Development of a Detection and Tracking of Moving Vehicles system for 2D LIDAR sensors

Konstantinos Konstantinidis, Mohsen Alirezaei, Sergio Grammatico

***Abstract*—This paper presents the development and evaluation of a Detection and Tracking of Moving Objects (DATMO) system, used for tracking nearby vehicles from a moving car. The developed system takes in raw 2D LIght Detection And Ranging (LIDAR) measurements as input and detects objects of interest by clustering them with the Adaptive Breakpoint Detector algorithm. The resulting clusters are fitted with oriented bounding boxes, by incorporating the Search-Based Rectangle Fitting algorithm. The tracking part of the system receives, extracted from the rectangles, L-shapes and associates them with already tracked vehicles using the Global Nearest Neighbor (GNN) algorithm. However, since LIDAR measures only the distance to surfaces that face the sensor, vehicle appearances change over time. In order to counteract tracking errors that originate from these changes, an L-shape switching algorithm is implemented. The kinematic poses of the tracked vehicles are estimated with two different tracking filters, a Kalman Filter, with a constant velocity model and an Unscented Kalman Filter (KF), with a Coordinated Turn model. The proposed system was evaluated in a simulation environment and the tests revealed that it can reliably estimate the position, speed, heading angle and dimensions of surrounding vehicles. Therefore, it can be reliably used in other research platforms to expand their environment perception capabilities.**

***Index Terms*—Detection, Tracking, DATMO, 2D LIDAR, ROS, Autonomous Vehicles.**

I. Introduction

In recent years, a lot of researchers have focused their efforts on providing solutions to the various problems that need to be addressed before vehicles can reliably perceive their surrounding environment and have divided environmental perception in three different tasks [1]. The first is localization, in which the vehicle localizes itself in the environment by establishing the spatial relationships between itself and stationary objects. The second one is mapping, which builds a map of the environment by establishing the spatial relationships between surrounding static objects. And the last task and the one that the developed system focuses on, is Detection and Tracking of Moving Objects (DATMO), which establishes the spatial and temporal relationships between the vehicle and moving objects. Therefore the aim of the proposed system is, given as input LIght Detection And Ranging (LIDAR) measurements to detect the surrounding moving vehicles and estimate their position (x, y), velocity (v_x, v_y), orientation (ψ), turn rate (ω) and dimensions (Length, Width).

DATMO systems for LIDAR sensors are designed based on three main approaches, the traditional, the model based and the grid based one [2]. The traditional approach, first divides the incoming sensor measurements into clusters and then associates them with objects from previous time instances. In more advanced systems, the clusters are fitted with geometric shapes whose center is then tracked with a parametric Bayesian filter [3], otherwise the geometric mean of each cluster is tracked [4]. The model based approach fits the sensor data directly onto geometric shape models by utilizing particle filters, which also handle data association [5], [6], [7]. Lastly, the grid based approach [8] is based around the construction and use of an occupancy grid, which models the space around the vehicle. The grid cells are then tracked using a Bayesian filter and in some systems, additional object level representations are fitted on top of the grid cells [9]. The development of the proposed system is based on the traditional approach, since it is the most modular and less computationally demanding of the three.

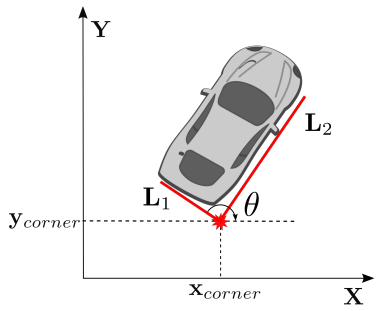
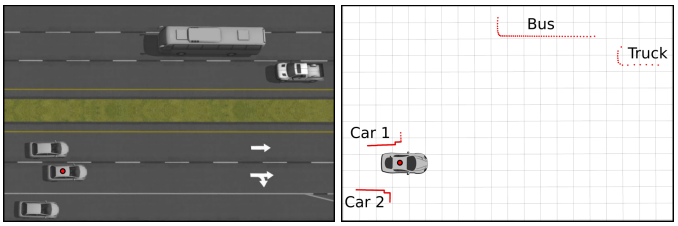


Fig. 1. The five measurements in an L-shape.

The operation of the system is based around extracting and tracking L-shapes, which are used to represent the surrounding vehicles (Fig. 1). L-shapes are defined based on five values, the position of their corner (x_{corner}, y_{corner}), their orientation (θ) and the lengths of their sides (L_1, L_2).

II. Detection of Moving Objects

The main goal of the detection stage is to differentiate moving objects from the LIDAR sensor measurements. LIDAR sensors calculate distance to neighboring objects by emitting a laser beam, capturing its reflection and calculating the distance by measuring the time of flight. The measurements of a LIDAR sensor can be better understood by examining Fig. 2. On the left, there is a screenshot from a simulation and on the right the resulting LIDAR measurements from a sensor on top of the ego vehicle. It should be noted, that this specific time instance will be used throughout this paper to explain and visualize the operation of the developed system.



(a) (b)

Fig. 2. Example of LIDAR data acquisition; (a) image of the simulation environment, (b) visual representation of the LIDAR data acquired at the same time instance.

The first step of the detection stage (Fig. 3) is a segmentation algorithm, which extracts clusters of LIDAR points from the raw LIDAR measurements. These clusters are then passed to a feature extraction algorithm, which extracts geometric shapes from the clusters. Common extracted geometric shapes are lines or rectangles for vehicles, circles for pedestrians and ellipses for bicycles and bikes [10]. Since the focus of the developed system is vehicle tracking, rectangles are fitted onto the clusters. Lastly, L-shapes are derived from the closest corner of every rectangle and those are passed to the tracking stage of the system, which will be presented in the next section.

