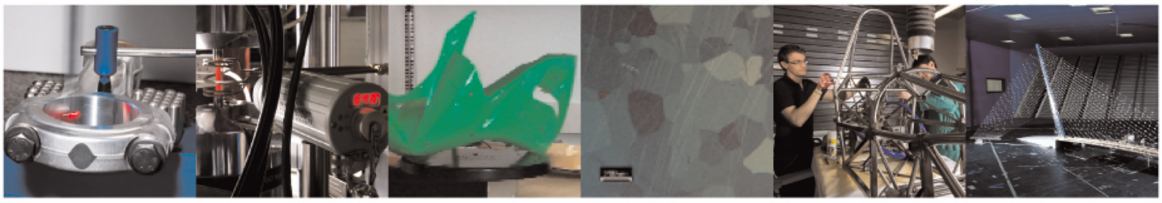


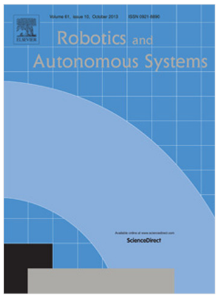
An integrated algorithm for ego-vehicle and obstacles state estimation for autonomous driving

Mattia Bersani, Simone Mentasti, Pragyan Dahal, Stefano Arrigoni, Michele Vignati, Federico Cheli, Matteo Matteucci

This is a post-peer-review, pre-copyedit version of an article published in *Robotics and Autonomous Systems*. The final authenticated version is available online at: http://dx.doi.org/10.1016/j.robot.2020.103662

DIPARTIMENTO DI MECCANICA ■ POLITECNICO DI MILANO
via G. La Masa, 1 ■ 20156 Milano ■ EMAIL (PEC): pecmecc@cert.polimi.it
http://www.mecc.polimi.it
Rev. 0

Journal Pre-proof



An integrated algorithm for ego-vehicle and obstacles state estimation for autonomous driving

Mattia Bersani, Simone Mentasti, Pragyan Dahal, Stefano Arrigoni, Michele Vignati, Federico Cheli, Matteo Matteucci

PII: S0921-8890(20)30502-9
DOI: https://doi.org/10.1016/j.robot.2020.103662
Reference: ROBOT 103662

To appear in: *Robotics and Autonomous Systems*

Please cite this article as: M. Bersani, S. Mentasti, P. Dahal et al., An integrated algorithm for ego-vehicle and obstacles state estimation for autonomous driving, *Robotics and Autonomous Systems* (2020), doi: https://doi.org/10.1016/j.robot.2020.103662.

An integrated algorithm for ego-vehicle and obstacles state estimation for autonomous driving

Mattia Bersani[a,*], Simone Mentasti[b], Pragyan Dahal[a], Stefano Arrigoni[a], Michele Vignati[a], Federico Cheli[a], Matteo Matteucci[b]

[a] *Department of Mechanical Engineering, Politecnico di Milano, Italy*
[b] *Department of Electronics, Information Engineering and Bioengineering, Politecnico di Milano, Italy*

Abstract

Understanding of the driving scenario represents a necessary condition for autonomous driving. Within the control routine of an autonomous vehicle, it represents the preliminary step for the motion planning system. Estimation algorithms hence need to handle a considerable number of information coming from multiple sensors, to provide estimates regarding the motion of ego-vehicle and surrounding obstacles. Furthermore, tracking is crucial in obstacles state estimation, because it ensures obstacles recognition during time. This paper presents an integrated algorithm for the estimation of ego-vehicle and obstacles' positioning and motion along a given road, modeled in curvilinear coordinates. Sensor fusion deals with information coming from two Radars and a Lidar to identify and track obstacles. The algorithm has been validated through experimental tests carried on a prototype of an autonomous vehicle.

Keywords: Obstacles tracking, Sensor fusion, State estimation, Autonomous driving

---

1. Introduction

State estimation represents an essential part of the control routine of an autonomous vehicle. Together with the behavioral layer and the higher-level route planner, it provides the initial and boundary conditions for the motion planning system, and it feeds information to the trajectory planner and the low-level trajectory follower, which actuates the vehicle [1]. Initial and boundary conditions (IC, BC) are usually provided in terms of road geometry, limitations given by regulations, ego-vehicle and obstacles current positions, and velocities. This overall architecture is schematized in the control loop presented in Fig.1.

The importance of vehicle state estimation has increased starting from 90s, when it became a fundamental task for the incoming active safety systems like

*corresponding author
Email address: mattia.bersani@polimi.it (Mattia Bersani)

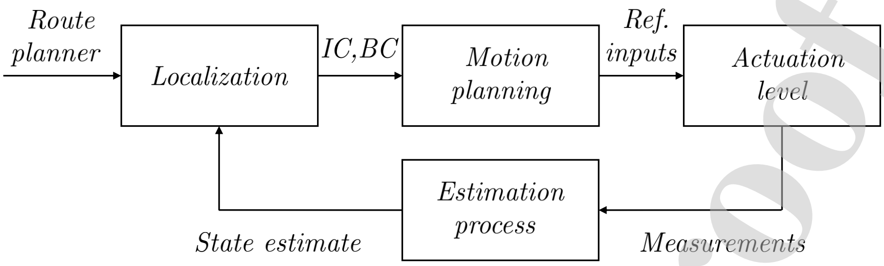


Figure 1. Representation of the control loop for an autonomous vehicle

ABS and ESP [2]. The measurements of yaw rate, wheels rotational speeds, and oil pressure in the brake circuit were provided to ensure the feedback control for vehicle handling. Moreover, kinematic quantities like vehicle sideslip were estimated to account for saturation effects in the tire contact patch. However, an autonomous vehicle's control routine requires further information about its relative motion with respect to road bounds and other vehicles.

For these reasons, one of the biggest challenges for the development and deployment of autonomous driving has been understanding the environment it operates in, which is extremely dynamic and uncertain. Various perception sensors have been developed and then used for this scope: ranging from stand-alone ones to full-suites, allowing localizing and perceiving the environment around the vehicle. Devices like Radars, Lidars, and cameras are very popular in this field, even though they provide different performances and information in terms of perception. Hence, various cost-effective combinations of sensor suites have been proposed to perceive the surrounding environment. The use of Radars [3], stereo cameras [4, 5] and Lidars [6] as stand-alone sensors has been done in the past for obstacle state estimation. Numerous studies have then been conducted based on the fusion of information coming from multiple sensors: camera, Lidar, and Radar [7, 8, 9], Radar and Lidar [10, 11]. Each of those sensors can provide heterogeneous information with different accuracy levels, which explains why they are usually combined to provide a fused representation of the environment. Among them, Radar is considered the most accurate sensor for what concerns the measurement of velocities as it exploits the Doppler effect. About positioning, the accuracy of Lidar measurements are considered better [10], while object classification is usually performed by cameras thanks to the high semantic content they provide[12].

Perception involves two major tasks: Simultaneous Localization and Mapping (SLAM) and Detection and Tracking of Moving Objects (DATMO). SLAM allows the map generation around the ego-vehicle while it simultaneously localizes itself through the sensor measurements. DATMO requires the ego-vehicle to detect any obstacle within the road bounds and keeps track of them in time, enabling the control system to account for each one's behavior within the current driving scenario. This must be guaranteed even during sensors malfunctioning, lack of sensors measurements due to asynchronous time sampling, abnormal weather conditions, occlusions, and any other circumstance leading to missing measurements that could cause blackouts. Hence, the estimation routine has to guarantee that this lack of information does not induce the motion planner to make wrong decisions.

Moreover, a proper modeling of the road environment close to the autonomous vehicle, besides allowing navigation, guarantees an efficient prediction of the relative positioning with respect to pedestrians, bicycles, other vehicles, and road bounds. The most common road definition models are: poly-line model, lane-let model, and Hermite spline model with increasing complexity and computational need in given order [13]. According to the different motion planners presented in [14, 15], the road map model of the track can be approximated through cubic Hermite spline interpolation [16]. The most important advantage of curvilinear coordinates $(s - n)$ with respect to Cartesian coordinates $(X - Y)$ is that each road characteristic can be described as a function of only one parameter (i.e., the abscissa s); thus, each function that approximates the centerline is at least surjective.

This paper focuses on state estimation for autonomous driving and presents an integrated algorithm that provides state estimates for the ego-vehicle and the surrounding obstacles. For the former, information about positioning, heading angle, and velocity of the vehicle itself are provided by two GPS receivers, inertial units, and odometry. About the latter, measurements are provided by a multi-sensor framework, which includes two Radars located within the vehicle front and rear bumpers and a Lidar mounted on the vehicle's top in correspondence of the center of gravity. Information about the surroundings is fused and provided to the tracking routine, according to DATMO. Exploiting the knowledge of the road map, the ego-vehicle is localized along the track within the road's local reference frame, from which the relative positioning and motion of each obstacle can be derived in curvilinear coordinates as shown in Fig. 2. Throughout this work, information about road boundaries and road shape is considered as known. This integrated algorithm has been implemented on the prototype of an autonomous vehicle presented in [17], and it has been validated through experimental tests carried in the Monza Eni Circuit [18]. The algorithm works at $20\,Hz$ on a soft real-time system based on ROS (Robot Operating System) [19], which allows dealing with asynchronous sensors.

This paper is articulated through the following sections: the state of the art on sensor fusion, state estimation and DATMO is reported in Section 2 while the general structure of the algorithm is presented in Section 3. Section 4 presents the ego-vehicle state estimation procedure, while obstacles state estimation and tracking are described in Sections 5 and 6. The validation of the estimation procedure is given in Section 7, where the experimental framework is presented together with results.

2. Related works

As stated in the previous section, DATMO can provide the estimate for each obstacle close to a vehicle even in uncertain conditions. Measurements filtering,

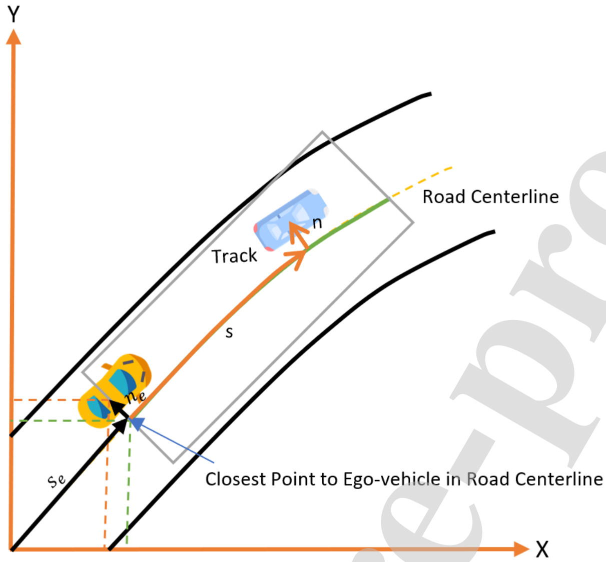


Figure 2. Coordinates transformation

data association, and data fusion are the main tasks for a multi-sensors and multi-objects estimation problem: this section presents a summary of the state of the art for each of them.

2.1. Filtering techniques

The measurements obtained from sensors in a real scenario are affected by noise and a high degree of uncertainty. Hence, filtering procedures are usually applied to ensure accurate estimates and tracking. Bayesian filters are widespread in literature: these filters exploit the Chapman-Kolmogorov theorem through the system transition density to achieve predicted probability density functions (PDF) for the objects under consideration. Measurements are then used to update the predicted PDF to find the posterior PDF, from which the estimates can be obtained. Prediction and update steps in Bayesian filtering involve complicated integrals that lead to a high computational burden. When the model of the observed system is linear, and noise is Gaussian distributed, the integrals can be computed analytically: in these conditions, Kalman filtering [3, 5, 12, 20] provides the optimal solution, that can also be derived through the minimization of the mean squared error. However, if the system behavior is nonlinear, Extended Kalman Filter (EKF) [7, 8] and Unscented Kalman Filter (UKF) [21, 22] are favored solutions. In particular, when nonlinearities become huge, EKF provides less accurate solutions due to the first-order linearization of the system's equations through Taylor-series expansion. Conversely, UKF is based on the so-called unscented transformation, which approximately provides

Gaussian distributed outputs even when dealing with nonlinear transformations. Particle filters or Sequential Monte Carlo (SMC) filters are other variants of Bayesian Filters that can be used for nonlinear systems and non-Gaussian Noise Distributions. As the name suggests, they use weighted particles, each represented by possible state estimation and posterior distribution. The usage of Random Finite Set (RFS) statistics is common in Multi-Object-Tracking (MOT). In particular, RFS enables MOT without a priori measurement association through the implementation of recursive Bayes filtering. When dealing with scenarios in which the birth and death of objects are regular, with a significant amount of clutter and false positives, the association process provided by traditional Bayes filters leads to erroneous results. Conversely, RFS allows accounting for objects birth (regular or spawning), occlusions, misdetections, and disappearances by taking the number of objects under consideration as a stochastic variable. Gaussian Mean-Probability Hypotheses Density (GM-PHD) Filter [23], Multi-Bernoulli Mixture (MBM) Filter, Poisson Multi-Bernoulli Mixture (PMBM) Filter, etc. are other filters adopted in the literature. A detailed study for these filters is presented in [24].

2.2. Pointcloud elaboration

As anticipated in the previous section, throughout this work, multi-sensors data fusion is considered to be done between two Radars and a Lidar. Contrary to the general case, Radars used in this work already provide preprocessed clustered point detections coming from an object. The processing for the 3D pointcloud coming from a Lidar sensor represents a more complex task. Referring to the robotics literature, obstacle detection from 3D pointcloud can be provided through a map-based approach or with deep learning-based techniques.

Authors in [25] implemented an occupancy grid for the space surrounding a robot in which each cell is labeled as empty or occupied. Scenarios with a large number of sensors usually employ multilayers based solution [26, 27], where each sensor provides a different occupancy grid, then fused to retrieve a representation of the environment. Other scenarios, where the terrain presents significant changes in height, require instead using more complex maps, which also considers changes in elevation [28, 29]. More straightforward solutions, based on 2.5D maps [30, 31], merge the reduced dimensions and limited computational requirements of a 2D grid with the height of the 3D approach. Recent approaches, specially designed for autonomous driving scenarios, also implement combinations of 2D and 3D based processing [32] using the original pointcloud to label the obstacles and the grid to perform planning. In general, most of the classification oriented systems prefer 3D pointcloud to identify and label the obstacles [33, 34, 35].

