Optical Flow Based Detection and Tracking of Moving Objects for Autonomous Vehicles

MReza Alipour Sormoli[1], Mehrdad Dianati[1], *Senior Member, IEEE,* Sajjad Mozaffari[1], and Roger Woodman[1]

***Abstract*—Accurate velocity estimation of surrounding moving objects and their trajectories are critical elements of perception systems in Automated/Autonomous Vehicles (AVs) with a direct impact on their safety. These are non-trivial problems due to the diverse types and sizes of such objects and their dynamic and random behaviour. Recent point cloud based solutions often use Iterative Closest Point (ICP) techniques, which are known to have certain limitations. For example, their computational costs are high due to their iterative nature, and their estimation error often deteriorates as the relative velocities of the target objects increase (>2 m/sec). Motivated by such shortcomings, this paper first proposes a novel Detection and Tracking of Moving Objects (DATMO) for AVs based on an optical flow technique, which is proven to be computationally efficient and highly accurate for such problems. This is achieved by representing the driving scenario as a vector field and applying vector calculus theories to ensure spatiotemporal continuity. We also report the results of a comprehensive performance evaluation of the proposed DATMO technique, carried out in this study using synthetic and real-world data. The results of this study demonstrate the superiority of the proposed technique, compared to the DATMO techniques in the literature, in terms of estimation accuracy and processing time in a wide range of relative velocities of moving objects. Finally, we evaluate and discuss the sensitivity of the estimation error of the proposed DATMO technique to various system and environmental parameters, as well as the relative velocities of the moving objects.**

***Index Terms*—Autonomous vehicles, optical flow, LiDAR, point cloud, DATMO, MODT, state estimation.**

I. Introduction

ACCURATE, reliable and fast perception of the surrounding environment is one of the most important technical challenges in the safe deployment of Autonomous/Automated Vehicle (AV) technologies. This problem includes the detection of surrounding objects and estimation of their states, i.e., their position and velocity. A particularly important element of this problem is associated with the Detection and Tracking of Moving Objects (DATMO), a.k.a. Moving Object Detection and Tracking (MODT) in some studies of the literature [1].

There is a wide range of DATMO techniques in the literature tailored for AVs that use camera perception sensors [2]–[4], LiDAR perception sensors [5]–[7] and Radar perception sensors [8]. LiDAR perception sensors are particularly popular for AVs as they inherently provide a wide field of view (FOV) and robust point clouds that can be used for highly accurate range estimations. Therefore, in this study, we focus on developing a DATMO technique primarily tailored for LiDAR perception sensors. However, we believe such a technique can be easily adapted to any type of perception sensor, such as depth cameras, that provides a point cloud output.

When it comes to designing a DATMO technique, minimising its processing cost/time and estimation error are two significant challenges. If an AV uses a LiDAR perception sensor, the velocity of the surrounding moving objects can be calculated by corresponding two consecutive point cloud scans. In traditional approaches, an object detection function is used as the first processing stage to identify objects in two consecutive LiDAR scans. Then, a tracking algorithm is applied to compute the velocity for the objects of interest [9]. Although these techniques are computationally efficient, their accuracy depends on the accuracy of the underlying object detection algorithms. While enhancing various elements of the techniques, for example, by utilising the geometric models of the moving objects in the detection process, can improve the performance of this category of DATMO techniques, such techniques do not perform well in many scenarios [21]. For example, if a vehicle travels at the speed of 90 km/hr on a highway, the lateral velocity estimation error of such techniques can exceed 2 m/sec. This is not regarded as an acceptable input to the planning modules that determine cut-in/cut-out intentions of vulnerable road users [22].

The problem can be exacerbated because the geometric models of the moving objects can significantly vary for various road users. This can have a direct impact on the estimation errors of the aforementioned DATMO techniques. To address these problems, a different category of DATMO techniques has emerged in the literature [23]–[26]. These techniques often use point cloud registration algorithms such as Iterative Closest Points (ICP) [22] and track all moving points in the point cloud [23]; hence, they are more accurate in velocity estimation. However, these DATMO techniques are computationally expensive because of their iterative nature [28]. Furthermore, the performance of the underlying point cloud registration methods can deteriorate when the deviation between two consecutive point cloud scans increases. For instance, if the relative speed of the Ego Vehicle (EV) and a target object of interest is high (e.g., 12 km/hr), the dislocation of the consecutive LiDAR scans is usually large, which can result in a large error in ICP-based point cloud registration algorithms [21]. This can result in poor performance of the latter category of the DATMO techniques.