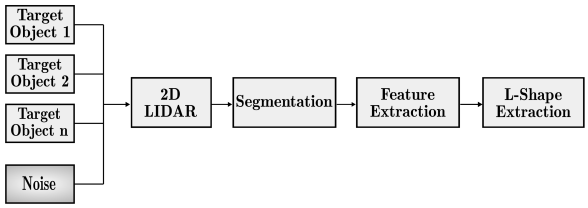


Fig. 3. Flowchart of the vehicle detection stage of the developed system.

A. Segmentation

The segmentation process is responsible for separating the raw LIDAR measurements (Fig. 2b) in groups that correspond to moving objects of the real world that need to be tracked. The algorithm used is the Adaptive Breakpoint Detector Algorithm [11], which clusters the 2D LIDAR point cloud of n points, $X \in R^{n\times 2}$, based on the euclidean distance between consecutive points. Consecutive points p_n and p_{n-1} are clustered together if their euclidean distance is lower than a predefined threshold distance $D_{\max}$.

$$\|p_n - p_{n-1}\| > D_{\max} \tag{1}$$

Otherwise, in case that (1) holds, a new cluster is started whose first point is p_n. In Fig. 4, we can see the threshold circle, which gets drawn around p_{n-1}, with a radius that equals to $D_{\max}$. In this diagram, the next point (p_n) is within the circle and the two points are clustered together. However, if the threshold distance $D_{\max}$ is fixed, the algorithm does not account for the fact that LIDAR point clouds become sparser

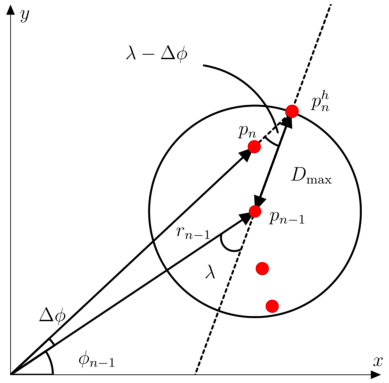


Fig. 4. Visualization of the Adaptive Breakpoint Detector Algorithm [12].

as the distance from the sensor increases. A way to overcome this limitation is by adapting the threshold distance $(D_{\max})$, according to the range distance r_n of the examined point. This is accomplished by drawing a line through the range point p_{n-1}, which represents the worst case for an incidence angle of a real world object that can be detected by the sensor. This line creates an angle λ with respect to the scanning angle ϕ_{n-1}. The maximum range distance r_n^h, for p_{n-1}, is calculated in the following way:

$$r_{n-1} \cdot \sin(\lambda) = r_n^h \cdot \sin(\lambda - \Delta\phi) \tag{2}$$

By reworking the equation above, $\|p_n^h - p_{n-1}\|$ is calculated, which can be used as a threshold distance $(D_{\max})$ in (1).

$$\|p_n^h - p_{n-1}\| = r_{n-1} \cdot \frac{\sin(\Delta\phi)}{\sin(\lambda - \Delta\phi)} \tag{3}$$

Lastly, because the sensor noise is not taken into account, problems can arise when the range distance is small. Therefore, the sensor error variance σ_r is added to the max distance

$$D_{\max} = \|p_n^h - p_{n-1}\| + \sigma_r. \tag{4}$$

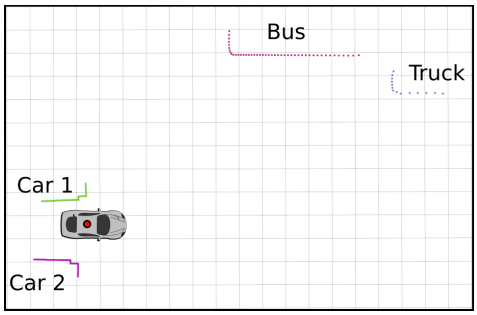


Fig. 5. Cluster segmentation with the Adaptive Breakpoint Detector algorithm.

In Fig. 5 the LIDAR points after the application of this algorithm are visualized and are drawn with a different color for every segmented cluster. We can observe that the algorithm created four different clusters which correspond accurately to the four surrounding vehicles of the simulation.

B. Feature Extraction

The purpose of the Feature Extraction process is to extract geometric shapes from the clustered points, which in this case are rectangles. The algorithm that was chosen and implemented for rectangle extraction is the Search-Based Rectangle Fitting algorithm [13], whose basic idea is to iterate through all possible directions and at each one; find a rectangle that contains all the LIDAR scan points. Afterwards, a performance score is calculated for each rectangle and the rectangle with the highest score is chosen as the best fitting rectangle.

The input of the algorithm is the n points of the examined cluster, $X \in R^{n\times 2}$, while its output are the line representations of the four edges of the fitted rectangle. The search space for θ ranges from $0°$ to $90°$, because the two sides of a rectangle are orthogonal, and therefore only one edge needs to be calculated, since the other is $\theta + \pi/2$. In Fig. 6 an example

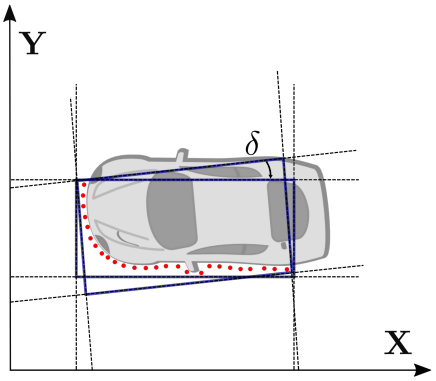


Fig. 6. Operation of the Search-Based Rectangle Fitting Algorithm.

of the algorithm's iterative nature is visualized, in which two rectangles that differ between them by an angle δ are fitted on the LIDAR range points. Although, both rectangles contain all the measurement points, one rectangle is better than the other at representing the vehicle that the points originated from and this is calculated by the performance score. The performance score used in this implementation is the point-to-edges closeness maximization criterion, which calculates how close the rectangle edges are to the LIDAR points.