In the last years, a different approach to pointcloud elaboration has emerged, the usage of deep learning techniques, particularly Convolutional Neural Network (CNN). The most successful solutions in the autonomous driving field do not try to label each pixel of the pointcloud but predict 3D bounding box around obstacles [36], [37], [38]. This guarantees low processing time and the ability to run in real-time. Nevertheless some approaches of 3D points semantic segmentation, mainly based on the PointNet [39] and PointNet++ [40] architecture, has emerged. Lastly, hybrid approaches, which combine the usage of occupancy grids and CNN has been proposed [41] to reduce the required computational power. Those solutions first reduce the pointcloud to a 2D occupancy grid and then process it with a classical 2D neural network; in such a way, the input size is considerably reduced compared to the original 3D pointcloud, and it can be processed much faster. The main disadvantage of those solutions is the need for dense pointcloud to feed the network with feature-rich data. This is possible with 32 and 64 planes Lidars. Still, with smaller sensors, with fewer planes, the obstacles become less defined, and the networks are generally not able to extract enough features to detect the obstacles, as shown in [42].

2.3. Sensor fusion

Data Association is one of the crucial steps in MOT problems. A critical assumption for this task, among others, is that the number of objects (n) is not a random variable, but it is considered as known during each filtering iteration. Global Nearest Neighbour (GNN) filters, Joint Probabilistic Data Association (JPDA) filters, and Multi Hypothesis Tracking (MHT) filters are the most commonly adopted approaches in MOT. These filters are presented in detail in [24, 43]. Kalman filtering and its advanced versions (EKF and UKF) are usually employed to ensure objects tracking. GNN filters perform association of measurements and estimates under the best association hypotheses (i.e., the one with the lowest association cost is considered while others are pruned). Although computationally cheap and fairly accurate in case of high Signal to Noise Ratio (SNR), performances can degrade in moderate or low SNR. However, JPDA considers a certain number of best assignment hypotheses for the association and computes marginal posterior densities with corresponding marginal association probability. Weighted merging of these posterior densities is done to extract the estimated state. With increased computational burden, JPDA performs better in low to medium SNR scenarios compared to GNN. MHT filter requires calculating a pre-defined number of best association hypotheses while pruning all others: in this way, posterior densities retains a certain number of most probable hypotheses. This allows for corrections in previous association decisions when new information from sensors is given.

Multi-sensors data fusion for autonomous driving can be described as centralized, decentralized, or hybrid architectures [44]. In centralized data fusion, also referenced as central level fusion, the sensors' raw data are minimally pre-processed at sensor level and then forwarded to be fused in the central module. Object discrimination and tracking are handled at central level. In decentralized data fusion, each sensor is tasked to identify and track objects. Fusion of these tracks is done in a centralized module and may involve feedback to the sensor module. Hybrid data fusion architectures are a combination of previous approaches. Two sets of information are conveyed from the sensor module: minimally pre-processed data to the central module and simultaneously tracks to decentralized fusion modules. The outputs of the decentralized modules are fed again to the central module for fusion purposes.

2.4. State estimation

In obstacles state estimation, algorithms have to deal with a large number of measurements collected by sensors. Hence, any possibility of filtering in advance any unwanted noise or false positives may help in reducing the computational burden. For this reason, the filtering process can be improved by exploiting the knowledge of road bounds leading to higher estimates accuracy [3, 13, 20]. In particular, authors in [20] have studied the possibility of providing estimates in curvilinear coordinates by tracking fusion and behavioral reasoning of obstacles within the road bounds. As anticipated in the previous section, conversion from Cartesian to curvilinear coordinates can be beneficial in multiple aspects of autonomous driving but even for communication systems between different vehicles [13]. Authors in [3] have presented estimates in curvilinear coordinates to analyze obstacles motion close to the ego-vehicle. Estimation and tracking are given through decentralized fusion mode based on a Radar sensor, while nearest neighbor filter ensures track-measurement association. Obstacles state estimation is done in Cartesian coordinates and later converted into curvilinear ones through traditional Kalman filtering. However, this conversion process is highly nonlinear, so estimates can be vulnerable to faulty results.

As presented in [45], road definition adaptation in the estimation process has allowed for the development of a cooperative algorithm between two vehicles expediting their lane level localization. Authors in [46] have represented the status of ego-vehicle, objects, and traffic participants in road coordinates (i.e., Curvilinear Coordinates). This conversion enabled them to accelerate and simplify the trajectory planning of the ego vehicle. Knowledge of the road curvature and geometry allows the planning task to be performed in a simplified environment by eliminating the road curvature and performing the planning task in a straight line. This reduced the computational burden and time consumed for performing an optimization task. The planned motion is again interpreted in a road environment for defining the designated motion.

For what concerns the ego-vehicle positioning, GPS sensors with RTK correction systems are becoming widespread in autonomous and intelligent vehicles. These sensors can be equipped with 6-DOFs inertial units (IMU), ensuring a cheap setup for the inertial navigation system. Authors in [47, 48] integrate a GPS receiver in the estimation process based on a kinematic vehicle model. They demonstrate how these sensors can improve the estimate accuracy even for the vehicle lateral velocity. This consideration, applied to autonomous driving, allows avoiding a complex reverse engineering process to tune parameters like tire cornering stiffness and relaxation lengths, mass, and moment of inertia (at least along the vertical axis) [49, 50]. Moreover, a couple of GPS receivers can be installed on the same vehicle to provide an estimation of the absolute heading angle [48, 51]. Accuracy increases if the receivers are located on the longitudinal axis. About the vehicle motion, lateral velocity in the center of gravity (CoG) can be derived by kinematic relationships assuming pure rolling contact

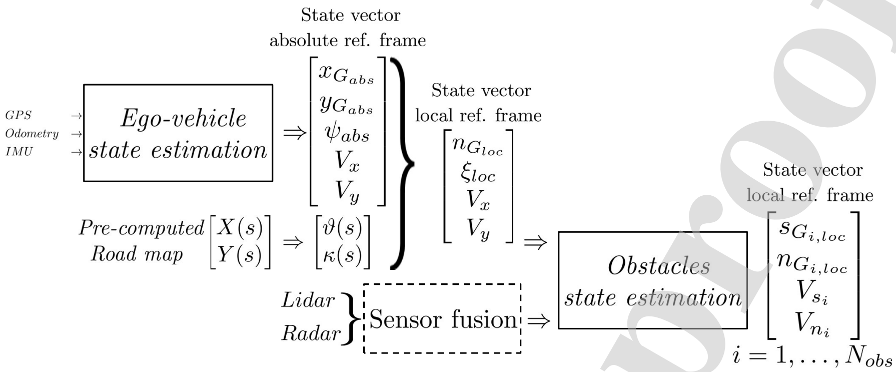


Figure 3. Scheme of the integrated estimation algorithm

and low longitudinal speed [52, 53]. Then, the estimated accuracy can be improved, accounting for vehicle lateral dynamics. In the literature, performances related to these two different modeling approaches have been compared many times [54, 55, 56, 57]. In general, dynamic and physical vehicle models ensure more accurate estimates as the vehicle speed increases, but a higher number of parameters must be tuned, and the computational cost increases. Authors in [51] implemented an EKF to provide positioning, heading angle, and lateral velocity for autonomous vehicles based on a kinematic single-track vehicle model. Authors in [58] compared performances between EKF and UKF for a similar estimation procedure. Results assess that UKF provides more accurate results, ensuring fast computational time.

Compared to the current state of the art, the presented work aims to estimate obstacles positioning and relative motion referenced to the ego-vehicle in curvilinear coordinates, which involves a highly non-linear measurement model. Hence, a UKF has been implemented as it represents a compromise between accuracy, computational effort, and ease of implementation.

3. Architecture of the estimation system

As stated in previous sections, the aim of the presented estimation system is to compute ego-vehicle and obstacles state vectors in curvilinear coordinates. As shown in Fig.3, measurements for ego-vehicle state estimation are given by GPS receivers, inertial units, and odometry. The estimation algorithm is then based on a UKF to provide vehicle positioning and heading angle in the global Cartesian reference frame. At the same time, longitudinal and lateral velocities are given according to the moving reference system centered with the vehicle (i.e., the vehicle reference frame, VRF).

$$x_{e,\,abs} = \begin{bmatrix} x_{G_{abs}} & y_{G_{abs}} & \psi_{abs} & V_x & V_y \end{bmatrix}^T \tag{1}$$

The G-subscript in (1) means that positioning is given in the vehicle center of gravity, which also represents the center of the moving reference system, in which velocities V_x and V_y are estimated. As stated in the previous section, the curvilinear framework provides many advantages compared to the Cartesian one when applied to autonomous driving. As shown in Fig. 2, through the pre-computed road map description it is possible to move from global coordinates to the local reference frame along the road centerline. Doing so, ego-vehicle positioning in (1) can be converted to curvilinear coordinates:

$$x_{e,\,loc} = \begin{bmatrix} n_{G_{loc}} & \xi_{loc} & V_x & V_y \end{bmatrix}^T \tag{2}$$

where $\xi_{loc} = \psi_{abs} - \theta_e$ represents the current relative heading direction of the ego-vehicle with respect to the road angle (θ_e). Once ego-vehicle positioning is computed with respect to the road centerline, the pre-computed road map provides the road description in terms of road angle and curvature ($\theta(s)$ and $\kappa(s)$, respectively) for the current local reference frame. The road description is given in terms of Hermite spline curves for the following $50\,m$, which corresponds to the overall estimation process's field of view (FoV). The state vector in (2) does not include the vehicle's absolute position along the track (i.e., the curvilinear abscissa s_G). Aiming to provide obstacles' positioning with respect to the ego-vehicle, this variable is not required because the vehicle is localized at any time step in a different local reference frame, to which the road map associates the corresponding road description.

Measurements of obstacles are provided by two Radar sensors and by a Lidar in VRF (i.e., the same one in which longitudinal and lateral velocities are given). For each tracked obstacle $i = 1, \ldots, N_{obs}$, state estimation (3) is provided in local reference frame in terms of longitudinal distance with respect to ego-vehicle $s_{i,\,loc}$ and lateral distance with respect to road centre line $n_{i,\,loc}$. Moreover, absolute velocities are given according to road tangential and orthogonal directions (V_{s_i} and V_{n_i}, respectively).

$$x_{o_i,\,loc} = \begin{bmatrix} s_{i,\,loc} & n_{i,loc} & V_{s_i} & V_{n_i} \end{bmatrix}^T \tag{3}$$

Throughout this work, the small objects assumption is adopted (i.e., an object is represented by a point, and its state is defined with positional and velocity values only, neglecting the orientation information). Hence, the relative orientation of obstacles with respect to the road is not included in the state vector. Even though an obstacle's orientation is an important information in the overall perception module, it is not considered in this application to speed up the implementation and ensure real-time. However, all the estimates are provided in the road reference, whose direction and limits are known in advance, and the algorithm computes magnitude and direction of the velocity's vector for each detected obstacle. Thus, if coupled together, these pieces of information can eventually provide a motion planner with an estimate of the obstacle's trajectory.

4. Ego-vehicle state estimation

Kalman filtering usage for vehicle state estimation, with EKF and UKF, is well-established to account for model nonlinearities. Furthermore, Kalman filtering requires a reasonable computational effort and allows managing different sampling frequencies from sensors: this guarantees that estimates can be provided even in a real-time control routine.

As stated in previous sections, ego-vehicle state estimation is provided in terms of positioning and velocity. According to the works presented in Section 2, a kinematic single-track vehicle model can be implemented within a range of speed typical of urban driving scenarios. Although a simple kinematic model guarantees fast implementation and interchangeability on different vehicles, the lack of accuracy can lead to estimation errors. These errors are mainly related to the lateral velocity estimation, which is strongly affected by tire cornering stiffness, geometry of the suspensions, saturation of friction in the tire contact patch, and load transfers. Even the vehicle's longitudinal dynamic is crucial when dealing with strong braking maneuvers that are very common, especially in the urban environment. Despite this, the estimate accuracy can be improved utilizing a GPS receiver with real-time kinematic (RTK) correction. In this way, the motion planning system will continuously receive precise and accurate estimates, at least in terms of positioning. Furthermore, including the vehicle's heading angle in (1) can lead to the motion planner to account for the car's mutual direction with respect to the road and other obstacles.