Motivated by the above challenges, in this paper, a novel DATMO technique is proposed for AVs that use LiDAR

[1] Mehrdad Dianati holds part-time professorial posts at the School of Electronics, Electrical Engineering and Computer Science (EEECS), Queen's University of Belfast and WMG at the University of Warwick. Other authors are with WMG, University of Warwick, e-mail: {mreza.alipour, m.dianati, sajjad.mozaffari, r.woodman}@warwick.ac.uk, m.dianati@qub.ac.uk

#Corresponding author: mreza.alipour@warwick.ac.uk

TABLE I
A BRIEF REVIEW OF THE LIDAR-BASED DATMO METHODS IN THE LITERATURE.
L: LEARNING-BASED, ICP: INTERACTIVE CLOSEST POINT METHOD, G: GRID-BASED REPRESENTATION

Category		References	Advantages	Disadvantages
Detection-Based Tracking	Model-Based	$[1]_l$ [9]	• independent detection and tracking • fusing different senors in detection level	• tracking performance relies on detection/classification • difficult to detect and track objects with unknown geometries
	Model-Free	[10] [11] $[12]^g_l$ $[13]_l$ $[14]^g$	• tracking object with different geometries	• more false-negatives (FNs) in detection and tracking
Direct Tracking	Model-Based	[15] [16] $[17]^g$ [18]	• sensor's physical model is considered → more accurate DATMO • geometry of the objects is tracked → accurate state estimation	• tracking performance decreases for objects with different shapes
	Model-Free (Point-Based)	$[7]^g$ $[19]^g$ [20] $[21]_{icp}$ $[22]_{icp}$ $[23]_{icp}$ $[24]_{icp}$ $[25]_l$ [26] $[27]^g_{icp+l}$	• tracking all scanned points • DATMO performance doesn't rely on geometric shape	• high computational cost • higher the relative velocity for moving objects → lower tracking performance

perception sensors. The proposed technique is inspired by *optical flow* algorithm [29]. In our approach, the 3D LiDAR scans are initially converted to 2.5D motion grids (Fig. 3) inspired by [19]; then, the 2D velocity of each cell in the grid is estimated by comparing two consecutive LiDAR scans. In the next step, a series of grid mask filters such as temporal and rigid-body continuity filters are applied to eliminate false positive detection. The LiDAR points are classified based on their velocity vectors, and each class is associated with a moving object. Finally, a Kalman Filter is used to track the velocity and position of the detected moving objects, considering their dynamic model. The main contributions of this paper are summarized as follows:

- Adopting optical flow technique to process the 3D point cloud data instead of complex ICP algorithms. This is used to generate a grid-based velocity vector field representing a dynamic driving environment.
- Introducing a two-layer filter applied to the velocity vector field eliminating the false positives and erroneous vectors. These filters are designed based on the spatial continuity of the vector field (rigid-body assumption) and temporal propagation to improve the estimation performance results.
- Introducing novel error model for velocity estimation as a function of a configuration set for target vehicles (TVs) w.r.t the ego vehicle (EV). This offers further insights to be incorporated into the downstream modules such as motion planning/prediction in the autonomous vehicle framework.

The performance of the proposed technique compared to the ICP-based methods, such as the one in [21], [27], model-free [14] and model-based indirect tracking methods [18], is evaluated in two steps. First, the compared DATMO techniques are evaluated on a synthetic dataset generated in MATLAB scenario designer, where various driving contexts are considered. In the next step, the KITTI tracking dataset from real driving scenarios [30] is used. Each one of the synthetic and KITTI datasets serves a different purpose in our study. The synthetic dataset enables generating a wide range of driving scenarios and various target vehicles (shape, velocity, dimension, etc.), which is not practically feasible in real-world data collection campaigns. The flexibility of this type of dataset becomes even more important when it comes to analysing error sensitivity to different factors that are easy to change or sweep in synthetically generated scenarios. On the other hand, testing the proposed DATMO techniques with data collected by contemporary sensors in real-world conditions helped us validate its performance in the real world. Comparing the estimation error distribution shows that the proposed DATMO outperform the state-of-the-art in both speed and yaw angle estimation. Moreover, the computational cost (without parallel calculations) shows improvements of about 10%, whereas parallelising the proposed method is easily available and could improve this metric even more significantly. The proposed error sensitivity analysis also revealed a meaningful correlation between the configuration of the TV and estimation error from which the researcher could benefit to develop motion planning/prediction algorithms.