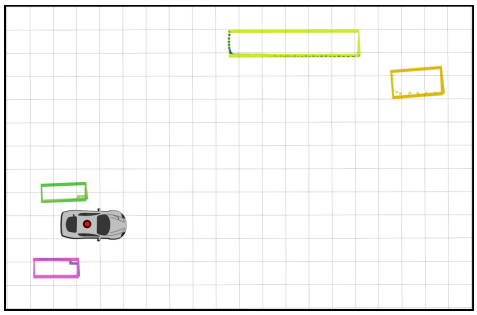


Fig. 7. Rectangle fitting with the Search-Based Rectangle Fitting algorithm.

In Fig. 7 the results of rectangle fitting in the developed system are visualized. It can be seen that the algorithm estimates the shape of the vehicles close to the sensor with high accuracy, but produces some error in the orientation estimation of the truck (orange rectangle).

C. L-shape Extraction

After every cluster of LIDAR points is fitted with a rectangle, an L-shape feature is extracted from every rectangle, mainly for two reasons. First, the information about the closest corner of a neighboring vehicle is important for collision avoidance systems and secondly by extracting L-shapes of the closest sides of neighboring vehicles, their appearance changes can be mitigated in later stages of the developed system.

L-shapes are extracted from the bounding rectangles by choosing as L-shape corner point, the corner point that is closest to the sensor. The two bounding box edges that connect to the corner point are named L_1 and L_2, by following a counterclockwise assignment convention, shown in Fig. 8. The orientation angle (θ) of the L-shape is defined as the orientation of L_1.

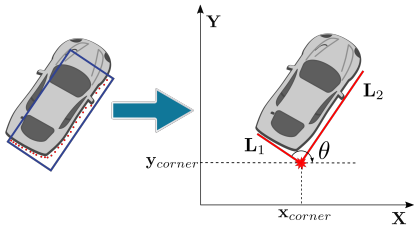


Fig. 8. Conversion of a fitted rectangle to an L-shape feature.

Summarizing the above, the L-shape feature contains five values that are extracted from the bounding box and which will be used in later stages for vehicle tracking. The position of the corner point (x_{corner}, y_{corner}), the lengths L_1, L_2 and the orientation angle θ.

III. Tracking of Moving Vehicles

The objective of the vehicle tracking stage is to estimate as accurately as possible the position, speed and dimensions of all detected vehicles. A flowchart of this stage is given in Fig. 9 and a brief explanation of its operation will be given below.

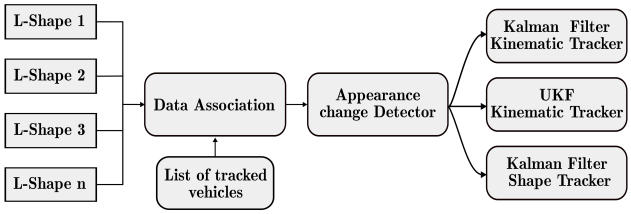


Fig. 9. Flowchart of the vehicle tracking stage of the developed system.

At the left side of the flowchart is the input into the tracking stage, which are the extracted by the detection subsystem L-shapes. The first process of the tracking stage is Data Association, in which the newly received L-shapes are associated with

tracked vehicles from previous timesteps. After the L-shapes are associated with vehicles, it is investigated if the observed corner of the vehicles changed and with it the direction of the associated L-shape. If this is true, the three trackers are updated to reflect the change. Lastly, the position of the L-shape is used to update the two L-shape kinematic trackers and its dimensions and orientation are used for updating the shape tracker. The Kalman Filter (KF) kinematic tracker uses a linear vehicle motion model and its aimed at systems with low computational capabilities, while the Unscented Kalman Filter (UKF) tracker uses a nonlinear motion model and it is geared towards system with higher capabilities.

A. Data Association and Track Management

Data association is the process of associating detection results with already tracked objects by working out which observations were generated by which targets. Data association in multiple vehicle tracking is complicated because of the inherent uncertainty of sensor measurements and the fact that the number of observations does not necessarily correspond to the number of neighboring objects. Furthermore, the true number of surrounding vehicles is difficult to estimate since some vehicles might be temporarily occluded or unobserved.

Track management for multiple object tracking consists of deducing the number of surrounding objects and identifying if each observation corresponds to an already known object, to a new object in the scene or to a false measurement.

1) Data Association: In the proposed system the data association method used is the Global Nearest Neighbor (GNN) filter, which associates clusters with objects based on euclidean distance, while ensuring that each cluster is assigned to at most one object.

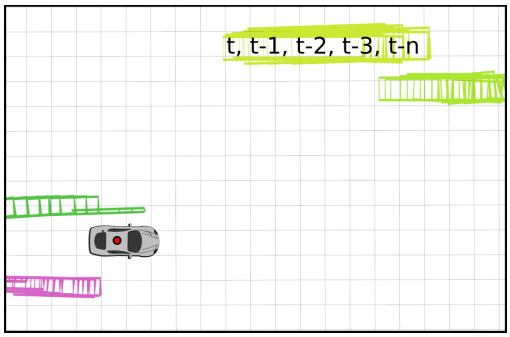


Fig. 10. Data association in the developed system.

Fig. 10 visualizes several consecutive time instances ($t,\ t-1,\ t-2,\ t-n$) from the simulation and the correct association of new measurements to already tracked vehicles can be observed. In case that there was an error, not all four object would have retained the same color throughout the time window, but there would be a color changes at the time instance of the association error.