The discrete time definition of the UKF is based on the nonlinear systems of equations (4) and (5), where process disturbance w_k and measurement noise v_k are assumed to be additive and zero mean distributed with covariance matrices Q_k and R_k as indicated in (6).

$$x_k = x_{k-1} + f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})\delta t \tag{4}$$

$$y_k = h_k(x_k, v_k) \tag{5}$$

$$\begin{aligned} w_{k-1} &\sim (0, Q_{k-1}) \\ v_k &\sim (0, R_k) \end{aligned} \tag{6}$$

The system is modeled based on a kinematic single-track vehicle model, which considers the IMU measurements as input with included disturbances (7). These measurements are the longitudinal and lateral accelerations in the vehicle CoG ($a_{G,x}$ and $a_{G,y}$), and the yaw rate ω. The sensor bias is eliminated during the initialization phase when the vehicle is standstill.

$$f_{k-1} = \begin{cases} \dot{x}_G = V_x cos\psi - V_y sin\psi \\ \dot{y}_G = V_x sin\psi + V_y cos\psi \\ \dot{\psi} = \omega \\ \dot{V}_x = V_y\dot{\psi} + a_{x,G} \\ \dot{V}_y = -V_x\dot{\psi} + a_{y,G} \end{cases} \tag{7}$$

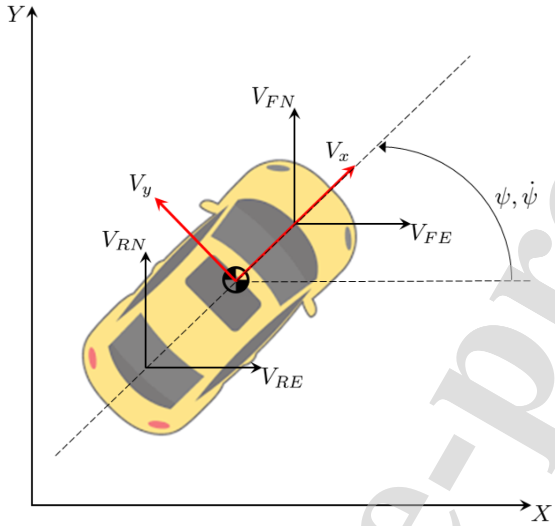


Figure 4. Representation of sensors orientation on the vehicle

The filter update equations integrate velocities and positions provided by the GPS receivers together with odometry (8). GPS measures velocities in the absolute reference system (ENU) through the Doppler effect, while odometry can be considered as given from exciters and encoders located on the ego-vehicle.

$$h_k = \begin{cases} V_{FE} = V_x cos\psi - (V_y + l_f \dot{\psi}) sin\psi \\ V_{FN} = V_x sin\psi + (V_y + l_f \dot{\psi}) cos\psi \\ V_{RE} = V_x cos\psi - (V_y - l_r \dot{\psi}) sin\psi \\ V_{RN} = V_x sin\psi + (V_y - l_r \dot{\psi}) cos\psi \\ V_{x,\, odom} = V_x \\ x_G = (x_F l_R + x_R l_F)/(l_F + l_R) \\ y_G = (y_F l_R + y_R l_F)/(l_F + l_R) \end{cases} \tag{8}$$

Parameters l_i and l_I, with $i \in [f,\, r]$ and $I \in [F,\, R]$ refer respectively to: distance between vehicle CoG and vehicle front and rear axis and distance between vehicle CoG and front and rear GPS receiver. Then, V_{FE}, V_{FN}, V_{RE}, and V_{RN} are the velocities in ENU coordinates measured by the GPS receivers, while $V_{x,\, odom}$ is the longitudinal speed of the vehicle given by odometry. The measurement model is represented by Fig. 4.

The unscented transformation (9) is applied to the estimated state vector

$\tilde{x}_{k-1}^{+}$ based on the state covariance matrix P_{k-1}^{+}.

$$\begin{aligned} \hat{x}_{k-1}^{(i)} &= \tilde{x}_{k-1}^{+} + \chi^{(i)} \quad i \in [1, \ldots, 2n] \\ \chi^{(i)} &= \left(\sqrt{nP_{k-1}^{+}}\right)_i^T \quad i \in [1, \ldots, n] \\ \chi^{(n+i)} &= -\left(\sqrt{nP_{k-1}^{+}}\right)_i^T \quad i \in [1, \ldots, n] \end{aligned} \tag{9}$$

The number of sigma points n can double the length of the state vector to speed-up calculations. Sigma points are passed through (4) to compute the matrix $\hat{x}_k^{(i)}$, which is used to evaluate the predicted state vector $\tilde{x}_k^{-}$ and covariance matrix P_k^{-} as indicated in (10) and (11). In both the equations, each sigma point is properly weighted through the parameter $w_i = 1/2n$.

$$\tilde{x}_k^{-} = \sum_{i=1}^{2n} w_i \hat{x}_k^{(i)} \tag{10}$$

$$P_k^{-} = \sum_{i=1}^{2n} w_i [\hat{x}_k^{(i)} - \tilde{x}_k^{-}][\hat{x}_k^{(i)} - \tilde{x}_k^{-}]^T + Q_{k-1} \tag{11}$$

A further unscented transformation (9) based on $\tilde{x}_k^{-}$ and P_k^{-} is required to evaluate a new set of sigma points ($\hat{x}_k^{(i)}$) to update the state vector prediction. This set of points is then propagated through the update equations of the filter (8) to calculate the predicted measurement matrix $\hat{y}_k^{(i)}$ from which the predicted measurements vector and the innovation covariance matrix P_y are evaluated according to (12) and (13).

$$\tilde{y}_k = \sum_{i=1}^{2n} w_i \hat{y}_k^{(i)} \tag{12}$$

$$P_y = \sum_{i=1}^{2n} w_i [\hat{y}_k^{(i)} - \tilde{y}_k][\hat{y}_k^{(i)} - \tilde{y}_k]^T + R_k \tag{13}$$

$$P_{xy} = \sum_{i=1}^{2n} w_i [\hat{x}_k^{(i)} - \tilde{x}_k^{-}][\hat{y}_k^{(i)} - \tilde{y}_k]^T \tag{14}$$

To conclude, the measurement update of the state estimates can be performed accounting for the cross covariance matrix given by (14), that is required to compute the Kalman gain matrix as indicated in (15). The updated state vector ($\tilde{x}_k^{+}$) and covariance (P_k^{+}) are obtained from equations (16) and (17).

$$K_k = P_{xy} P_y^{-1} \tag{15}$$
$$\tilde{x}_k^+ = \tilde{x}_k^- + K_k[y_k - \tilde{y}_k] \tag{16}$$
$$P_k^+ = P_k^- - K_k P_y K_k^T \tag{17}$$

The estimation process presented in this section provides the vehicle positioning and heading in global coordinates (i.e., in the absolute reference frame). Longitudinal and lateral velocities in the state vector (1) are given in the moving reference system centered with the vehicle CoG. Since positioning must be provided in the road local reference frame, as indicated by the state vector in (2), we have to solve an optimization problem before performing the estimation to position the vehicle within the road map. The pre-computed road map describes the road centerline in terms of heading and curvature in curvilinear coordinates with a discretization step of $ds = 0.5\,m$. A minimization algorithm computes the two smallest distances between the ego-vehicle position and each of the sampled map points through the euclidean norm. This brute force approach is performed only during the filter initialization phase: starting from the second iteration, a warm start is used to account for the previous vehicle position, to reduce the computational effort. Once the closest points are found, the algorithm computes the tangent to the road centerline close to the vehicle θ_e to provide the lateral position in this local reference frame $n_{G_{loc}}$ and the relative heading angle ξ_{loc}. Thus, the experimental setup required to provide initial conditions to a motion planner shall include a GPS receiver coupled with an inertial unit (IMU), an encoder on the steering shaft, and a couple of exciters for the measurement of the longitudinal vehicle's speed. Then, as explained in section 2, adding a GPS receiver located along the longitudinal axis of the car, it becomes possible to give an accurate estimate of the absolute heading angle.

5. Data processing and sensor fusion

The sensing architecture for obstacles state estimation consists of two Radar sensors mounted respectively on the front and the rear bumpers of the car, and a Lidar mounted on the roof. The *Continental ARS 408-21* long-range Radar sensor retains a 180°field of view in the horizontal plane, while a *Velodyne VLP-16* Lidar guarantees a 360°coverage.

5.1. Radar data

Data coming from the Radar sensors are already pre-processed and provided as clusters of detections in VRF. Those clusters give information on real objects and not single points. For each of them, the Radar measures the longitudinal and lateral distance in the VRF. Moreover, in the same reference frame, it also returns the longitudinal and lateral components of the relative velocity with respect to the ego-vehicle ($V_{x,rel}^{VRF}$ and $V_{y,rel}^{VRF}$ respectively). To compute these two velocities, the Radar sensors require the current ego-vehicle longitudinal speed and yaw rate, received through CAN-bus communication. Hence, the raw measurements available for each object can be summarized in the following:

$$y_{r\,o_i^{VRF}} = \begin{bmatrix} x_{P_i}^{VRF} & y_{P_i}^{VRF} & V_{x_i}^{VRF} & V_{y_i}^{VRF} \end{bmatrix}^T \quad (18)$$

where $V_{x_i}^{VRF}$ and $V_{y_i}^{VRF}$ are respectively the longitudinal and later components in the VRF of the $i-$object absolute velocity. These components can be computed as reported in (19) because Radar sensors evaluate each detected object's relative velocity with respect to the ego-vehicle accounting for the yaw rate of the VRF (i.e., accordingly to the relative motions theorem).

$$\begin{cases} V_{x_i}^{VRF} = V_x^{VRF} + V_{x,\,rel}^{VRF} \\ V_{y_i}^{VRF} = V_y^{VRF} + V_{y,\,rel}^{VRF} \end{cases} \quad (19)$$

For what concerns every single object's relative positioning, the measures can be considered related to the closest part of the leading (or following) vehicle. As indicated in (20), longitudinal and lateral distances from objects are derived accounting for the displacement between the sensors and the ego-vehicle CoG, i.e. l_{R_i} with $i \in [front, rear]$.

$$\begin{cases} x_{P_{i,\,front}}^{VRF} = x_{P_{i,\,front}} + l_{R_{front}} \\ x_{P_{i,\,rear}}^{VRF} = -x_{P_{i,\,rear}} - l_{R_{rear}} \end{cases} \quad (20)$$

A complete list of the provided data for each cluster identified by the Radar is provided in Table 1. As reported, the internal pre-processing of raw Radar detections guarantees not only objects measurements in VRF, but also tracking of the cluster in time, and an estimation of the related probability of existence and class.
Object tracking is already performed by the Radar sensor ($ID(o_i)$), but this information is not considered within the presented algorithm because it is strongly affected by noise. Nevertheless, the related probability of existence is used to filter out objects characterized by $p(o_i) \leq 99\%$. Indeed, some preliminary experimental tests demonstrated that lower probability measures are mostly due to misleading and false positive. For this reason, all clusters with a probability of existence lower than this threshold are removed. The filtering process is completed by clustering all the object detections within a pre-defined spatial threshold, whose value changes according to the object class indicated by the Radar sensor. During clustering, all measurements related to positioning and relative motion of each object are mediated between them. To conclude, this sensor provides an estimation of the standard deviation for each measurement of a given cluster ($\sigma_{\forall meas}$). This information is used during the following filtering process to account for the noise that affects measurements. During clustering, only the largest standard deviation for any different measurement is considered.

5.2. Lidar elaboration

Lidar measurements are provided as 3D pointclouds referenced to the sensor position, located in the ego-vehicle CoG. Thus, pointclouds processing is required to derive objects information in a similar form to that given by the Radar

data	description
$ID(o_i)$	ID of the tracked object
$x_{P_i}^{VRF}$	longitudinal distance in VRF
$y_{P_i}^{VRF}$	lateral distance in VRF
$V_{x,rel}^{VRF}$	longitudinal relative speed in VRF
$V_{y,rel}^{VRF}$	lateral relative speed in VRF
$\sigma_{\forall meas}$	standard deviation for all measurements
$Class(o_i)$	object typology (pedestrian, motorcycle, car)
$p(o_i)$	probability of existence

Table 1. Information provided by the Radar for each identified cluster

to ensure sensor fusion and state estimation. The low number of planes of the VLP-16 Lidar made impossible to implement one of the deep learning-based approaches presented in Section 2.2, due to the sparseness of the Pointcloud and the low number of features. Thus, a solution based on a 2D occupancy grid, similar to the one used in mobile robotics, has been adapted. Unlike the classical robotics scenarios, where the area of interest is limited to only a few meters around the robot, and the ground plane is generally flat. In this case, the obstacles can be at a high distance (i.e., 20 meters), move at high speed, and be as big as a truck. For all these reasons, we had to implement our pipeline for Lidar obstacle detection, leveraging on the classical occupancy grid approach, but adapting it to this new scenario. The pipelines in Fig. 5 and 6 show the operations required to convert a set of 3D points into a list of obstacles on the horizontal plane. In particular, this pipeline can be divided into two blocks: the first one concerns the conversion from 3D points to a 2D occupancy grid, while the latter deals with obstacle identification and tracking on the bi-dimensional grid.

The conversion of a 3D pointcloud into a 2D occupancy grid can be divided into some fundamental steps, shown in Fig. 5. The first block consists of the rotation of the pointcloud and ground plane fitting. The sensor is indeed mounted on the ego-vehicle roof with a slightly negative pitch to cover the frontal area better. Ground plane removal allows excluding all points belonging to the road surface to reject false positive. To perform this task, an approach similar to the one presented in [59] is implemented, in which the plane fitting problem is based on RANSAC (RANdom SAmple Consensus). An initial guess for the normal direction to the horizontal plane can be derived in standstill conditions by measuring the projection of the acceleration of gravity along each dimension of the triaxial accelerometer of the vehicle's inertial unit (IMU); while the distance of the plane has been previously measured in a controlled environment. During

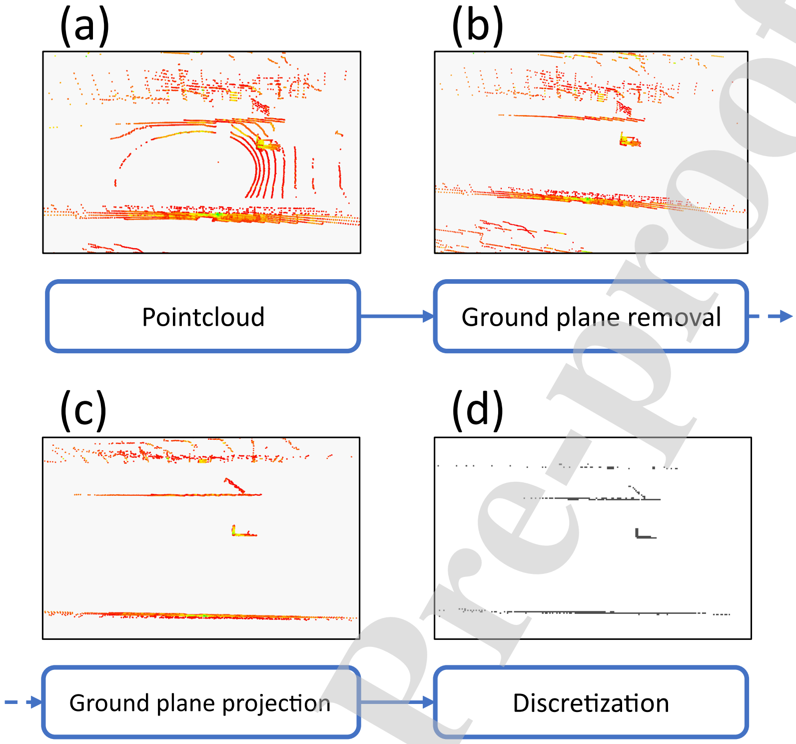


Figure 5. Pointcloud elaboration pipeline. The input pointcloud (a) is first processed to remove the ground plane point (b). Then is projected on a 2D plane parallel to the ground (c), and lastly converted into a binary grid map (d).

this step, all points above a fixed threshold, in our case $4.0\,m$, are also removed. This operation prevents the projection of noise from trees or traffic sign above the car clearance on the occupancy grid.