The rest of the paper is organized as follows. An overview of the existing related work in the literature is described in Section II. The system model and problem formulation are given in Section III. The proposed DATMO method is explained in Section IV. The performance evaluation methodology and results are discussed in Sections V to VI. Finally, the key findings and conclusions of the study are given in Section.VII.

II. RELATED WORKS

In order to review available point cloud based DATMO/-MODT approaches in the literature, in this paper, they are categorized into two main classes: 1) *detection-based methods*

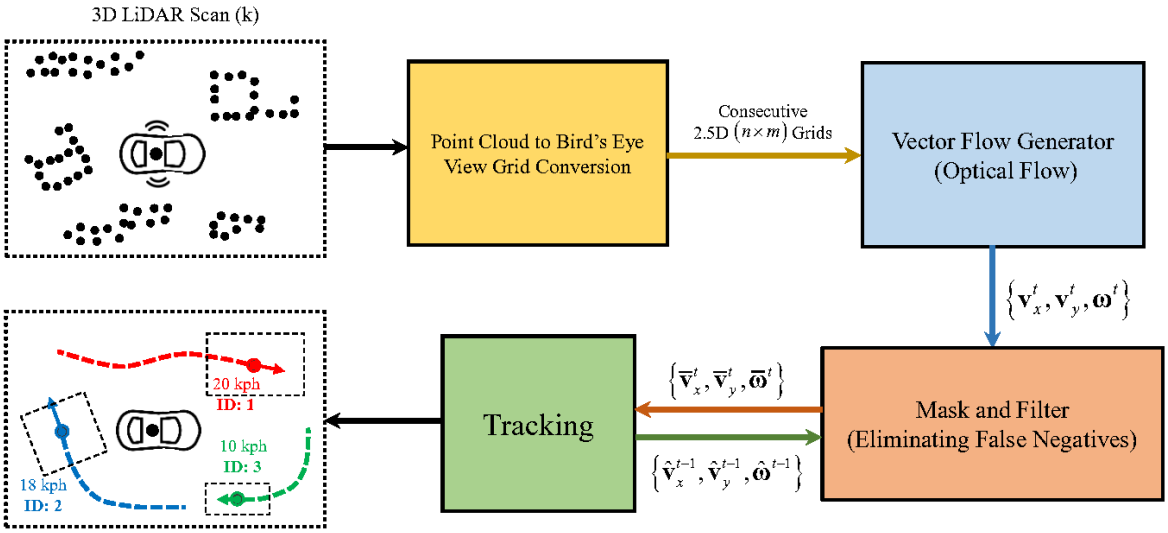

Fig. 1. High-level schematic system diagram of optical flow based DATMO for AVs. ω, $\mathbf{v_x}$, and $\mathbf{v_y}$ all are $n \times m$ matrices. The same colour code (blocks and signals) is used to expand and explain in different sections

also known as *traditional methods* [31]; 2) *direct tracking methods* which is further divided into *model-based* and *point-based* approaches.

A. Detection-Based Tracking

The *detection-based* or indirect algorithms track the abstracted objects, patterns, bounding boxes, or clusters [10], [11] by applying different filters such as variants of the Kalman filter or particle filter. Therefore, the tracking performance for these methods relies on both classification algorithms (or pattern recognition) and filter structure [9]. There is a great number of research aimed to improve the object tracking task by developing enhanced classification/clustering steps (before tracking) using learning-based [12], [13] or geometric model estimation [1] algorithms, but all are still classified under the first category of DATMO methods. There are other studies in this category, such as [14], focused on reducing the computational complexity by applying the classification and subsequent tracking only on the moving point. In a *detection-based* approach the detection and tracking steps are independent and various sensor data could be used in the detection algorithm without changing the tracking part.

B. Direct Tracking

In the direct tracking methods, the sensor model and/or object's geometric model is used to estimate corresponding points in space without prior detection. This method could be further divided into model-based and model-free (point-based) approaches.

