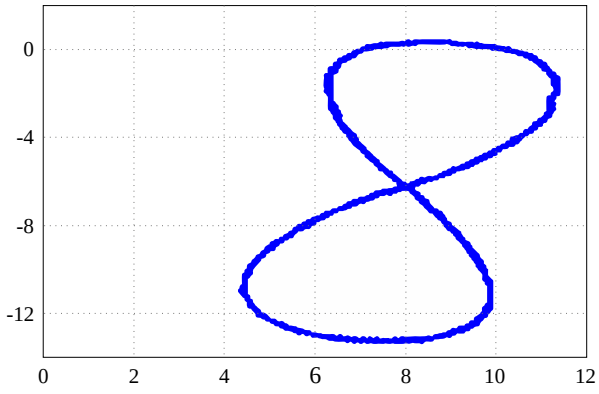

Fig. 7. Test track of a walking pedestrian

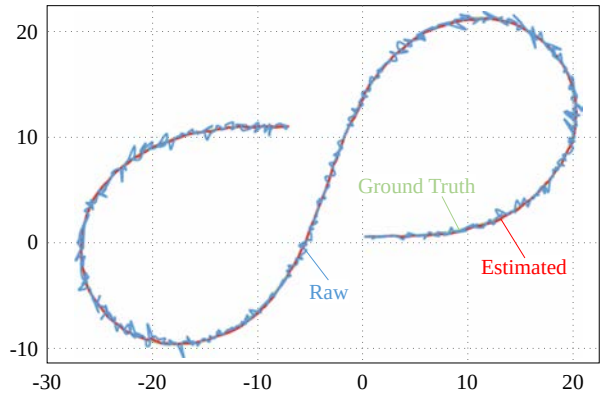

Fig. 8. Performance of the UKF position estimation on the bicycle track

Table 4. Consistency evaluation of Kalman filters

	NIS average	NIS min	NIS max	NIS ¿ 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

Table 5. Fusion evaluation of the UKF (sensor 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%

used to compare both UKF and EKF performances on the five state variables: $p_x, p_y, v_x, v_y, and?$. 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 Fig. 8), longitudinal velocity (shown in Fig. 9), yaw angle (shown in Fig. 11), and yaw rate (shown in Fig. 12). The bicycle tracking results using the EKF are reported in the third column of Tab. 3 and the estimated longitudinal velocity profile is depicted in Fig. 10. The UKF design-consistency indicator (NIS) results are reported in Tab. 3. The table reports values for the 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 5th 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 Tab. 5 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