Once the ground plane is removed, the pointcloud includes only points belonging to obstacles. Thus, it is possible to project each one of them on a 2D plane using the normal direction retrieved in the previous step: the result is a set of 2D points on a plane parallel to the road surface. Although this process provides a significant simplification of the data, the set of measurements is still too complex to be directly used. Discretization is then carried out through the application of a grid on the identified horizontal plane. In particular, the grid is divided into square cells with side equal to $0.3\,m$: by iterating through each element, if the number of points in the cell is higher than a pre-computed threshold, the cell is set to occupied. The squared cells' size, equal to $0.3\,m$, is a good trade-off between accuracy and computational power. This value allows us to have a small occupancy grid, which can be computed and processed in real-time, but also retrieve the obstacle position with low error. The output of this filtering phase is a binary grid that describes the ego-vehicle surroundings

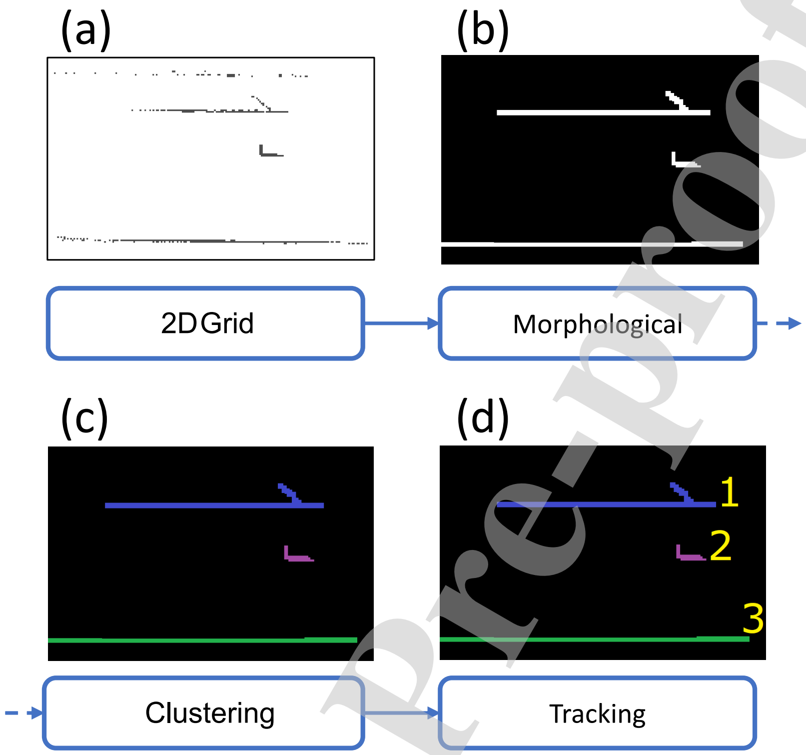


Figure 6. Occupancy grid elaboration pipeline. The occupancy grid (a) is first elaborated using morphological operations to remove noise and connect components (b). Then is processed using a clustering algorithm to identify all objects (c). Lastly, tracking is performed through consecutive frames of the identified obstacles (d).

with only a few thousand cells. The use of a threshold parameter is needed as it allows to reduce further the detection of false positives related to noise. Its value can be tuned based on experimental measurements with a decreasing value depending on the radial distance to consider the variable density of the pointcloud, as shown in [27].

The previous phase's output is a simplified representation of the area surrounding the ego-vehicle compared to a 3D dense pointcloud. However, this information cannot be directly supplied to the control routine of an autonomous vehicle. For this reason, a further elaboration block takes as input the 2D occupancy grid to return a small list of fully characterized obstacles.

The occupancy grid provides information regarding objects in each cell, but contiguous elements, which are parts of the same object, are considered separately. Thus, clustering is required to merge elements in the 2D-grid. As a preliminary step, a set of morphological operations is needed to connect areas that might belong to the same object but are not directly connected. This might happen due to some obstructions or the particular shape of the object itself, causing the number of points belonging to a specific cell to be lower than the filtering threshold explained before. This operation also filters single points in the occupancy grid, which are imputable to noise in the sensors, and can easily generate false positives. The result is still an occupancy grid where all elements belonging to an obstacle are connected. Further filtering is performed by discharging merged elements that are detected more than $50\,m$ meters ahead of the ego-vehicle, and $\pm 15\,m$ in the lateral direction.

Clustering is based on the OpenCV [60] implementation of SAUF (Scan plus Arraybased Union-Find) [61]; the output is a list of all the connected components in the occupancy grid which belong to real obstacles, defined by the relative position of the respective center of symmetry (CoS) with respect to the ego-vehicle and its equivalent dimension ρ_{o_i}. The length and width of each identified object are not considered because the mesh adopted for the 2D grid is not sufficiently fine to provide a measure of the heading. This causes a loss of accuracy in the estimation routine but allows us to provide obstacle measures at high frequency.

Obstacles tracking ensures accurate state estimation for many reasons. It allows to predict the relative positioning of obstacles with respect to the ego-vehicle also if measurements are not available; moreover, data coming from sensors that are not synchronized can be used for sensor fusion. For what concerns Lidar data processing, a feature-based approach guarantees preliminary obstacle tracking. In particular, for this stage, we use an object descriptor built using the obstacle dimensions and position. The first time the algorithm detects a specific obstacle, it assigns a unique ID and the respective features (i.e., dimensions and position). At each successive Lidar reading, the algorithm compares the previously detected obstacles with the current ones starting from the previously known locations. Warm starts are used to speed up calculations, together with a growing window that expands from given locations to search in the neighborhoods for objects with similar sizes. If a candidate tracked obstacle is found close enough to the previous one and with comparable dimensions, the same obstacle ID is assigned. When this process is completed for all obstacles, different IDs are automatically set for all elements coming from new readings that have not yet been tracked. Moreover, to account for noisy measurements or sensor misreadings, the algorithm keeps track of the older obstacles for which the matching has not been satisfied for 5 iterations (i.e., 0.25 s). Doing so, the algorithm can reassign IDs to un-tracked obstacles. To conclude, Lidar data processing provides a list of tracked obstacles characterized by relative positions with respect to ego-vehicle, size, and ID.

5.3. Sensor fusion

All measurements obtained through the processing of raw data from the two Radar sensors and the Lidar are expressed in VRF. The knowledge of the road limits is exploited for clutter removal, applied to all the processed measurements. Any measurement out of the road bounds is assumed to be clutter and hence removed. Radar measurements reported in (18) provide relative positioning and motion of the clustered detections belonging to the same obstacle with respect to the ego-vehicle. On the other hand, Lidar measurements provide relative positioning of each obstacle with respect to the ego-vehicle, and information about obstacle identification during time. Although the presented pre-processing of Lidar data already ensures tracking, the multi-sensors data fusion architecture proposed in this work can be considered centralized. Indeed, multi-sensors data pre-processing represents the input for a central module in which object discrimination and tracking are performed basing on the complete set of data.

Measurements from the two Radar sensors are synchronized with respect to each other, while they are asynchronous with respect to the data coming from Lidar. Thus, they are received associated with different timestamps. Moreover, as explained in Section 7, the estimation routine is driven at 20 Hz, while Radar and Lidar data processing are provided respectively at 14 and 16 Hz. Thus, it can happen that both sensors measurements do not retain the same timestamp and that no new measurements are available at a given time instant. For these reasons, sensor fusion is based on a LIFO routine (last in first out) in case of different timestamps. If Radar and Lidar measurements are available at the same time, fusion is performed through weighted averaging. In this case, the fused obstacle retains velocity measurements from the Radar, while positioning is computed assuming that Lidar measurements are more accurate, as they are related to the obstacle CoS. For a Radar measurement $y_{r\,o_j}^{VRF}$, a Lidar measurement $y_{l\,o_i}^{VRF}$ is considered for fusion if two criteria are satisfied:

1. For all $i \in [1, 2....N_{obs,\,l}]$ the Euclidean norm between $y_{r\,o_j}^{VRF}$ and $y_{l\,o_i}^{VRF}$ is minimum;
2. This minimum distance is smaller than the size of the object ρ_{o_i} estimated through Lidar processing.

$$\begin{cases} x_{F_j}^{VRF} = 0.8\, x_{l,\,o_i}^{VRF} + 0.2\, x_{r,\,o_j}^{VRF} \\ y_{F_j}^{VRF} = 0.8\, y_{l,\,o_i}^{VRF} + 0.2\, y_{r,\,o_j}^{VRF} \\ V_{x,\,F_j}^{VRF} = V_{x,\,r_j}^{VRF} \\ V_{y,\,F_j}^{VRF} = V_{y,\,r_j}^{VRF} \end{cases} \tag{21}$$

Sensor Id Assignment: To each measurement is assigned an Id based on the sensor it was originated from. The knowledge of the origin of the measurements was deemed helpful to perform gating task and measurement to track association. Since Lidar measurements do not provide information regarding velocity of detections, the gating task needs to be customized completely, basing it only on positional values. Similarly, the predicted state update by measurements needs to consider the unavailability of velocity measurement from the Lidar sensor as only positional values are used for update.

$$\begin{cases} Id_k^i = A,\ z_k^i \text{ Radar object detection without fusion with Lidar} \\ Id_k^i = AB,\ z_k^i \text{ Radar and Lidar Fused object detection} \\ Id_k^i = B,\ z_k^i \text{ is Lidar object detection without fusion with Radar} \end{cases} \quad (22)$$

Where, A, AB and B are some numerical constants used to identify the measurement origination in further estimation steps. Numerical values of these constants are not relevant as they are used solely for the purpose of identification of measurement origination. Whenever the measurement vector is specified in following sections, it must be implicitly understood that the Sensor *ID* comes assigned to it.
The fused measurement vector is calculated as in Equation (21). The objects identified by the Lidar sensor that do not satisfy these fusion criteria mentioned above, with respect to an object found by a Radar sensor, are assigned with different fusion *Id*, signifying that the measurement was obtained from Lidar only and was not fused with Radar data.

6. Obstacles state estimation and tracking

Tracking obstacles in autonomous driving allows establishing a control routine that considers the same obstacle during time to define proper control policies. This is mandatory both during vehicle following and overtaking maneuvers. Tracking can be performed only once the state estimation routine has provided measurement prediction for each obstacle, that must be defined in VRF according to Radar and Lidar data processing algorithms. For each obstacle, the state vector (3) defined in curvilinear coordinates according to the local reference frame of the road requires a highly nonlinear transformation to move each measurement prediction to VRF. For this reason, Unscented Kalman Filtering has been adopted to provide obstacles state estimates, which represents a compromise between accuracy and computational effort. The discrete-time implementation of the UKF is equal to the one defined in equations (4) and (5) with process disturbance and measurement noise assumed as additive and zero mean distributed (6). In the following, the model and measurement equations are presented.

The state variable $s_{i,loc}$ represents the distance computed along the road centerline in the local reference frame, between the ego-vehicle and the obstacle. While the variable $n_{i,loc}$ represents a measure of how much the obstacle is displaced with respect to the centerline, and V_{s_i} and V_{n_i} are the components of the obstacle absolute velocity in curvilinear coordinates. Since the small objects' hypothesis is adopted throughout this work, each obstacle is considered a single point (i.e., its heading angle is not estimated).

The two different reference frames are represented in Fig. 7, where θ_e and θ_o are the heading angles of the road centerline in correspondence of ego-vehicle and obstacle respectively. About $\vec{s_e}$ and $\vec{n_e}$, they are the tangential and normal directions to the local reference frame of the road, that is centered in the

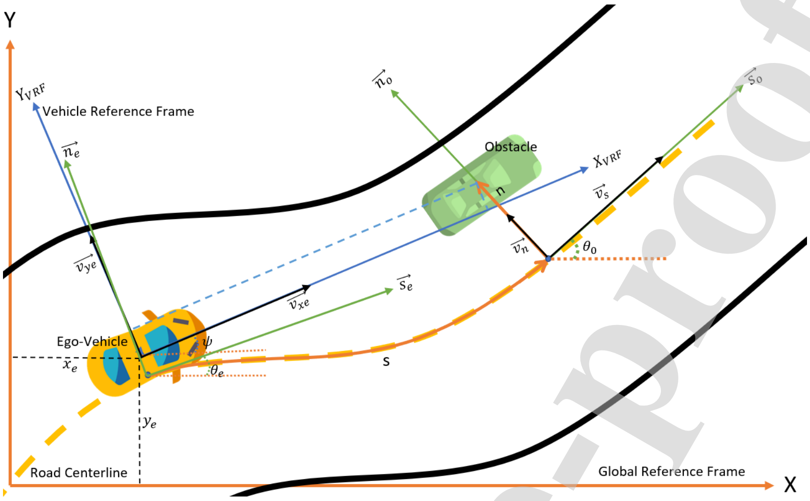


Figure 7. Representation of road and vehicle reference frame

closest point corresponding to the ego-vehicle belonging to the road centerline. Similarly, $\vec{s_o}$ and $\vec{n_o}$ are the main directions of the road in correspondence of the obstacle. Finally, ψ is the estimated heading angle of the ego-vehicle in the absolute reference frame.