2) Track Management: Tracks is the name given to objects that are tracked by a DATMO system and track management is the process of managing the list of tracks. The main goal of track management is to reduce the amount of tracked objects, both for reducing the amount of computations performed at each timestep but also for preventing false data associations. The track management system that is used is simple in its design and operates in the following way. After every measurement update and clustering step, all the clusters not associated with any already tracked object are used to initiate new tracks. The tracks that are associated with newly detected clusters are unaffected, while the not associated tracks are immediately deleted.

B. L-shape Tracker

In order to track the position of the L-shape corner, the proposed system implements two solutions. The first one uses a Kalman Filter for tracking the corner of the L-shape, while the second one uses an Unscented Kalman Filter (UKF). The first approach is based on the work presented in [14], while the second one is novel. The system implements two solutions for two main reasons: the first being for comparing the accuracy of the Kalman Filter and the UKF in this particular application. And the second one is, providing the users of the system with options, so they can make a choice depending on the accuracy demanded by their application and the available computational resources of their platform. The dimensions ($L_1,\ L_2$), orientation (θ) and turn rate (ω) of the L-shape are tracked by a separate Kalman Filter.

1) Kalman Filter Kinematic Tracker: The Kalman Filter used for tracking the motion of the corner point uses a Constant Velocity (CV) model to estimate position and velocities, with the following state vector $\mathbf{x}_{CV}$.

$$x_{CV} = \begin{bmatrix} x_{corner} & y_{corner} & v_x & v_y \end{bmatrix}^T \tag{5}$$

The kinematic model A_{CV} that is used to track the position

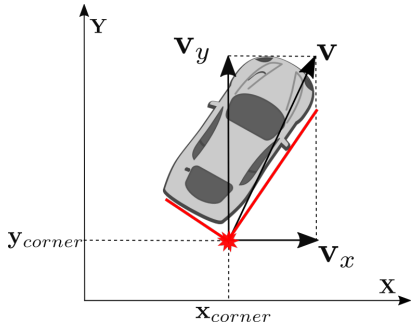


Fig. 11. The kinematic model models the motion of the corner point.

(x_{corner}, y_{corner}) and velocities v_x, v_y of the corner point, as show in Fig. 11, is the following:

$$A_{CV} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where T is the sampling time. Given that only the position of the corner point is measured, the measurement vector is:

$$z_{CV} = \begin{bmatrix} x_{corner} & y_{corner} \end{bmatrix}^T \tag{7}$$

and consequently the measurement model is the following:

$$H_{CV} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{8}$$

2) UKF Kinematic Tracker: The kinematic tracker that uses a UKF has one main advantage over the one presented before and that is that nonlinear motion models can be used. The implemented one uses as kinematic function f_{CTM}, a Coordinated Turn Model (CTM) [15], which in addition to the position and speed tracks also the turn rate (ω). Therefore, the state vector $\mathbf{x}_{CTM}$ is:

$$\mathbf{x}_{CTM} = \begin{bmatrix} x & y & v_x & v_y & \omega \end{bmatrix}^T \tag{9}$$

And its kinematic function f_{CTM} is the following:

$$f_{CTM} = \begin{bmatrix} x + \frac{v_x}{\omega}\sin(\omega T) - \frac{v_y}{\omega}(1-\cos(\omega T)) \\ y + \frac{v_x}{\omega}(1-\cos(\omega T)) + \frac{v_y}{\omega}\sin(\omega T) \\ v_x\cos(\omega T) - v_y\sin(\omega T) \\ v_x\sin(\omega T) + v_y\cos(\omega T) \\ \omega \end{bmatrix}. \tag{10}$$

The measurement vector and matrices are similar to the ones used in the Kalman Filter, since the available measurements are the same.

$$\begin{aligned} z_{CTM} &= \begin{bmatrix} x_{corner} & y_{corner} \end{bmatrix}^T \\ H_{CTM} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{11}$$

3) Shape Tracker: The shape of the target vehicle is tracked using a Kalman Filter and a state vector composed of line lengths (L_1, L_2), the orientation of $L_1(\theta)$ and the turn rate (ω). Those states are visualized in Fig. 12 and are contained in vector $\mathbf{x}_s$.

$$\mathbf{x}_S = \begin{bmatrix} L_1 & L_2 & \theta & \omega \end{bmatrix}^T \tag{12}$$

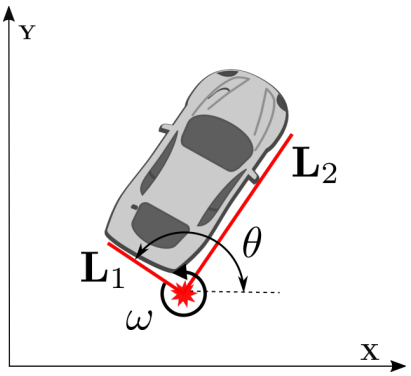


Fig. 12. The shape filter estimates the orientation, turn rate and size of the L-shape.

For estimating the vehicle's shape, a static model is applied to the line lengths (L_1, L_2) based on the assumption that the vehicle size does not change over time. For estimating the L-shape's yaw, and since the yaw rate does not change particularly fast, a constant turn rate model is chosen. The two above models are combined in a single process matrix:

$$A_S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{13}$$

where T the sampling time and A_S is the process matrix containing the static model for the line lengths and the constant turn rate model for the orientation and the yaw rate.

Among the states of the shape model, only the yaw rate is not contained in the L-shape and therefore the measurement vector and model are the following:

$$\begin{aligned} z_S &= \begin{bmatrix} L_1 & L_2 & \theta \end{bmatrix}^T \\ H_S &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \end{aligned} \tag{14}$$

4) Corner Point Switching: The designed system is tracking L-shapes, which represent the closest corner of observed vehicles. However, while the ego vehicle and the observed ones are moving, it is expected that the closest corners of the other vehicles will be periodically changing and therefore the extracted L-shapes should also be changing. All the possible extracted L-shapes from a vehicle are drawn in Figure 13 and are marked by values C1 to C4.