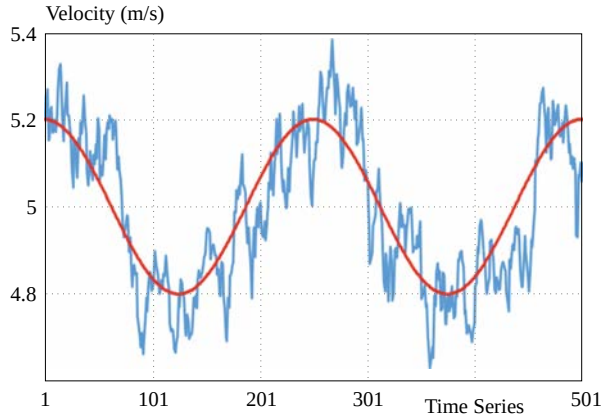

Fig. 9. UKF-velocity-estimation performance on the bicycle track

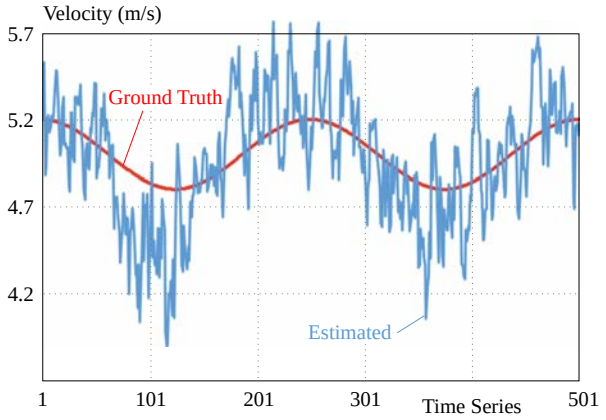

Fig. 10. EKF-velocity-estimation performance on the bicycle track

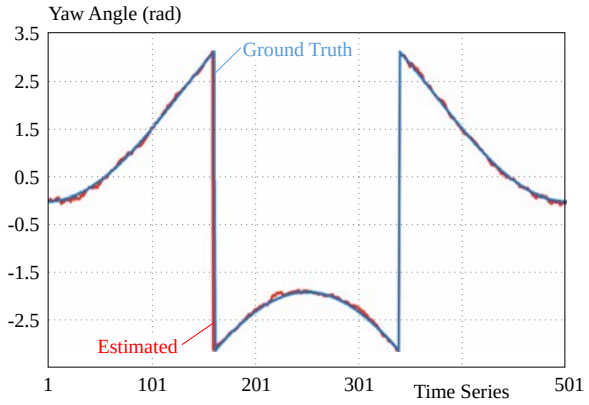

Fig. 11. UKF-yaw-angle estimation performance on the bicycle track

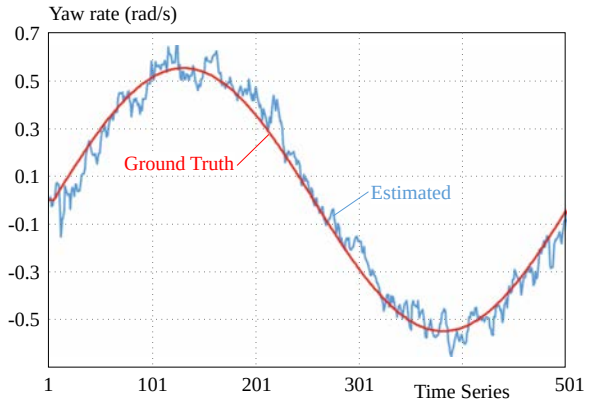

Fig. 12. UKF-yaw-rate estimation performance on the bicycle track

Table 6. Sensor fusion execution time of EKF and UKF

Phase	EKF- μs	UKF - μs
Predict	4.827	27.32
Update	16.40	14.20
Total	21.23	41.52

cases to test the consistency of the UKF in their cases. The reported values 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, Tab. 3 details the results. It is obvious that 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 nonlinearities in the object model (*eg* v_x, v_y, and ψ) have seen more accuracy improvement. Comparing the velocity profiles in both Fig. 9 and Fig. 10 gives some insight into the achieved improvement.

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 (Tab. 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. Then the measurement cycle is 33.3 ms which is large enough to be utilized for tracking 25 objects using EKF or 13 objects using UKF according to the data in Tab. 7.

8 Conclusion

In this paper, 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 Kalman filter 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 objects 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.

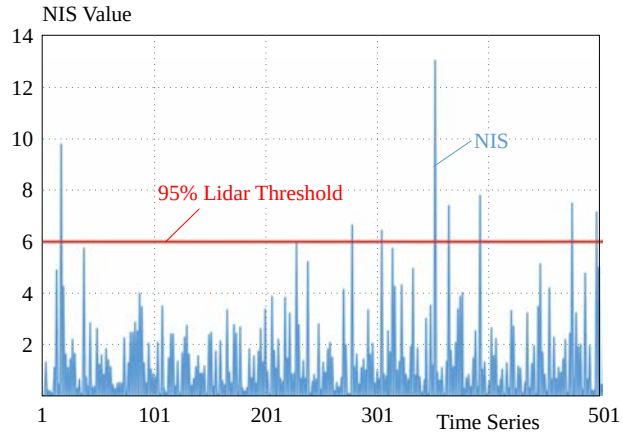

Fig. 13. Lidar NIS values for UKF on the bicycle track

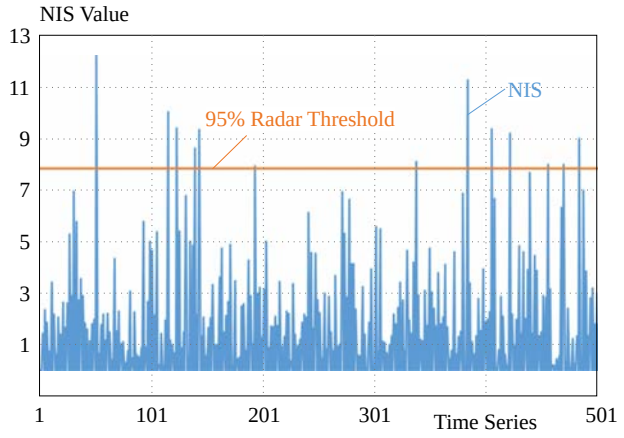

Fig. 14. Radar NIS values for UKF on the bicycle track

Table 7. LR_ODT execution time for single object

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

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, show how outstanding is the improvement in tracking performance using the later (-29% RMES 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 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, animals, *etc.*

References

[1] W. Farag, "Traffic signs classification by deep learning for advanced driving assistance systems", *Intelligent Decision Technologies*, IOS Press, vol. 13, no.3, pp. 215-231, (2019).