The definition of the nonlinear transformation that allows moving from curvilinear to Cartesian coordinates in VRF is required to ensure the measurement prediction during the filtering process. For this purpose, an Euler-based conversion model is devised. In particular, this model allows computing the Cartesian coordinates (x_c, y_c) corresponding to the point that is $s_{i,\,loc}$ away from the ego-vehicle, measured along the road centreline (Fig. 8). The calculation is based on Euler integration with step size equal to $\delta s = 0.5\,m$. Once the ego-vehicle is localized on the track, the road map provides the road heading for the next $50\,m$. Given N the required number of steps, with $N = s_{i,\,loc}/\delta s$, the model computes:

$$\begin{cases} x^{k+1} = x^k + \cos(\theta_{s_k})\delta s \\ y^{k+1} = y^k + \sin(\theta_{s_k})\delta s \end{cases} \tag{23}$$

where the point (x^N, y^N) is approximately equal to (x_c, y_c), and θ_{s_k} is the road heading angle for each step. Then, the lateral displacement of the obstacle from the road centerline $n_{i,\,loc}$ is used to compute its position in the Global Reference Frame.

$$\begin{cases} x_g = x_c + \cos(\frac{\pi}{2} - \theta_0)n_{i,\,loc} \\ y_g = y_c + \sin(\frac{\pi}{2} - \theta_0)n_{i,\,loc} \end{cases} \tag{24}$$

Finally, the rotation matrix based on the heading angle of the ego vehicle ψ_{abs}

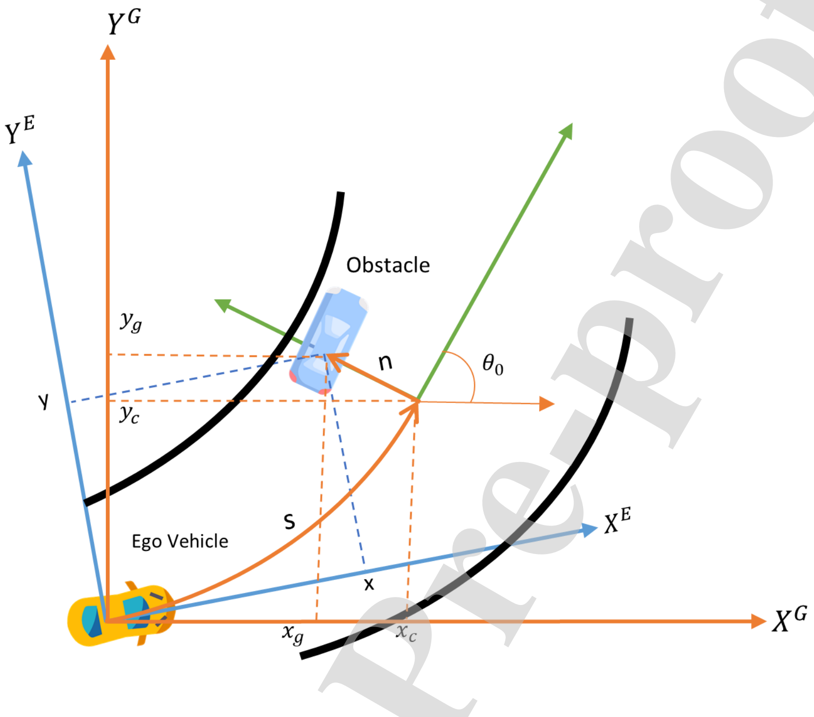


Figure 8. Coordinate conversion between curvilinear and Cartesian reference frame

allows computing the relative positioning in VRF.

$$\begin{bmatrix} x_{o_i}^{VRF} \\ y_{o_i}^{VRF} \end{bmatrix} = \begin{bmatrix} cos(\psi_{abs}) & -sin(\psi_{abs}) \\ sin(\psi_{abs}) & cos(\psi_{abs}) \end{bmatrix} \begin{bmatrix} x_g \\ y_g \end{bmatrix} \tag{25}$$

The presented mathematical model assumes that the ego-vehicle is located on the road centerline. However, the estimated lateral displacement is considered through the following equation:

$$y_{o_i}^{VRF} = y_{o_i}^{VRF} + \cos(\psi_{abs} - \theta_e) n_{G_{loc}} \tag{26}$$

The conversion of the obstacle absolute velocity from curvilinear coordinates to VRF can be done rotating the velocity vector two times as in (27). The former accounts for the road's heading angle in correspondence of the obstacle θ_o to transform velocity components from road to Cartesian global reference frame. About the latter one, it moves the two components in VRF through the ego-vehicle absolute heading angle(ψ_{abs}).

$$\begin{bmatrix} V_{x,\,o_i}^{VRF} \\ V_{y,\,o_i}^{VRF} \end{bmatrix} = \begin{bmatrix} cos(\psi_{abs}) & -sin(\psi_{abs}) \\ sin(\psi_{abs}) & cos(\psi_{abs}) \end{bmatrix} \begin{bmatrix} cos(\theta_o) & sin(\theta_o) \\ -sin(\theta_o) & cos(\theta_o) \end{bmatrix} \begin{bmatrix} V_{s_i} \\ V_{n_i} \end{bmatrix} \tag{27}$$

The filter initialization is performed with the measurements obtained from sensor fusion. During the first iteration, these processed measurements in VRF are equated into Curvilinear Co-ordinate frame to initialize tentative tracks. Concurrently, initialization of the tracking routine is done using these tentative tracks. If they are retained during the next second (i.e., for 20 iterations), the tracked hypothesis is converted to a confirmed tracked obstacle. If not, any other tentative track is deleted.

Once initialization is completed, state prediction is performed based directly on the previously tracked obstacles state estimates and covariance. Indeed, the constant velocity lane changing model (CVLC) [20], which defines the obstacles motion in curvilinear coordinates, it is a linear model, as shown in (28).

$$\tilde{x}^-_{o_i\,k} = \begin{bmatrix} s_{i,\,loc} \\ n_{i,\,loc} \\ V_{s_i} \\ V_{n_i} \end{bmatrix}_k = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{F_{k-1}} \begin{bmatrix} s_{i,\,loc} \\ n_{i,\,loc} \\ V_{s_i} - V_s \\ V_{n_i} \end{bmatrix}_{k-1} + \begin{bmatrix} \omega_{a_s}(\delta t^2/2) \\ \omega_{a_n}(\delta t^2/2) \\ V_{s\,k-1} + \omega_{a_s}\delta t \\ \omega_{a_n}\delta t \end{bmatrix} \tag{28}$$

The terms ω_{a_s} and ω_{a_n} are used to add Gaussian noise within the linear model that describes obstacle motion in curvilinear coordinates. They can be considered with zero mean and associated to the standard deviation of accelerations in curvilinear coordinates respectively as $N(0, \sigma^2_{a_s})$ and $N(0, \sigma^2_{a_n})$. Moreover, according to linear Kalman filtering in discrete time, the state prediction covariance for each obstacle can be computed as in (29), where F_{k-1} is the matrix of the linear model and $P^+_{o_i\,k-1}$ is the covariance matrix of the state updated by the measurements at the previous step. Nevertheless, the unscented transformation is still performed as in (9) to allow computing the cross covariance matrix (14). In practice, among all the $2n+1$ sigma points, only one is used to perform state prediction and covariance, while the remaining $2n$ are required to compute $P_{xy,\,o_i}$.

$$P^-_{o_i\,k} = F_{k-1}P^+_{o_i\,k-1}F^T_{k-1} + Q_{k-1} \qquad i = 1, \ldots, N_{obs} \tag{29}$$

It is important to notice that while obstacle positioning is relative to the road reference frame, V_{s_i} and V_{n_i} are the components of the absolute velocity of a tracked obstacle. Thus, to ensure the correct prediction of $s_{i,\,loc}$ at the current time step, it is required to consider the difference between obstacle and vehicle velocity along the road direction V_s. This is valid in case of both positive and negative values of $s_{i,\,loc}$. Equation (30) summarizes the clockwise rotation required to obtain the components of the ego-vehicle absolute velocity in curvilinear coordinates.

$$V_s = cos(\xi_{loc})V_x + sin(\xi_{loc})V_y \tag{30}$$

The predicted state vector $\tilde{x}^-_{o_i\,k}$ for each tracked obstacle is used to perform the unscented transformation (9) through the covariance matrix $P^-_{o_i\,k}$. The new sigma points $(2n)$ are then used to compute the predicted measurements matrix

$\hat{y}^{(i)}_{o_i\,k}$. As shown in Section 4, this matrix is computed by feeding measurements equations (23) to (27) with sigma points to obtain the predicted measurements vectors for each obstacle $\tilde{y}_{o_i\,k}$ and the related innovation covariance matrix $P_{y,\,o_i}$. This is shown in (12) and (13).

To reduce the number of association hypotheses required to compare predicted measurements with the ones received from sensor fusion module $y_{o_i\,k}$, fused measurements are taken into account only if they fall within a gate created around predicted measurements $\tilde{y}_{o_i\,k}$. Under the assumption of Gaussian distributed noise, it is possible to adopt ellipsoidal gates [24]. In particular, an ellipsoidal gate is defined through a gating probability P_G, which represents the probability that the object measurement is inside the gate, together with a cumulative distribution $\chi^2(n)$ required to compute the gate size G. Then, the so-called *Mahalanobis distance* can be calculated as in (31) to find which fused measurements are inside the gates:

$$D^2(y_{o_j\,k}, \tilde{y}_{o_i\,k}) = [y_{o_j\,k} - \tilde{y}_{o_i\,k}]^T (P_{y,\,o_i})^{-1} [y_{o_j\,k} - \tilde{y}_{o_i\,k}] \tag{31}$$

for $i = 1, \cdots, n$ and $j = 1, \cdots, m$. About n and m, they indicate respectively the number of predicted measurements during the current time step and the number of tracked objects at the previous one. Any measurement $y_{o_j\,k}$ that does not satisfy the criterion (32) is hence disregarded from the association set and will be used to initialize new tentative obstacles. Conversely, all the measurements included in ellipsoidal gates are collected and used for association.

$$D^2(y_{o_j\,k}, \tilde{y}_{o_i\,k}) < G \tag{32}$$

Association is done gathering all the selected measurements in one single matrix. Although grouping by gating is computationally cheaper, for a moderate number of tracked obstacles the exhaustive method does not reduce performances. Association is then performed through a GNN algorithm that considers only the best association hypotheses due to the lowest cost while discharging all the others. To do so, the cost matrix L is defined through the likelihoods of association between tracked objects and measurements inside gates, together with the likelihoods of misdetection. These likelihoods can be calculated by knowing the probability of detection $p(d)$ as in (33), assuming that the one assigned by the Radar $(p(o_i) > 0.99)$ is much lower with respect to the one guaranteed through the processing of Lidar data.

$$l_k^{i,0} = log(1 - p(d)) \tag{33}$$

$$\begin{aligned} l_k^{i,j} =& log\Big(\frac{p(d)}{\lambda(c)}\Big) - \frac{1}{2} log\Big(det(2\pi P_{y,\,o_i})\Big) + \\ & - \frac{1}{2}[y_{o_j\,k} - \tilde{y}_{o_i\,k}]^T (P_{y,\,o_i})^{-1} [y_{o_j\,k} - \tilde{y}_{o_i\,k}] \end{aligned} \tag{34}$$

This formulation is valid only if the value of $p(d)$ is assumed as constant and the clutter intensity $\lambda(c) = \lambda / FoV$ is positive and constant, where $\lambda(c)$ can be considered as the average number of clutters along the bounded FoV per time step. The average number of clutters for each time step $\lambda = 2$ has been determined through simulation based on real data, processed as stated in Section 5.

The cost matrix L is hence a $[n\cdot(m+n)]$ rectangular matrix, as shown in (35), in which the $[n \cdot m]$ left sub-matrix considers only real detections and is defined by likelihoods of association between tracked objects and measurements. On the other hand, the $[n \cdot n]$ right sub-matrix collects all the misdetections determined by the corresponding likelihoods.

$$L = \left[\begin{array}{cccc|cccc} -l^{1,1} & -l^{1,2} & \cdots & -l^{1,m} & -l^{1,0} & \infty & \cdots & \infty \\ -l^{2,1} & -l^{2,2} & \cdots & -l^{2,m} & \infty & -l^{2,0} & \cdots & \infty \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -l^{n,1} & -l^{n,2} & \cdots & -l^{n,m} & \infty & \infty & \cdots & -l^{n,0} \end{array}\right] \tag{35}$$

Moreover, given the assignment matrix A, the corresponding assignment cost can be defined by solving the optimization problem (36). The solution to this problem is found adopting the 2D assignment algorithm described in [62].

$$\min tr(A^T L) = \sum_{i=1}^{n} \sum_{j=1}^{m+n} A^{i,j} L^{i,j} \tag{36a}$$

subject to:

$$A^{i,j} \in \{0 \quad 1\} \tag{36b}$$

$$\sum_{j=1}^{n+m} A^{i,j} = 1 \tag{36c}$$

$$\sum_{i=1}^{n} A^{i,j} \in \{0 \quad 1\} \tag{36d}$$

The optimal solution ensures the optimal correspondence between tracked objects and measurements required to update state prediction and covariance for each obstacle as in (16) and (17). If no measurements are provided from sensor fusion or there are no measurements inside any gate, equations (37) and (38) are adopted.

$$\tilde{x}_k^+ = \tilde{x}_k^- \tag{37}$$

$$P_k^+ = P_k^- \tag{38}$$

As previously stated, non associated measurements are used as new generations to initialize the tracking process. If a tentative obstacle is updated with the assigned measurement throughout 1 second (i.e., for 20 iterations), it is

Figure 9. Picture of the vehicle, the Lidar sensor is visible on the roof, while the two Radar are incorporated in the rear and front bumpers

confirmed as a real obstacle. Moreover, a warning can be provided to the controller if an object suddenly appears close to the ego-vehicle but then it is not confirmed. Although this could be justified to enhance safety, simulation tests carried on experimental data showed that objects that suddenly appear close to the ego-vehicle without being tracked earlier could be considered as clutters. On the other hand, if no measurements are assigned to a tentative obstacle during the following 5 iterations (i.e., $0.25\,s$), this is deleted. An obstacle that has already been confirmed is kept in record for 10 iterations (i.e., $0.5\,s$): if any measurement is associated with it, this obstacle is still seen as confirmed and state estimation provided.