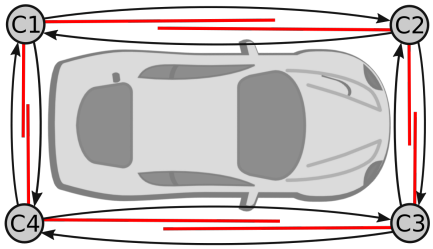
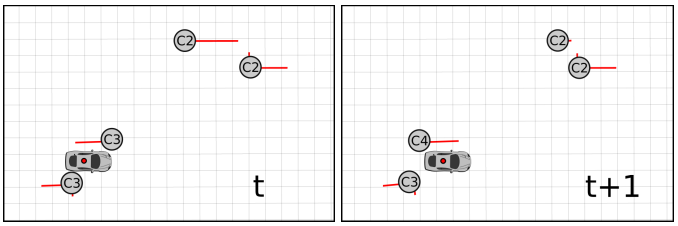


Fig. 13. Visualization of all the corner points of a vehicle (C1-C4) and of the clockwise and counter clockwise changes between them (black arrows).

An example of a corner change occurrence can be observed in Figure 14, in which two instances from the simulation are given side by side. In the first time instance (t), the overtaking vehicle at the left of the ego-vehicle is tracked by its front-right corner (C3). As it moves ahead in the second time instance $(t+1)$, the closest corner changes to the one at the vehicle's rear-right corner (C4).

In the proposed system, corner point changes were detected based on the Mahalanobis distances of the new measurement to the already tracked corner point and the two neighboring ones. If for example, the already tracked corner point is C3, with state vector $(\mathbf{c_3})$, then the state vectors of corner points C2 $(\mathbf{c_2})$ and C4 $(\mathbf{c_4})$ are calculated and their values are compared with the state vector of the new L-shape measurement $(\mathbf{m})$, via the Mahalanobis distance. The Mahalanobis distance



(a) (b)

Fig. 14. Clockwise change of closest corner point, from corner C3 to corner C4.

(D_M), which is a multivariate distance metric, is calculated based on the following formula:

$$\begin{aligned} D_M^2 &= (\mathbf{c}-\mathbf{m})^T \cdot P^{-1} \cdot (\mathbf{c}-\mathbf{m}) \\ \mathbf{c} &= [x_{corner},\ y_{corner},\ \theta,\ L_1,\ L_2,]^T \\ \mathbf{m} &= [x_{corner}, y_{corner}, \theta, L_1, L_2,]^T \end{aligned} \tag{15}$$

where P is the covariance matrix of the already tracked corner point.

If the shortest Mahalanobis distance is between the measurement vector $\mathbf{m}$ and the previously tracked corner vector, for example $\mathbf{c_3}$, then no corner switch took place. However, if the shortest distance is with $\mathbf{c_4}$, a clockwise switch took place and if it is with $\mathbf{c_2}$, a counter clockwise switch took place. In the example of Figure 14, we can observe that a clockwise corner switch between corners C3 and C4 occurred. In such cases, the filters that track the L-shape cannot be immediately updated because the new L-shape measurement corresponds to another corner than the one that the filter states are tracking. Therefore, the filter states should be altered, so that the represent the same corner with the new measurement.

In case that a clockwise switch is detected, the filter states are changed in the following way:

$$\begin{aligned} x_{\text{corner}}^{Cj} &= x_{\text{corner}}^{Ci} + L_1^{Ci}\cos\theta^{Ci} \\ y_{\text{corner}}^{Cj} &= y_{\text{corner}}^{Ci} + L_1^{Ci}\sin\theta^{Ci} \\ v_x^{Cj} &= v_x^{Ci} + L_1^{Ci}\omega\sin\theta^{Ci} \\ v_y^{Cj} &= v_y^{Ci} + L_1^{Ci}\omega\cos\theta^{Ci} \\ \theta^{Cj} &= \theta^{Ci} - \pi/2 \\ L_1^{Cj} &= L_2^{Ci} \\ L_2^{Cj} &= L_1^{Ci} \\ &\text{where} \begin{cases} j = i+1, & i<4 \\ j = 1, & i=4 \end{cases} \end{aligned} \tag{16}$$

where the superscripts Ci and Cj represents the corner number before and after the model change.

In case that a counter clockwise switch is detected the kinematic and shape states are changed in the following way:

$$\begin{aligned} x_{\text{corner}}^{Cj} &= x_{\text{corner}}^{Ci} + L_2^{Ci}\sin\theta^{Ci} \\ y_{\text{corner}}^{Cj} &= y_{\text{corner}}^{Ci} + L_2^{Ci}\cos\theta^{Ci} \\ v_x^{Cj} &= v_x^{Ci} + L_2^{Ci}\omega\cos\theta^{Ci} \\ v_y^{Cj} &= v_y^{Ci} + L_2^{Ci}\omega\sin\theta^{Ci} \\ \theta^{Cj} &= \theta^{Ci} + \pi/2 \\ L_1^{Cj} &= L_2^{Ci} \\ L_2^{Cj} &= L_1^{Ci} \\ &\text{where} \begin{cases} j = i-1, & i>1 \\ j = 4, & i=1 \end{cases} \end{aligned} \tag{17}$$

5) L-shape to Box Model Conversion: In the L-shape to box model conversion stage the kinematic state is changed from corner point motion to vehicle motion. The position of the center of the box model can be calculated by using the geometric information of the shape model

$$\begin{aligned} x_{center} &= x_{\text{corner}} + \varepsilon_x, \quad \text{where } \varepsilon_x = (L_1\cos\theta + L_2\sin\theta)/2 \\ y_{center} &= y_{\text{corner}} + \varepsilon_y, \quad \text{where } \varepsilon_y = (L_1\sin\theta - L_2\cos\theta)/2. \end{aligned} \tag{18}$$