In *model-bases direct tracking* DATMO algorithms, prior knowledge about the geometric shape and dynamic model of the moving vehicles are used to track the states and the geometric shape of the objects [15], [16] without detecting the objects first [9], [18]. Tracking the geometry helps to predict the dynamic properties with higher precision and discard tracked objects with strange shapes or geometry changes. However, the tracking accuracy declines for moving objects with different shapes and geometry like cyclists or pedestrians.

The second subclass of the *direct tracking* approach (*point-based tracking*) is a geometric model free in which every point is tracked in consecutive LiDAR scans. However, these scanned point clouds could be used directly or represented in the form of 2D/3D grid space before being used in a *grid-based* tracking algorithms [19]. The key advantage of the *point-based* DATMO stems from the fact that there is no assumption about the geometric shape of the object, and the objects are classified/detected based on tracking corresponding scanned points on them. But tracking all points makes the computation process expensive and limits the method in terms of the maximum number of moving objects in a scene [20]. To overcome this challenge, before tracking scanned points they are divided into static and moving categories by generating a static obstacle map (SOM) [21] or filter objects of interested with the help of deep learning methods [27]..

Point cloud registration (PCR) algorithms are widely used in model-free DATMO methods. After clustering point clouds in consecutive scans, corresponding clusters are detected and PCR algorithms such as interactive closest point (ICP) [32] are applied to each set (two clusters from the same object in different time steps) to calculate precise relative motion [21]–[26]. Although, low standard deviation of error has been reported for tracking velocity (0.4 m/sec) and orientation (1.81 deg) for the moving objects [21], these methods suffer from a number of considerable drawbacks. First of all, the computational time is not deterministic and depends on the number of moving objects. Secondly, The performance of the ICP algorithm highly depends on the initial conditions and the performance deteriorates when the relative velocity of the moving objects (with respect to EV) increases. Finally, because the PCR algorithms are based on the iterative optimization process, parallelizing these algorithms is not simple and straightforward. Various methods reviewed in this section are summarized along with advantages/disadvantages in Table.I.

III. System Model and Problem Definition

We consider a system consisting of an EV equipped with LiDAR sensors and multiple target vehicles (TVs) such as cars, vans (or other large vehicles with different shapes), and bikers on a segment of a road as shown in Fig. 2. The estimation of speed and direction of motion for all TVs is desired.

As illustrated in Fig. 1, the input of the DATMO pipeline is a 3D point cloud (P^l) generated by raycast LiDAR, and the desired output is a set of 2D velocity vectors ($\{\hat{\mathbf{v}}_1^t, \hat{\mathbf{v}}_2^t, \ldots, \hat{\mathbf{v}}_o^t\}$) each belongs to a unique track (trace of velocity vectors in a certain period of time). It should be noted that "o" is the number of moving objects at a time "t" (stationary objects are not included). Unlike [21], in this study, we don't assume a small relative velocity for the moving targets. Moreover, the update rate of the LiDAR sensor is presumed 10 Hz, so, in order to avoid losing data, the proposed algorithm should be able to calculate the desired output (within a radius of 120 m) in less than 100 ms (regardless of the number of moving objects). However, we assumed that the ego vehicle and the surrounding objects move in the horizontal plane (xy in Fig.3). Therefore the velocity in the vertical direction (z) is ignored and not reported in the estimated output. The estimated velocities are in the local coordinate system attached to the EV.

IV. Proposed Method

The proposed DATMO is illustrated using a block diagram of processes as shown in Fig. 1. The algorithm between input and output signals is divided into four main processes summarized as follows:

(A) 3D LiDAR sensor data are converted to 2.5D grayscale grid map.
(B) A velocity vector field generated using the optical flow algorithm.
(C) The false positive estimation are eliminated by a filtering mask calculated based on the continuum property of the velocity vector field for rigid-body motion
(D) Finally fusing all measured information and dynamic model in an Extended Kalman Filter (EKF)

The core process is the optical flow calculation and the rest are either for preparing input data for this step or post-processing the generated vector flow for filtering and tracking the true positives. In the following subsections, each process in the pipeline is described further in detail.