7. Experimental results

The presented algorithm provides ego-vehicle and obstacles state estimation in curvilinear coordinates for an autonomous vehicle. Ego-vehicle state estimation is computed in the global reference frame and then collocated in the road's local reference frame. This is done by exploiting the map's knowledge, which associates to each point of the centerline the description of the road heading and curvature along the considered FoV. Once the ego-vehicle is collocated within the road map, raw data coming from Radar sensors and Lidar are processed and fused in VRF to provide tentative obstacles to the tracker. Then, state estimation is performed in curvilinear coordinates. The algorithm has been validated during some experimental campaigns carried on Monza Eni Circuit.

The instrumented vehicle, showed in Fig. 9, is a prototype for an autonomous driving car [17] equipped with sensors for the measurement of absolute positioning, odometry, and motion. In particular, the sensor suite includes:

- two *Piksi Multi* GPS receivers are located along the vehicle's longitudinal axis, coupled with a ground station through 4G connection. They provide positioning in absolute coordinates with RTK correction and velocities in East-Nord-UP (ENU) reference frame. As shown in section 4, velocities allow predicting the measurement of the ego-vehicle heading angle within the UKF. Measurements are provided at $10\,Hz$;

- an IMU located in correspondence of the vehicle CoG, which measures linear accelerations and angular velocities on the three principal axes. Measurements are available at $100\,Hz$.

- odometry is given at $20\,Hz$ by an encoder mounted on the steering wheel to measure the steering angle, while two exciters on the rear axle provide the longitudinal speed of the vehicle;

- two *Continental ARS 408-21* Radar sensors provide relative positioning and motion of obstacles in VRF at $14\,Hz$. They are located in the front and rear bumpers of the vehicle;

- a *Velodyne VLP-16* Lidar mounted on the roof provides 3D pointclouds at $20\,Hz$.

The overall estimation routine runs at $20Hz$ on a soft real-time system based on ROS (Robot Operating System). This allows managing the different sampling frequencies, because triggering can be based on ROS timestamps. If no measurements arrive from the GPS receivers state prediction is used instead of state estimation ((37), (38)).

Concerning ego-vehicle estimates, accuracy can be assessed by analyzing the predicted heading angle and the lateral speed in the vehicle CoG, which are not measured by any sensor included within the listed suite. To do so, a further automotive optical sensor has been mounted on the vehicle during some experimental campaigns to collect ground truth data regarding vehicle sideslip. The comparison between measured and estimated longitudinal and lateral speeds is presented in Fig.10. The figure points out the comparison between measurements and estimates during a steering pad maneuver completed on a circle with a radius equal to $27\,m$. As shown in the first two subplots, the vehicle's longitudinal speed increases approximately from 20 to $40\,km/h$, while the steering angle is worth about $100\,deg$. The third and last subplot points out a strong correlation between estimated and measured lateral speed. Moreover, during the presented maneuver the vehicle is close to the tires' friction saturation: this is highlighted to assess the effectiveness of the estimation algorithm.

For what concerns the estimation of heading angle, it is not possible to define a ground truth basing on the angle between the horizontal and the straight line that connects the measures given by the GPS receivers at the same time step. Although the RTK correction ensures that the measurement error for positioning decreases up to a few centimeters, this still affects the heading angle's estimate with an error that depends on the distance between the two receivers. For the presented vehicle, these drift effects produce an error that varies in the

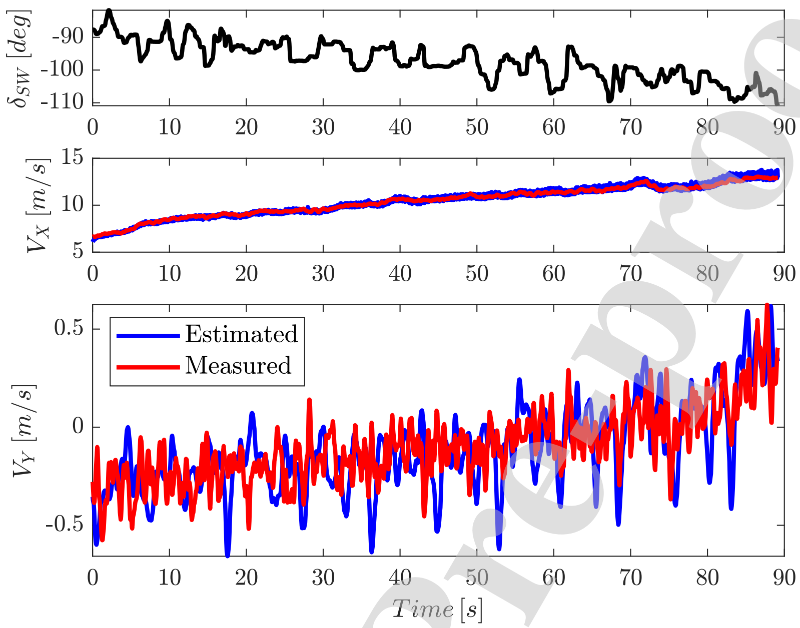


Figure 10. Comparison between estimated and measured lateral speed during a steering pad manoeuvre performed at increasing speed

range $\pm 10\,deg$, which is too high to guarantee a significant ground truth. A further possibility is to analyze in time the angle found by tracking subsequent positions of the rear GPS receiver (i.e., the one less affected by steering effects). However, this angle is the tangent to the trajectory completed by the rear part of the vehicle (γ_R), which is related to the vehicle heading angle as indicated by Eq.(39).

$$\begin{cases} \gamma_R - \beta_R = \psi_{abs} \\ \beta_R = atan((V_y - \dot{\psi} l_R)/V_x) \end{cases} \tag{39}$$

Here, the vehicle's sideslip angle is reported to the rear's GPS receiver, considering the variation of lateral speed. This is done accounting for the distance to the vehicle CoG and the yaw rate. Given that the heading angle ψ_{abs} is constant along the vehicle, it is possible to state that the estimate is correct if the difference $\gamma_R - \psi_{abs} - \beta_R$ is null for any time instant. This difference is reported for the aforementioned steering pad in Fig.11, whose offset from null is constant and equal to $+0.06\,deg$. This result assesses the performance of the estimation also during a challenging driving maneuver. Indeed, although the sideslip angle β_R increases from 1 to $5\,deg$, the offset remains constant. Regarding the high level of noise in the plot, this is due to the lateral speed measurements provided

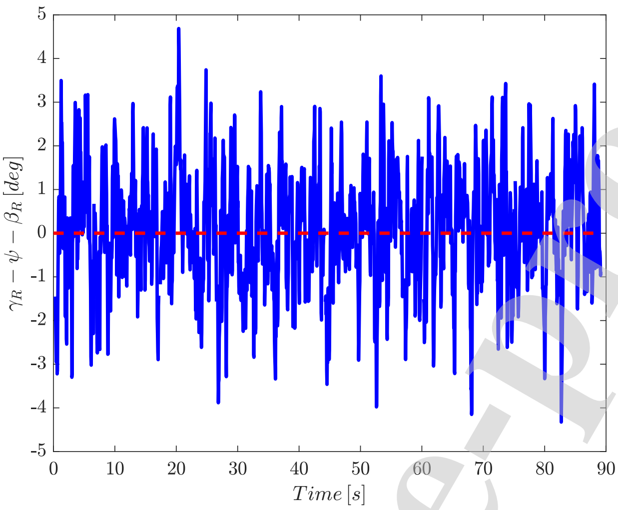


Figure 11. Validation of the estimation of the ego-vehicle heading angle. The difference between trajectory angle and sideslip angle at the rear, and the heading angle must be null

by the automotive optical sensor. Furthermore, a qualitative representation of the vehicle's estimated heading angle is reported in Fig. 12. Two different plots point out the vehicle's direction during the first two tight chicanes of the track that the vehicle performs respectively from the bottom to the top of the first plot, and from left to right in the second one. The quality of the estimate can be evaluated observing the direction during straights, superimposed to the predicted position of the vehicle CoG. At the same time, during curving, the heading angle is comparable to the tangent to the trajectory.

The validation of the obstacles' state estimation module is allowed by a set of experimental data collected in some significant mutual maneuvers between the ego-vehicle and a designated obstacle vehicle (FIAT Talento, a van whose dimensions are 5x2x2 m). To assess the accuracy of the algorithm, the absence of false positives, and the accuracy of the estimated state vector are analyzed. The results discussed in this section derives from a vehicle-following maneuver: the obstacle is driving ahead of the autonomous vehicle between turn 3 and turn 6, hence the road curvature changes significantly during the test. The algorithm performs well in filtering clutters within and out of road bounds. Moreover, the presented results prove that it performs well also during tight curvature scenarios.

A snapshot from the described scenario is reported in Fig. 13. For ease of viewing, the overall framework with ego-vehicle, measurements, and obstacles is shown in Cartesian coordinates, in the global reference frame. Nevertheless, the plot reports the estimated positioning of the obstacle in curvilinear coordinates in the road reference frame, i.e., the longitudinal distance $s_{i,loc}$, and the

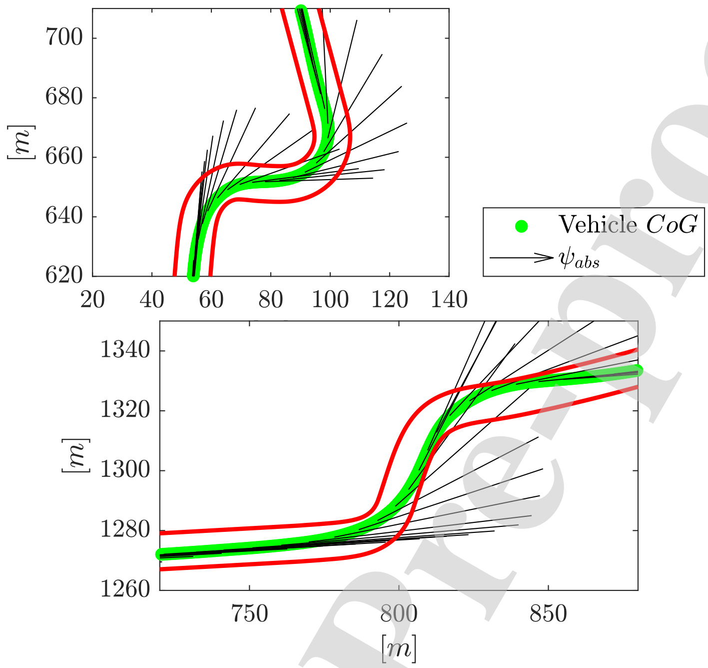


Figure 12. Heading angle estimate for the ego-vehicle

lateral displacement to centreline $n_{i,\,loc}$. The ego-vehicle estimated position is represented by a (○).

The plot shows multiple measurements obtained from Radar and Lidar sensors. The (○) marks are clustered Radar objects fed to the road filtering module which provides (*) as output. These results are the inputs to the next sensor fusion module. Similarly, the (○) mark represents the output of the Lidar processing module and (*) are the Lidar measurements within road boundaries fed as inputs to the sensor fusion module. As shown in the plot, information about road width allows filtering all the measurements related to any obstacle or object out of interest. In this particular instance, measurements coming from both the Lidar and the Radar processing modules are simultaneously available for the fusion module. Thus, fused measurements computed by Eq. 21, are represented by the (○) mark. As explained in previous sections, this output is used for object initialization and association in the remaining steps of the estimation routine. The algorithm is also able to ensure the accuracy of the object cardinality, which in this scenario is consistently equal to one, by implementing a track confirmation and removal routine. Although the figure illustrates two detections from the sensor fusion module, the tracking algorithm accurately confirms a single object while providing its state estimate as confirmed. The

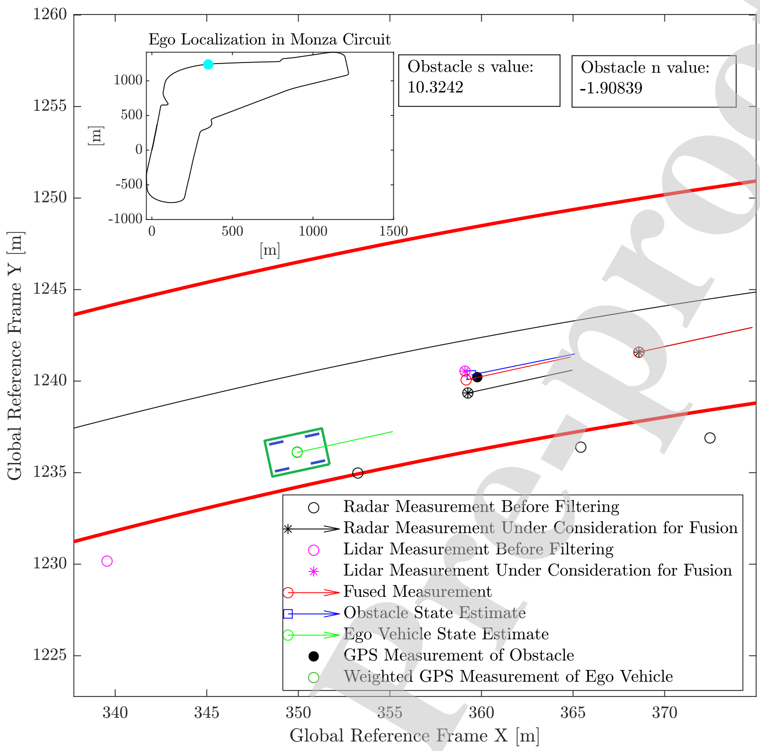


Figure 13. Visual representation of obstacle identification and tracking. The green box represents the ego-vehicle, while the circles represent the obstacles identified by the different sensors. (attached video V1.avi)

obstacle state estimate is represented by (□), while the estimated distances in curvilinear co-ordinates are mentioned in the bottom part of the figure. Moreover, for those clusters or estimates whose velocity is known or computed, the plot points out a vector that represents its magnitude and direction.