Since the velocity of the corner point is the sum of the target velocity and the tangential velocity, the rotational motion of the corner point includes both translational velocity and rotational velocity. The rotational motion is assumed as a uniform circular motion, since a constant turn rate model for the shape model has already been assumed. Therefore, an equation for tangential velocity can be derived by using the distance from the center as the radius (r) of the circular motion.

$$\begin{aligned} v_{x,\text{center}} &= v_{x,\text{corner}} - r\omega\cos\alpha \\ v_{y,\text{center}} &= v_{y,\text{corner}} - r\omega\sin\alpha \\ &\text{where } \alpha = \tan^{-1}(\varepsilon_y/\varepsilon_x) - \pi/2. \end{aligned} \tag{19}$$

A vehicle's yaw (ψ) is typically difficult to estimate from an L-shape because of the ambiguities, since it is hard to judge which is the front part and which is the rear part of the vehicle. Therefore, it would be reasonable to keep the following four hypotheses:

$$\psi_i = \operatorname{atan2}(n_2, n_1) + i\frac{\pi}{2}, i \in \{0,1,2,3\} \tag{20}$$

where ψ_i represents the yaw angle of the vehicle [16]. Next, for calculating the yaw of the vehicle, these four angles are compared with the angle of the vehicle's speed and the one with the least absolute difference from the speed's orientation is chosen as the vehicle yaw. The turn rate of the box model is taken equal to the turn rate (ω) of the tracked L-shape.

In Fig. 15 the box models calculated by the system are visualized, while arrows are used to represent the estimated velocities of the surrounding vehicles. The starting point of those arrows are the estimated centers of the surrounding vehicles.

It can additionally be observed, that the dimensions of the box models are bigger than the extracted rectangles, since they

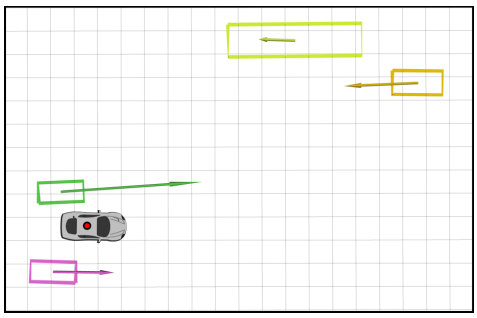


Fig. 15. Visualization of the estimated box models.

are based on the estimations of the shape filter and that the estimated orientation of the truck at the top right corner is closer to the true value, than the one estimated by the rectangle fitting algorithm (Fig. 7).

IV. Experimental Evaluation

In this section, a simulation experiment[1] that was used to test and evaluate the proposed system will be presented and the evaluation results will be explained. During this simulation experiment the output of the system and the reference measurements are both recorded and they are later analyzed to assess the accuracy of the proposed system. The reference measurements provided by the simulation environment are very accurate and with sufficiently higher frequency (100 Hz) than the estimations of the system (12 Hz) and therefore they are used as ground truth data. For evaluating the tracking performance, the system's estimates for the position (x, y), velocities (v_x, v_y), orientation (ψ), turn rate (ω) and dimensions (Length, Width) of the box models are plotted against the ground truth data, which also refer to the center of the vehicles.

In Fig. 2a, which is a snapshot of the simulation environment during this experiment, we can observe that there a total of five vehicles, from which the three at the bottom lanes are controlled via joystick input, while the bus and truck at the top lanes are following straight paths. The ego vehicle is the Prius in the middle lane and during this experiment it overtakes the car that drives on the emergency lane. At the same time, the third car which also starts in the middle lane, changes to the left lane, overtakes the ego-vehicle and then merges back in the middle lane in front of it.

The estimations of the system are plotted against the ground truth data on Fig. 16 for the car at the emergency lane and at Fig. 17 for the car at the left lane. At the first two rows of the figures there are graphs for the position $(x,\ y)$ and velocities (v_x, v_y) of the tracked cars center, which are estimated by the kinematic filters. At the bottom two rows there are graphs of the orientation (ψ), turn rate (ω) and dimensions of the vehicles, which are estimated by the shape Kalman Filter.

In Fig. 16, we can observe that both filters produce identical estimates for the position $(x,\ y)$ of the car in the emergency lane. However, the velocities estimated by the two filters are different, with the KF overshooting the reference and the UKF undershooting it. This can be explained, by the different process models that the filters employ. The orientation of the vehicle is estimated with high accuracy by the shape filter, after the vehicle starts moving. The turn rate however is not be estimated accurately since its abrupt changes were not tracked correctly by the filter. Lastly, it is evident that the accuracy of length and width estimation, depends on which side the incoming measurements represent. Therefore, when the length is measured the length estimation is accurate and when the width is measured its estimation is more accurate. The two spikes at the beginning of those diagrams can be explained by the fact that, the orientation of the vehicle can not estimated when the vehicles are stationary.

The results for the overtaking car are drawn in Fig. 17, where it can be observed that the system has generally similar performance. However, in the (v_y) graph, a spike can be seen, which is attributed to two rapid corner switches that occur while the vehicles are almost stationary. In both cases, it is evident that the UKF offers no significant benefits to the KF, and that is probably due to the motion of the two vehicles, which is generally straight, while the model of the UKF estimates the kinematics of constant turns.