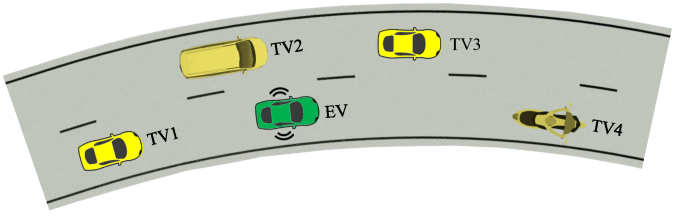

Fig. 2. Ego vehicle (EV) and target vehicles (TVs), including cars, vans, and bikers moving with different velocities

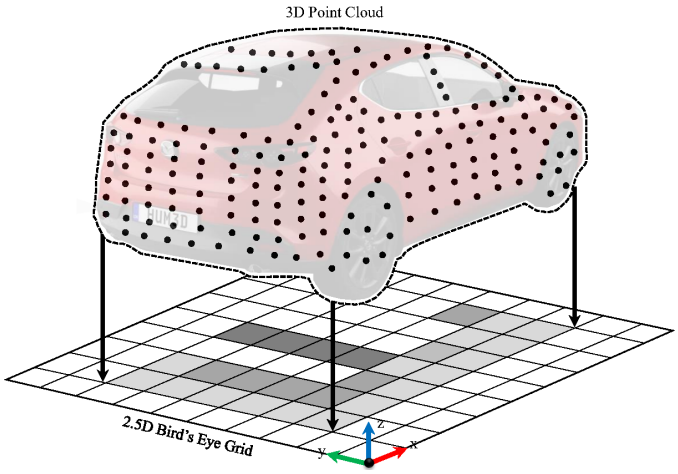

Fig. 3. 3D point cloud conversion to 2.5 bird's eye view grid. Darker grids corresponds to higher value of G_{ij}, and $G_{ij} = 0$ in white cells.

A. Point Cloud to Bird's Eye View Conversion Process

The optical-flow algorithm is the main component of the proposed method, and this process requires 2D grayscale images to calculate the velocity vector field. However, the input data from LiDAR is 3D scattered point cloud. Similar to [19] a conversion block is utilized for mapping the point cloud input to a bird's eye view grid which is also known as 2.5D grid map (Fig. 3). The input signal which is fed into the conversion process is a point cloud containing L points each has three coordinates without intensity data, e.g. the l^{th} point is represented by $P^l = \{p_x^l, p_y^l, p_z^l\}$. The output of the conversion process (input of the vector flow generator) is a grayscale 2D image which is called 2.5D grid map in this paper. In other words, the output is a $n \times m$ matrix in which cells are normalized between 0 and 255. Each cell is referred to by an ij pair where i (j) is an integer between 1 and N (M). Moreover, the centre of each cell in the grid has also a coordinate (G_{ij}^x, G_{ij}^y). Based on the grid's resolution, all cells have the same dimensions of w and h in x and y directions, respectively. A value is assigned to each cell of this grid space based on the height of the corresponding points in the point cloud data i.e. the points with the same x and y values (see Eq. 2). This value is calculated based on a linear combination of the mean and standard deviation of the height of the corresponding points projected on the horizontal plane. This concept has been illustrated in Fig. 3 and the value assigned to cell ij is obtained as follows:

$$G_{ij} = \frac{1}{h_{\max}} \left[a \cdot \mu \left(P_z^g \right) + b \cdot \sigma \left(P_z^g \right) \right] \tag{1}$$

where, a and b are constant weight, and $h_{\max}$ is normalizing constant. $\mu(\cdot)$ and $\sigma(\cdot)$ are mean and standard deviation functions, respectively. Superscription g is a set of integers $1, 2, .., L$ that satisfies the following condition:

$$\begin{cases} \left(G_{ij}^x - w/2\right) \leq p_x^g < \left(G_{ij}^x + w/2\right) \\ \left(G_{ij}^y - h/2\right) \leq p_y^g < \left(G_{ij}^y + h/2\right) \end{cases} \tag{2}$$

Based on this definition, $G_{ij} = 0$ means that there is no point in the point cloud corresponding to the cell ij above

ground. Lower values (relative to a threshold) show that the points are from ground [19]. On the other hand, higher values are assigned for the points on vertical planes (bigger height standard deviation) or horizontal plane with high z components (bigger average height).

B. Motion Vector Flow Generation using Farneback Optical Flow Algorithm

A dense optical flow algorithm is used to calculate the velocity (linear and angular) for each cell of the grid map generated in the previous step using two consecutive frames of the converted 2.5D grayscale grid map (Fig.4). The output signal of this process carries three $n \times m$ matrices including linear and angular velocities of each cell in the grid. Two matrices for linear velocities in x and y directions ($\mathbf{v_x}$ and $\mathbf{v_y}$, respectively), and one for angular yaw rate in the z-direction (ω).