[2] W. Farag and Z. Saleh, "Road Lane-Lines Detection Real-Time for Advanced Driving Assistance Systems", *Intern.Conf.on Innovation and Intelligence for Informatics, Computing, and Technologies* (3ICT'18), Bahrain, 18-20 November , (2018).

[3] W. Farag, "A Comprehensive Real-Time Road-Lanes Tracking Technique for Autonomous Driving", *International Journal of Computing and Digital Systems (IJCDS)*, vol. 9 (3), pp. 349-362, (2020).

[4] W. Farag and Z. Saleh, "Behavior Cloning for Autonomous Driving using Convolutional Neural Networks", *Intern. Conf. on Innovation and Intelligence for Informatics*, Computing, and Technologies (3ICT'18), Bahrain, 18-20 November , (2018).

[5] W. Farag, "Recognition of traffic signs by convolutional neural nets for self-driving vehicles", *International Journal of Knowledge-based and Intelligent Engineering Systems*, IOS Press, vol. 22, no: 3, pp. 205 214, (2018).

[6] W. Farag and Z. Saleh, "Tuning of PID Track Followers for Autonomous Driving", *Intern.Conf.on Innovation and Intelligence for Informatics, Computing, and Technologies* (3ICT'18), Bahrain, 18-20 November , (2018).

[7] W. Farag, "Complex Trajectory Tracking Using PID Control for Autonomous Driving", *International Journal of Intelligent Transportation Systems Research*, Springer, September , (2019).

[8] W. Farag, "Complex-Track Following Real-Time Using Model -Based Predictive Control", *International Journal of Intelligent Transportation Systems Research*, Springer, June, (2020).

[9] W. Farag and Z.Saleh, "An Advanced Road-Lanes Finding Scheme for Self-Driving Cars", *2nd Smart Cities Symposium* (SCS'19), IET Digital Library, Bahrain, 24-26 March, (2019).

[10] W. Farag, "Safe-driving cloning by deep learning for autonomous cars", *International Journal of Advanced Mechatronic Systems*, Inderscience Publishers, vol. 7, no.6, pp. 390-397, (2019).

[11] W. Farag, "Cloning Safe Driving Behavior for Self-Driving Cars using Convolutional Neural Networks", *Recent Patents on Computer Science*, Bentham Science Publishers, The Netherlands, vol. 12, no. 2, pp. 120-127(8), (2019).

[12] W. Farag and Z. Saleh, "An Advanced Vehicle Detection and Tracking Scheme for Self-Driving Cars", *2nd Smart Cities Symposium (SCS19)*, IET Digital Library, Bahrain, 24-26 March, (2019).

[13] E. Yurtsever, J. Lambert, A. Carballo and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies, ", , rXiv:1906.05113v2 [cs.RO] 6 January 2020.

[14] E. Che, J. Jung, M. J. And and Olsen, "Object Recognition, Segmentation, and Classification of Mobile Laser Scanning Point Clouds: A State of the Art Review", *Sensors 2019*, 19, 810; DOI:10.3390/s19040810.

[15] Y. Xie, J. Tian and X. Zhu, "A Review of Point Cloud Semantic Segmentation, ", , rXiv:1908.08854v2[cs.CV] 3 Sep 2019.

[16] P. Lidman and S. Luu, "Clustering, shape extraction and velocity estimation applied to radar detections", *M. Sc. thesis*, Dept.of Elect.Eng., Chalmers University of Technology, Gothenburg, Sweden 2018.

[17] W. Farag, "Real-Time Detection of Road Lane-Lines for Autonomous Driving", *Recent Advances Computer Science and Communications*, Betham Science, pp. 265 274, vol. 13-2, 2020.

[18] W. Farag, "A Comprehensive Vehicle-Detection-and-Tracking Technique for Autonomous Driving", *International Journal of Computing and Digital Systems (IJCDS)*, vol. 9 (4), pp. 567-580, (2020).

[19] K. Dietmayer, D. Kellner and J. Klappstein, "Grid-based dbscan for clustering extended objects radar data", *IEEE Intelligent Vehicles Symposium*, Alcala de Henares, Spain, June 2012.

[20] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Comm. ACM*, 24(6): 381395, June 1981.

[21] D. Gohring, M. Wang, M. Schnurmacher and T. Ganjineh, "Radar/lidar sensor fusion for car-following on highways", *IEEE Int. Conf. Autom. Robot. Appl.*, pp. 407412, 2011.

[22] N. Kaempchen, K. C. Fuerstenberg, A. G. Skibicki and K. C. Dietmayer, "Sensor fusion for multiple automotive active safety and comfort applications", J. Valldorf and W. Gessner, (editors), *Advanced Microsystems for Automotive Applications*, vol. 2, pages 137163.Springer, 2004.