[1]A video of the experiment can be found at: https://youtu.be/JrbJjmIv730

V. Conclusion

The DATMO system presented in this paper, encompasses the complete process of multi-object tracking; it receives 2D LIDAR measurements as input and estimates the kinematic state and dimensions of surrounding vehicles. The detection part of the system differentiates objects in the LIDAR measurements using the Adaptive Breakpoint Detector algorithm and calculates rectangles around the detected objects with the Search-Based Rectangle Fitting algorithm. The tracking part of the system receives L-shapes and associates them with already tracked vehicles, using the Global Nearest Neighbor algorithm. It estimates kinematic poses with two different tracking filters, a Kalman filter, with a constant velocity model and a UKF, with a nonlinear constant turn kinematic model.

The proposed system was evaluated in a simulation experiment and it was shown that it can reliably estimate the position, speed, heading angle, turn rate and dimensions of surrounding vehicles and that it successfully mitigates appearance changes. Since the system is able to run in a real-time manner, it is expected to be applicable for a variety of algorithm development projects using low-cost platforms.

References

[1] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.

[2] A. Petrovskaya, M. Perrollaz, L. Oliveira, L. Spinello, R. Triebel, A. Makris, J.-D. Yoder, C. Laugier, U. Nunes, and P. Bessiere, "Awareness of road scene participants for autonomous driving," in *Handbook of Intelligent Vehicles*. Springer, 2012, pp. 1383–1432.

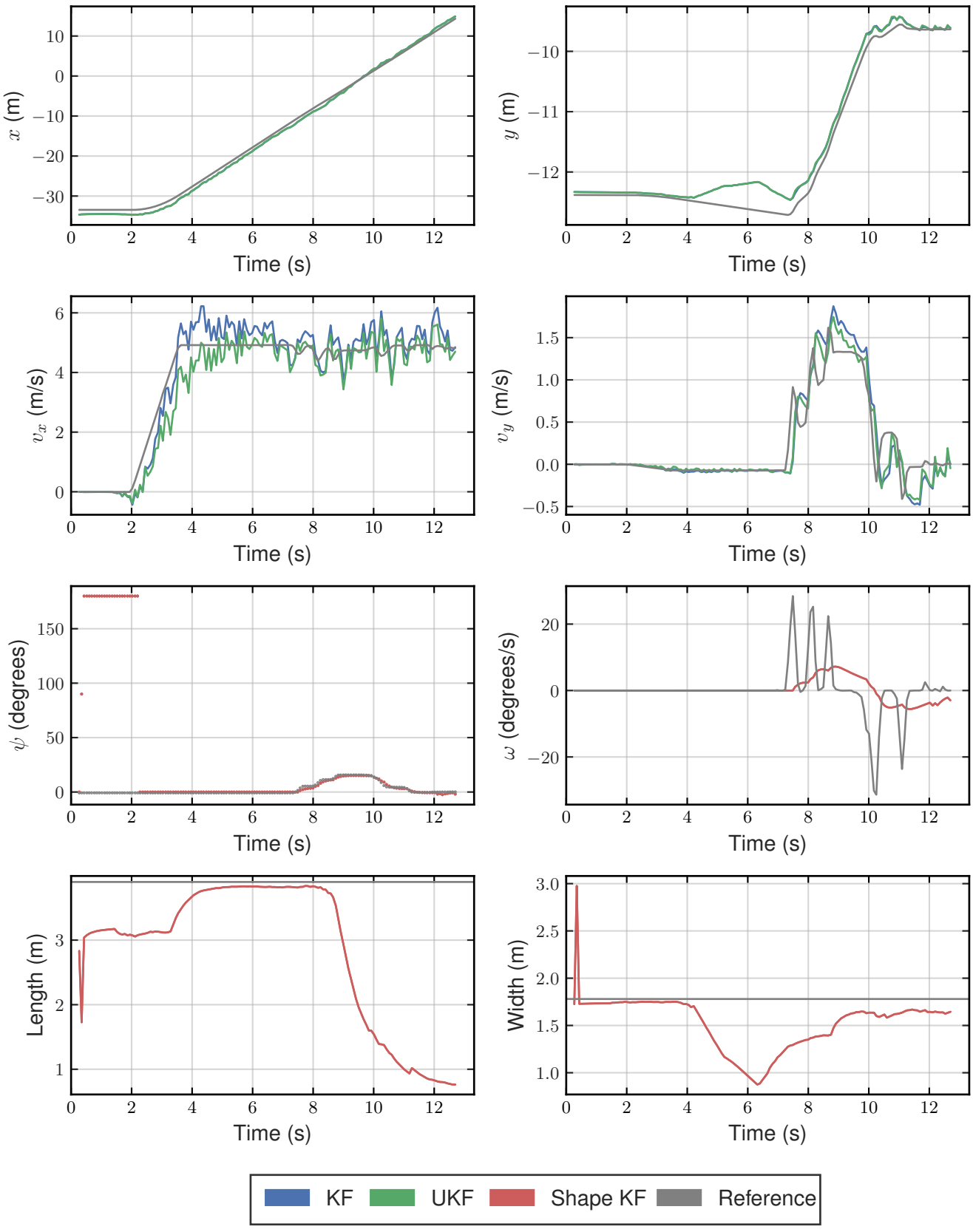


Fig. 16. Estimated states by the developed system against the simulation ground truth, for the car at the left of the ego-vehicle.