To conclude, Fig. (14) illustrates the comparison of the estimated obstacle's state vector with the ground truth given by the GPS receiver installed on the tracked obstacle vehicle with RTK correction. Both the GPS measurements and the estimates are represented in vehicle reference frame (VRF). Due to unavailability of ground truth in curvilinear co-ordinates, estimates are converted from curvilinear coordinates to VRF by applying the Euler model presented in the previous section. The root mean square error ($RMSE$) is computed as the distance between the estimated position of the obstacle vehicle and the real one. In the described scenario, the algorithm performs the estimation with $RMSE = 0.6039\,m$, that is reasonable compared to the size of the obstacle.

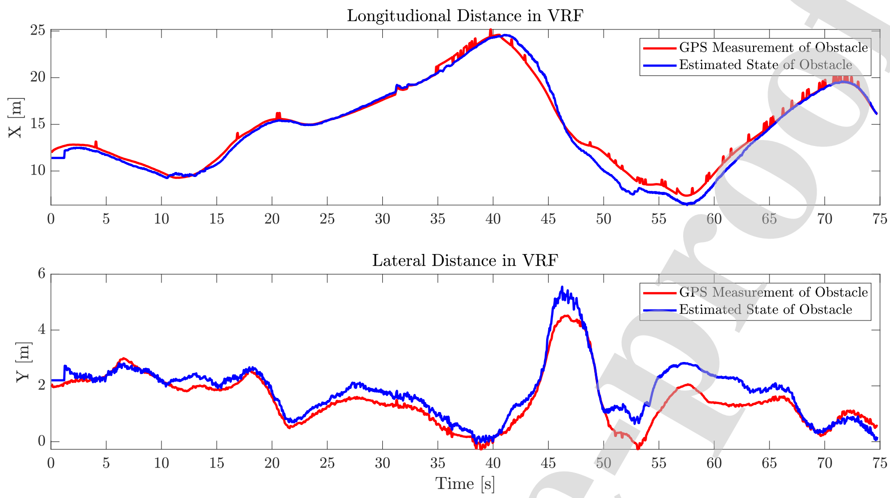


Figure 14. Comparison between the estimated relative position between the obstacle and the ego-vehicle, and the ground truth given by the GPS receiver

8. Conclusions

The presented paper focuses on state estimation applied to autonomous vehicles. It describes an integrated algorithm that computes ego-vehicle and obstacles' state estimation in curvilinear coordinates, according to the road reference frame. The ego-vehicle's state vector includes positioning, heading angle, and the longitudinal and lateral components of velocity in the vehicle reference frame (VRF). Estimates are provided in Cartesian coordinates and then converted to the local reference frame of the road. About the obstacles in the surrounding of the ego-vehicle, the presented algorithm computes their relative position and absolute velocity in curvilinear coordinates according to the road reference frame, under the assumption of small dimensions. Measurements of obstacles are provided by a multi-sensor framework, which includes two Radars located within the vehicle front and read bumpers and a Lidar mounted on the vehicle top in correspondence of the center of gravity. Sensor fusion provides the tracking module with filtered measurements, allowing to associate each of them to the respective obstacle. Association is performed through GNN. Due to strong nonlinearities in each measurement model of the two filters, both the estimation routines are based on Unscented Kalman Filters. The integrated algorithm has been validated through experimental tests carried in the Monza ENI circuit. The overall estimation routine runs at 20 Hz on a soft real-time system based on ROS: this allows managing the different sampling frequencies of each sensor.

To conclude, the presented estimation algorithm provides a detailed set of initial conditions for any motion planning routine for autonomous vehicles. In future works, ego-vehicle dynamic behavior will be considered at least in the lateral direction; moreover, a camera will be installed on the car, to improve sensor fusion and object tracking, basing on the high semantic content of images.

9. Acknowledgement

The research leading to these results has received funding from project TEINVEIN: TEcnologie INnovative per i VEicoli Intelligenti, CUP (Codice Unico Progetto - Unique Project Code): E96D17000110009 - Call "Accordi per la Ricerca e l'Innovazione", cofunded by POR FESR 2014-2020 (Programma Operativo Regionale, Fondo Europeo di Sviluppo Regionale – Regional Operational Programme, European Regional Development Fund).

The authors gratefully acknowledge the administrators of the Monza ENI Circuit for their availability and technical support.

References

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, E. Frazzoli, A survey of motion planning and control techniques for self-driving urban vehicles, IEEE Transactions on intelligent vehicles 1 (2016) 33–55.

[2] A. Van Zanten, R. Bosch Gmbh, Evolution of electronic control systems for improving vehicle dynamic behaviour, in: Proceedings of the International Symposium on Advanced Vehicle Control (AVEC), 2002.

[3] J. Kim, K. Jo, W. Lim, M. Lee, M. Sunwoo, Curvilinear-coordinate-based object and situation assessment for highly automated vehicles, IEEE Transactions on Intelligent Transportation Systems 16 (2015) 1–17. doi:10.1109/TITS.2014.2369737.

[4] R. Danescu, F. Oniga, S. Nedevschi, Modeling and tracking the driving environment with a particle-based occupancy grid, IEEE Transactions on Intelligent Transportation Systems 12 (2011) 1331–1342. doi:10.1109/TITS.2011.2158097.

[5] A. Barth, U. Franke, Tracking oncoming and turning vehicles at intersections, in: 13th International IEEE Conference on Intelligent Transportation Systems, 2010, pp. 861–868. doi:10.1109/ITSC.2010.5624969.

[6] B. Fortin, R. Lherbier, J. Noyer, A model-based joint detection and tracking approach for multi-vehicle tracking with lidar sensor, IEEE Transactions on Intelligent Transportation Systems 16 (2015) 1883–1895. doi:10.1109/TITS.2015.2391131.

[7] B. Jahromi, T. Tulabandhula, S. Cetin, Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles, Sensors 19 (2019) 4357. doi:10.3390/s19204357.

[8] H. Cho, Y. Seo, B. V. K. V. Kumar, R. R. Rajkumar, A multi-sensor fusion system for moving object detection and tracking in urban driving environments, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 1836–1843. doi:10.1109/ICRA.2014.6907100.

[9] H. Zhu, K.-V. Yuen, L. Mihaylova, H. Leung, Overview of environment perception for intelligent vehicles, IEEE Transactions on Intelligent Transportation Systems PP (2017). doi:10.1109/TITS.2017.2658662.

[10] D. Göhring, M. Wang, M. Schnürmacher, T. Ganjineh, Radar/lidar sensor fusion for car-following on highways, in: The 5th International Conference on Automation, Robotics and Applications, 2011, pp. 407–412. doi:10.1109/ICARA.2011.6144918.

[11] J. Hollinger, B. Kutscher, R. Close, Fusion of lidar and radar for detection of partially obscured objects, 2015, p. 946806. doi:10.1117/12.2177050.

[12] C. Yi, K. Zhang, N. Peng, A multi-sensor fusion and object tracking algorithm for self-driving vehicles, Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 233 (2019) 2293–2300. doi:10.1177/0954407019867492.

[13] E. Héry, S. Masi, P. Xu, P. Bonnifait, Map-based curvilinear coordinates for autonomous vehicles, 2017, pp. 1–7. doi:10.1109/ITSC.2017.8317775.

[14] G. Inghilterra, S. Arrigoni, F. Braghin, F. Cheli, Firefly algorithm-based nonlinear mpc trajectory planner for autonomous driving, in: 2018 International Conference of Electrical and Electronic Technologies for Automotive, 2018, pp. 1–6.

[15] S. Arrigoni, E. Trabalzini, M. Bersani, F. Braghin, F. Cheli, Nonlinear mpc motion planner for autonomous vehicles based on accelerated particle swarm optimization algorithm, 2019, pp. 1–6. doi:10.23919/EETA.2019.8804561.

[16] L. Zheng, B. Li, B. Yang, H. Song, Z. Lu, Lane-level road network generation techniques for lane-level maps of autonomous vehicles: A survey, Sustainability 11 (2019) 4511. URL: `http://dx.doi.org/10.3390/su11164511`. doi:10.3390/su11164511.

[17] M. Vignati, D. Tarsitano, M. Bersani, F. Cheli, Autonomous steer actuation for an urban quadricycle, in: 2018 International Conference of Electrical and Electronic Technologies for Automotive, 2018, pp. 1–5.

[18] Monza eni circuit, 2020. `https://www.monzanet.it/`.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, Ros: an open-source robot operating system, in: ICRA workshop on open source software, volume 3, Kobe, Japan, 2009, p. 5.

[20] K. Jo, M. Lee, J. Kim, M. Sunwoo, Tracking and behavior reasoning of moving vehicles based on roadway geometry constraints, IEEE Transactions on Intelligent Transportation Systems PP (2016) 1–17. doi:10.1109/TITS.2016.2605163.

[21] D.Simon, Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches, Wiley, 2006.

[22] S. J. Julier, J. K. Uhlmann, Unscented filtering and nonlinear estimation, in: Proceedings of the IEEE, volume 92, 2004.

[23] R. Juang, P. Burlina, Comparative performance evaluation of gm-phd filter in clutter, in: 2009 12th International Conference on Information Fusion, 2009, pp. 1195–1202.

[24] Y. X. Lennart Svensson, Karl Granstrom, Multi-object tracking for automotive systems, https://courses.edx.org/courses/course-v1:ChalmersX+ChM013x+3T2019/course/, 2019.

[25] P. Tripathi, K. Nagla, H. Singh, S. Mahajan, Occupancy grid mapping for mobile robot using sensor fusion, in: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), IEEE, 2014, pp. 47–51.

[26] D. V. Lu, D. Hershberger, W. D. Smart, Layered costmaps for context-sensitive navigation, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2014, pp. 709–715.

[27] S. Mentasti, M. Matteucci, Multi-layer occupancy grid mapping for autonomous vehicles navigation, in: 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), IEEE, 2019, pp. 1–6.

[28] P. Fankhauser, M. Hutter, A universal grid map library: Implementation and use case for rough terrain navigation, in: Robot Operating System (ROS), Springer, 2016, pp. 99–120.

[29] P. Fankhauser, M. Bloesch, M. Hutter, Probabilistic terrain mapping for mobile robots with uncertain localization, IEEE Robotics and Automation Letters (RA-L) 3 (2018) 3019–3026. doi:10.1109/LRA.2018.2849506.

[30] J. Gu, Q. Cao, Path planning for mobile robot in a 2.5-dimensional grid-based map, Industrial Robot: An International Journal 38 (2011) 315–321.

[31] A. A. Souza, L. M. Goncalves, 2.5-dimensional grid mapping from stereo vision for robotic navigation, in: 2012 Brazilian Robotics Symposium and Latin American Robotics Symposium, IEEE, 2012, pp. 39–44.

[32] M. Himmelsbach, A. Mueller, T. Lüttel, H.-J. Wünsche, Lidar-based 3d object perception, in: Proceedings of 1st international workshop on cognition for technical systems, volume 1, 2008.

[33] D. Anguelov, B. Taskarf, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, A. Ng, Discriminative learning of markov random fields for segmentation of 3d scan data, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, IEEE, 2005, pp. 169–176.

[34] J.-F. Lalonde, N. Vandapel, D. F. Huber, M. Hebert, Natural terrain classification using three-dimensional ladar data for ground robot mobility, Journal of field robotics 23 (2006) 839–861.

[35] Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, N. Sun, Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving, IEEE Robotics and Automation Letters 3 (2018) 3434–3440.

[36] B. Zhu, Z. Jiang, X. Zhou, Z. Li, G. Yu, Class-balanced grouping and sampling for point cloud 3d object detection, arXiv preprint arXiv:1908.09492 (2019).

[37] Y. Yan, Y. Mao, B. Li, Second: Sparsely embedded convolutional detection, Sensors 18 (2018) 3337.

[38] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, O. Beijbom, Pointpillars: Fast encoders for object detection from point clouds, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 12697–12705.

[39] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.

[40] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, arXiv preprint arXiv:1706.02413 (2017).

[41] B. Li, T. Zhang, T. Xia, Vehicle detection from 3d lidar using fully convolutional network, arXiv preprint arXiv:1608.07916 (2016).

[42] G. Yang, S. Mentasti, M. Bersani, Y. Wang, F. Braghin, F. Cheli, Lidar point-cloud processing based on projection methods: a comparison, 2020. arXiv:2008.00706.

[43] J. van Genderen, Tracking and data fusion: a handbook of algorithms, by yaakov bar-shalom, peter k. willett and xin tian, International Journal of Image and Data Fusion 4 (2013) 102–104. URL: https://doi.org/10.1080/19479832.2012.749304. doi:10.1080/19479832.2012.749304.

[44] H. Winner, S. Hakuli, F. Lotz, C. Singer, Data fusion of environment-perception sensors for adas, in: Handbook of Driver Assistance Systems - 2016, 2016.

[45] E. Héry, P. Xu, P. Bonnifait, Along-track localization for cooperative autonomous vehicles, in: 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 511–516.

[46] J. Hudecek, L. Eckstein, Improving and simplifying the generation of reference trajectories by usage of road-aligned coordinate systems, in: 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014, pp. 504–509. doi:10.1109/IVS.2014.6856502.

[47] J. Ryu, J. Christian Gerdes, Integrating inertial sensors with global positioning system (gps) for vehicle dynamics control, Journal of Dynamic Systems Measurement and Control-transactions of The Asme - J DYN SYST MEAS CONTR 126 (2004).

[48] D. M. Bevly, J. Ryu, J. C. Gerdes, Integrating ins sensors with gps measurements for continuous estimation of vehicle sideslip, roll, and tire cornering stiffness, IEEE Transactions on Intelligent Transportation Systems 7 (2006) 483–493.

[49] T. Wenzel, K. Burnham, M. Blundell, R. Williams, Dual extended kalman filter for vehicle state and parameter estimation, Vehicle System Dynamics - VEH SYST DYN 44 (2006) 153–171.