[23] P. Zarchan and H. Musoff, "Fundamentals of Kalman Filtering: A Practical Approach", *American Institute of Aeronautics and Astronautics*, Incorporated, 4th Ed., ISBN978-1-62410-276-9, 2013.

[24] R. O. Chavez-Garcia and O. Aycard, "Multiple Sensor Fusion and Classification for Moving Object Detection and Tracking", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no.2, pp. 252534, 2015.

[25] A. Rangesh, and M. M. Trivedi, "No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles Using Cameras and LiDARs", *IEEE Transactions on Intelligent Vehicles*, Vol. 4, no. 4, December 2019.

[26] H. Hajri, and M.-C. Rahal, "Real-Time Lidar and Radar High-Level Fusion for Obstacle Detection and Tracking with evaluation on a ground truth", *arXiv:1807. 11264v2[cs. RO]*, (July 2019).

[27] B. S. Jahromi, T. Tulabandhula and S. Cetin, "Real-Time Hybrid Multi-Sensor Fusion Framework for Perception Autonomous Vehicles", *Sensors*, 19(20), 4357, (2019), DOI:10.3390 /s19204357.

[28] E. A. Wan, and R. Van der Merwe, "The unscented Kalman filter for nonlinear estimation", *IEEE Adaptive Sys. for Signal Processing*, Comm., and Control Symposium, Alberta, Canada, (Oct.2000).

[29] S. J. Julierx and J. K. Uhlmann, "Unscented filtering and nonlinear estimation", *Proceedings of the IEEE*, 92(3): 401422, (2004).

[30] G. A. Einicke and L. B. White, "Robust Extended Kalman Filtering", *IEEE Trans.Signal Process.*, 47(9): 25962599, (September 1999).

[31] M. C. Best and K. Bogdanski, "Extending the Kalman filter for structured identification of linear and nonlinear systems", *Int. J. Modelling*, Identification, and Control, vol. 27, no. 2, (2017).

[32] R. Schubert, E. Richter and G. Wanielik, "Comparison and Evaluation of Advanced Motion Models for Vehicle Tracking", *11th Inter. Conf. on Information Fusion*, Cologne, Germany, (July 2008).

[33] J. Sander, X. Xu, M. Ester and H.-P. Kriegel, "A density-based algorithm for discovering clusters large spatial databases with noise", *Proc. of the 2nd Inter. Conf. on Knowledge Discovery and Data Mining*, pp. 226231, (August 1996).

[34] GCC C++ https://gcc.gnu.org/, accessed on 11th March, (2020).

[35] Ubuntu Linux https://www.ubuntu.com/, accessed on 11th March, (2020).

[36] M. Nagiub and W. Farag, "Automatic selection of compiler options using genetic techniques for embedded software design", *IEEE 14th Inter. Symposium on Comp. Intelligence and Informatics (CINTI)*, Budapest, Hungary, November 19, (2013).

[37] Eigen http://eigen.tuxfamily.org/index.php?title=Main_Page, accessed on 11th March, (2020).

[38] R. Pich, "Online tests of Kalman filter consistency. International Journal of Adaptive Control and Signal Processing", 30(1), pp. 115124, (2016).

[39] S. Zhao, and B. Huang, "On Initialization of the Kalman Filter", *6th Inter. Symposium on Adv. Control of Ind. Processes (AdCONIP)*, Taipei, Taiwan, May 28-31, (2017).

[40] K. Saho, "Kalman Filter for Moving Object Tracking: Performance Analysis and Filter Design", *IntechOpen*, Rijeka, (2018).DOI: 10.5772/intechopen.71731.

[41] W. Farag, "Synthesis of intelligent hybrid systems for modeling and control", *PhD Thesis*, University of Waterloo, Canada, (1998).

Received 8 May 2020

Wael Farag gained his PhD from the University of Waterloo, Canada in 1998; MSc from the University of Saskatchewan, Canada in 1994; and BSc from Cairo University, Egypt in 1990. His research, teaching and industrial experience focus on embedded systems, mechatronics, autonomous vehicles, renewable energy, and control systems. He has combined 17 years of industrial and senior management experience in Automotive (Valeo), Oil & Gas (Schneider) and Construction Machines (CNH) positioned in several countries including Canada, USA & Egypt. Moreover, he has 10 Years of academic experience at Wilfrid Laurier University, Cairo University, and the American University of the Middle East. Spanning several topics of electrical and computer engineering. He is the holder of 2 US patents; ISO9000 Lead Auditor Certified and Scrum Master Certified.

Received 8 May 2020