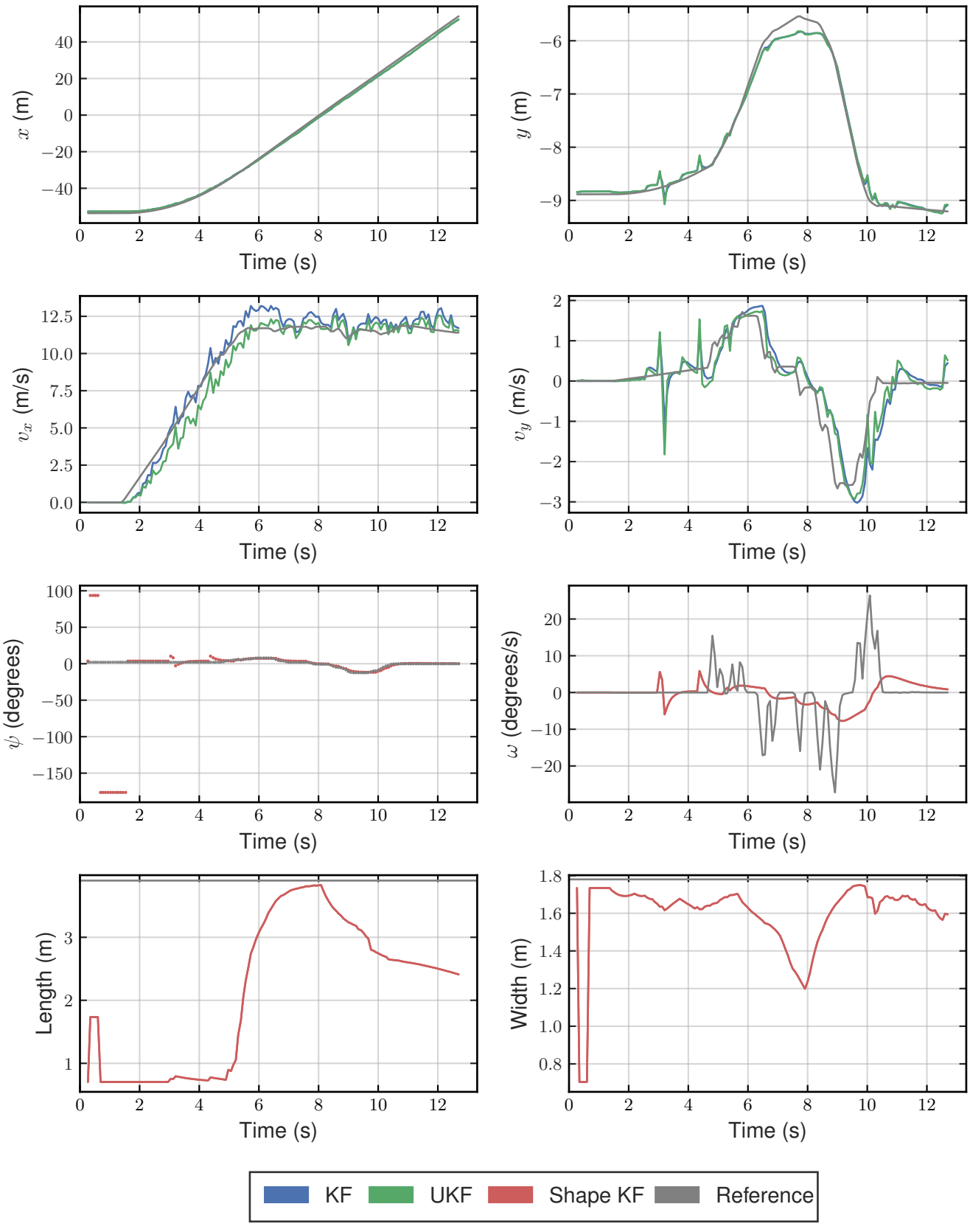


Fig. 17. Estimated states by the developed system against the simulation ground truth, for the car at the left of the ego-vehicle.

[3] J. Choi, S. Ulbrich, B. Lichte, and M. Maurer, "Multi-target tracking using a 3d-lidar sensor for autonomous vehicles," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 881–886.

[4] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[5] A. Petrovskaya and S. Thrun, "Model based vehicle tracking for autonomous driving in urban environments," *Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland*, vol. 34, 2008.

[6] T. Chen, R. Wang, B. Dai, D. Liu, and J. Song, "Likelihood-field-model-based dynamic vehicle detection and tracking for self-driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3142–3158, 2016.

[7] K. Granstrom, M. Baum, and S. Reuter, "Extended object tracking: Introduction, overview and applications," *arXiv preprint arXiv:1604.00970*, 2016.

[8] T.-D. Vu, J. Burlet, and O. Aycard, "Grid-based localization and local mapping with moving object detection and tracking," *Information Fusion*, vol. 12, no. 1, pp. 58–69, 2011.

[9] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1331–1342, 2011.

[10] N. Kaempchen, M. Buehler, and K. Dietmayer, "Feature-level fusion for free-form object tracking using laserscanner and video," in *Intelligent vehicles symposium, 2005. Proceedings. IEEE*. IEEE, 2005, pp. 453–458.

[11] G. A. Borges and M.-J. Aldon, "Line extraction in 2d range images for mobile robotics," *Journal of intelligent and Robotic Systems*, vol. 40, no. 3, pp. 267–297, 2004.

[12] T. Johansson and O. Wellenstam, "Lidar clustering and shape extraction for automotive applications," Master's thesis, Chalmers University of Technology, 2017.

[13] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, "Efficient l-shape fitting for vehicle detection using laser scanners," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 54–59.

[14] D. Kim, K. Jo, M. Lee, and M. Sunwoo, "L-shape model switching-based precise motion tracking of moving vehicles using laser scanners," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 598–612, 2018.

[15] M. Roth, G. Hendeby, and F. Gustafsson, "Ekf/ukf maneuvering target tracking using coordinated turn models with polar/cartesian velocity," in *17th International Conference on Information Fusion (FUSION)*. IEEE, 2014, pp. 1–8.

[16] X. Shen, S. Pendleton, and M. H. Ang, "Efficient l-shape fitting of laser scanner data for vehicle pose estimation," in *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. IEEE, 2015, pp. 173–178.