[50] J. Chen, J. Song, L. Li, G. Jia, X. Ran, C. Yang, Ukf-based adaptive variable structure observer for vehicle sideslip with dynamic correction, IET Control Theory Applications 10 (2016) 1641–1652.

[51] J. Yoon, H. Peng, A cost-effective sideslip estimation method using velocity measurements from two gps receivers, IEEE Transactions on Vehicular Technology 63 (2014) 2589–2599.

[52] R. Rajamani, Vehicle Dynamics and Control, Springer, 2012.

[53] J. Villagra, B. d'Andrea-Novel, M. Fliess, H. Mounier, Estimation of longitudinal and lateral vehicle velocities: An algebraic approach, in: 2008 American Control Conference, 2008.

[54] A. Y. Ungoren, H. Peng, A study on lateral speed estimation, International Journal of Vehicle Autonomous Systems, Vol. 2 (2004).

[55] J. Farrelly, P. Wellstead, Estimation of vehicle lateral velocity, in: Proceeding of the 1996 IEEE International Conference on Control Applications IEEE International Conference on Control Applications held together with IEEE International Symposium on Intelligent Control, 1996, pp. 552–557.

[56] L. Imsland, T. A. Johansen, T. I. Fossen, J. C. Kalkkuhl, A. Suissa, Vehicle velocity estimation using modular nonlinear observers, in: Proceedings of the 44th IEEE Conference on Decision and Control, 2005, pp. 6728–6733.

[57] D. Selmanaj, M. Como, G. Panzani, S. Savaresi, Robust Vehicle Sideslip Estimation Based on Kinematic Considerations, in: IFAC PapersOnline, Volume 50 Issue 1, 2017.

[58] M. Bersani, M. Vignati, S. Mentasti, S. Arrigoni, F. Cheli, Vehicle state estimation based on kalman filters, in: 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), 2019, pp. 1–6. doi:10.23919/EETA.2019.8804527.

[59] D. Zermas, I. Izzat, N. Papanikolopoulos, Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 5067–5073.

[60] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools (2000).

[61] K. Wu, E. Otoo, K. Suzuki, Optimizing two-pass connected-component labeling algorithms, Pattern Analysis and Applications 12 (2009) 117–135.

[62] D. F. Crouse, On implementing 2d rectangular assignment algorithms, IEEE Transactions on Aerospace and Electronic Systems 52 (2016) 1679–1696. doi:10.1109/TAES.2016.140952.

An integrated algorithm for ego-vehicle and obstacles state estimation for autonomous driving

Highlights:

- The estimation process is a fundamental task for autonomous driving.
- Estimates are related to the ego-vehicle and the surrounding obstacles.
- The estimation routine handles in proper way the model nonlinearities.
- Estimates are provided in the local reference frame of the road.
- The algorithm performs sensor-fusion and estimation in real-time.

Mattia Bersani} is currently a doctoral candidate at Politecnico di Milano. He obtained the M.S. degree in mechanical engineering from Politecnico di Milano, Milano, Italy, in 2017. His research activity is focused on the definition and implementation of control and state estimation algorithms for autonomous and remote-driven vehicles.

Simone Mentasti is a PhD student at Dipartimento di Elettronica Informazione e Bioingegneria of Politecnico di Milano, Italy. He obtained the M.S. degree in computer science from Università Statale di Milano, Italy, in 2017. His interest focuses on robotics, perception, sensor fusion, sensor calibration and deep learning for autonomous driving cars. His research concerns the development of a sensor fusion framework for autonomous vehicles able to retrieve a uniform representation of the environment surrounding the car.

Stefano Arrigoni} is currently an Assistant Professor at Politecnico di Milano. He received the M.S. degree in mechanical engineering and the Ph.D. degree in applied mechanics both from Politecnico di Milano, Milano, Italy, in 2013 and 2017, respectively. His research interests lie in the area of autonomous vehicles with a focus on trajectory planning techniques and V2V communication.

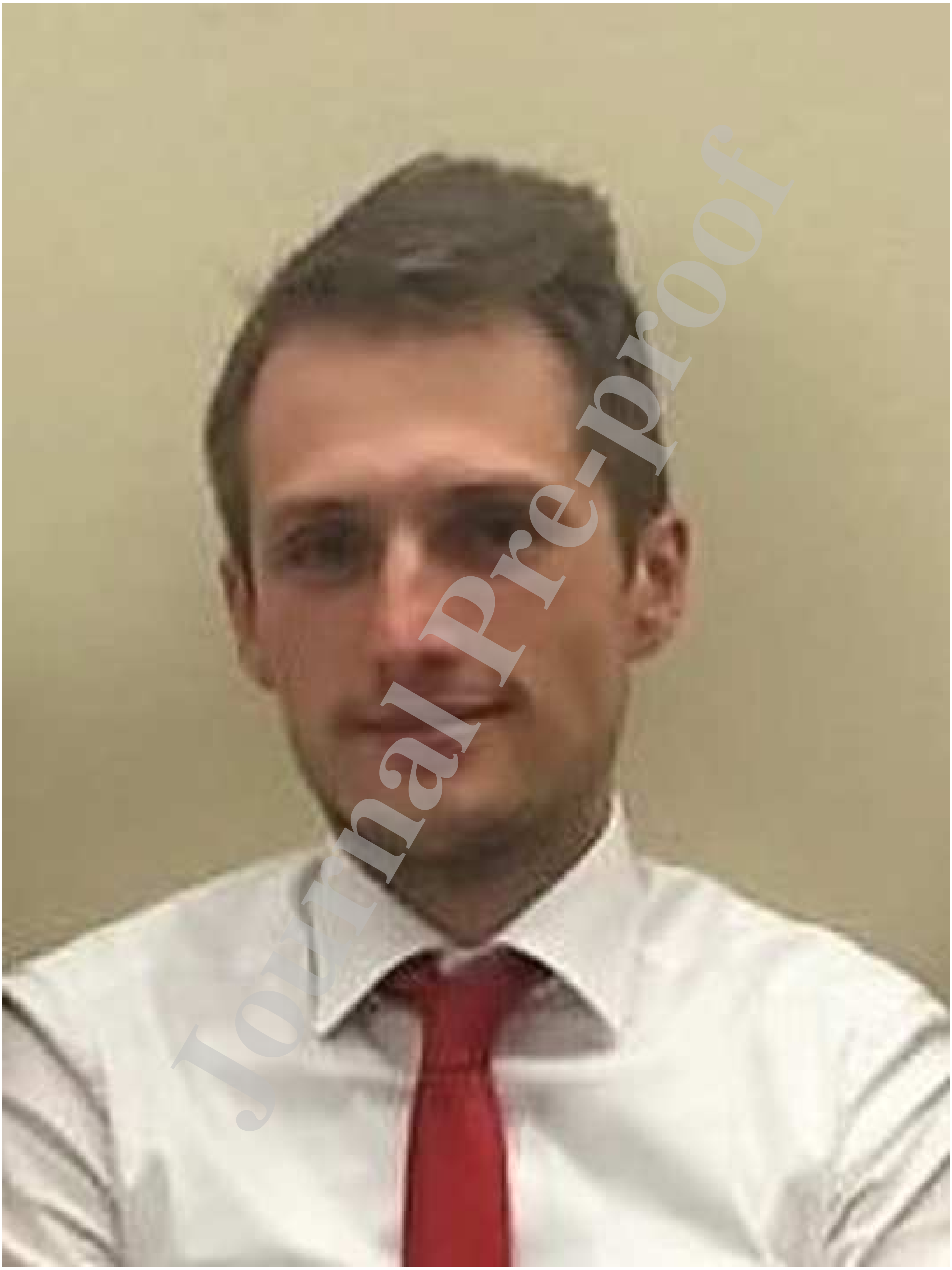
Federico Cheli} received the M.S. degree in mechanical engineering from the Politecnico di Milano, Milano, Italy, in 1981. He is currently a Full Professor at the Department of Mechanical Engineering, Politecnico di Milano and author of more than 380 publications on international journals and conferences. His scientific activity concerns research on vehicle performance, handling and comfort problems, active control, ADAS, and electric and autonomous vehicles. He is a member of the editorial board of the International Journal of Vehicle Performance and International Journal of Vehicle Systems Modeling and Testing.

Matteo Matteucci (PhD) is Associate Professor at Dipartimento di Elettronica Informazione e Bioingegneria of Politecnico di Milano, Italy. In 1999 he got a Laurea degree in Computer Engineering at Politecnico di Milano, in 2002 he got a Master of Science in Knowledge Discovery and Data Mining at Carnegie Mellon University (Pittsburgh, PA), and in 2003 he got a PhD in Computer Engineering and Automation at Politecnico di Milano (Milan, Italy). His main research topics are pattern recognition, machine learning, machine perception, robotics, computer vision and signal processing. His main research interest is in developing, evaluating and applying, in a practical way, techniques for adaptation and learning to autonomous systems interacting with the physical world. He has co-authored more than 150 scientific international publications and he has been the principal investigator in national and international funded research projects on machine learning, autonomous robots, sensor fusion and benchmarking of autonomous and intelligent systems

Pragyan Dahal received B. Tech in Mechanical Engineering from Suresh Gyan Vihar University, India in 2015. Currently he is working towards completing MSc in Mechanical Engineering with Mechatronics specialization at Politecnico Di Milano, Italy. His main research interests include Environment Perception, Multi Object Tracking (MOT) algorithms, Control and Path Planning etc of Autonomous Vehicle.

Michele Vignati received his master degree (2013) and PhD (2017) in mechanical engineering in Politecnico di Milano. From 2019 he is Assistant Professor (RTDA) in the research field of applied mechanics. In particular, he focuses on mechanical systems dynamics and control applied in the automotive field of which he has more than 20 publications in international journals and conferences. In 2018 he won the best paper award for a paper presented at the AVEC'18 international conference.

Author Photo oto;Bersani.jpg



Author Photo

:o;mentasti.jpg

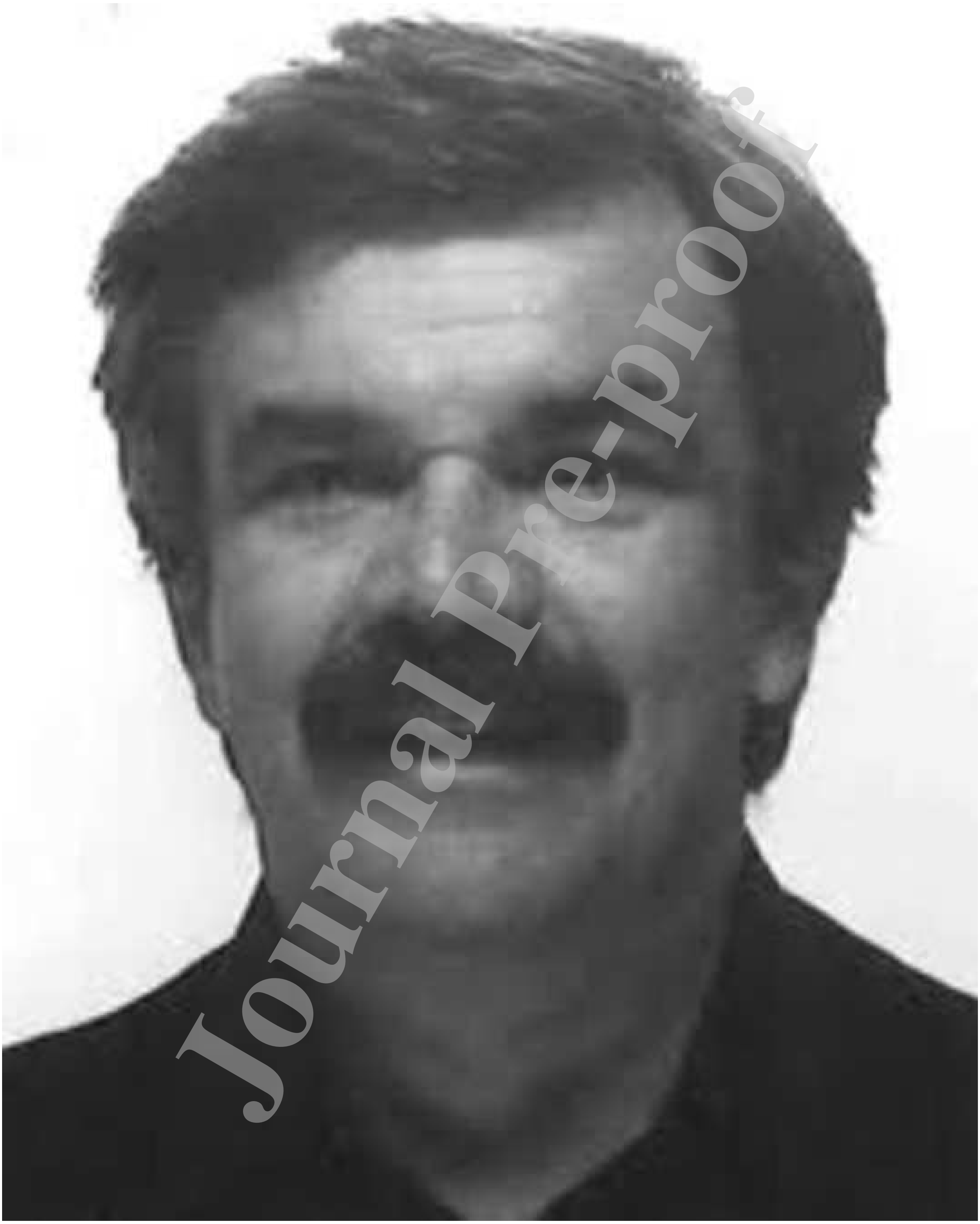

Journal Pre-proof

Author Photo

oto;Vignati.jpg

Author Photo

Photo;Cheli.jpg



Author Photo

;Matteucci.jpg

Author Photo sApp Image
2020-01-15 at 13.56.25.jpeg

Declaration of interests

✘ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

✘ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Mattia Bersani
Simone Mentasti
Pragyan Dahal
Stefano Arrigoni
Michele Vignati
Federico Cheli
Matteo Matteucci