There are several optical flow algorithms available in the literature for estimating the per-pixel motion between two consecutive images [33]. In this study, we need dense vector flow (not sparse) to calculate the velocity of all occupied cells with high accuracy and low computational cost. Although any dense optical flow algorithm could be used in the proposed DATMO framework, the well-known Gunar-Farneback optical flow generator [29] has been used here. This algorithm satisfies the requirements (accuracy-cost trade-off) more efficiently compared to other methods [34]. However, the Farneback algorithm employs expanded polynomial transformation of adjacent cell's brightness to estimate the dense velocity distribution for each grid cell [29], and this may cause estimating non-zero velocities for unoccupied cells in the neighbourhood of the occupied cells. This challenge is addressed in the next part (filtering and masking process).

The optical flow algorithm calculate linear velocity distribution, however, the angular velocity is also required for accurate state tracking of the vehicles (sec.IV-D). Based on vector field

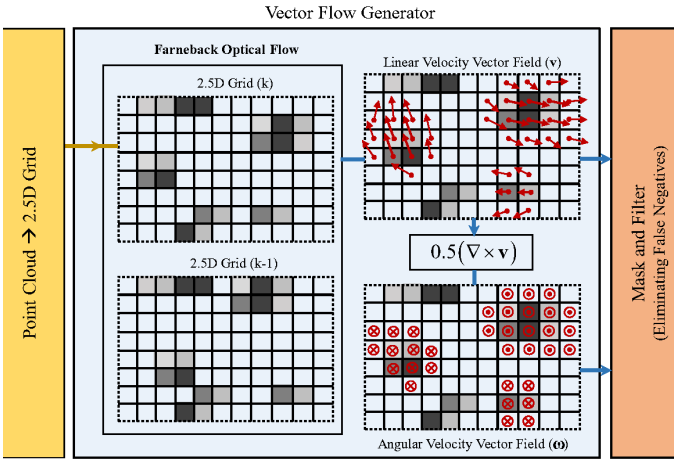

Fig. 4. Optical flow based velocity vector field generation process. Grayscale brightness refers to the occupied cells which contain LiDAR scanned points. $\otimes$ and $\odot$ show the angular velocity vectors perpendicular to the motion plane in $-z$ and $+z$, respectively. NOTE: for reading the system diagrams used in this paper the signals are expanded (rectangles with dashed line frame) to illustrate data that is carried between processing blocks.

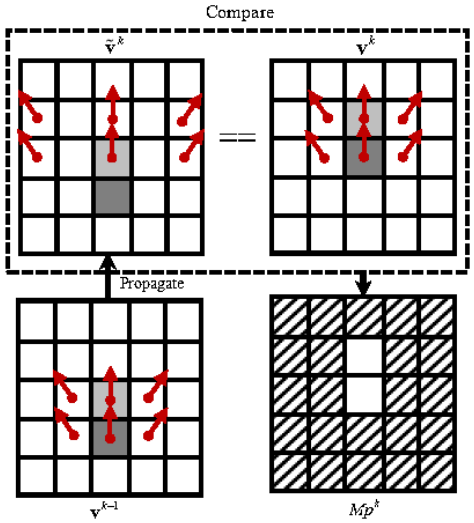

Fig. 5. Vector field propagation mask in time step k

theory [35] and rigid body assumption for each moving object in the scene, the angular velocity of each cell (in z direction) is obtained by the Eq. 3. (see [36] and Appendix).

$$\omega = 0.5\,(\nabla \times \mathbf{v}) \tag{3}$$

Where $\mathbf{v}$ is the linear velocity vector field, and ∇ is the curl operator.

Therefore, the output of the vector flow generation process includes angular velocity in z direction, in addition to 2D linear velocity in x and y directions.

C. Masking and Filtering the vector field

Due to the dense nature of the vector field obtained by the Farneback optical flow algorithm, the generated vector field contains false positive values for cells which are not occupied or do not belong to moving objects (static). The masking process is to filter out undesirable false positives and provides the final estimated velocity vectors, so the output of this step is a subset of its input. In this section, the mask is obtained in two steps and prepare the final vector filed for the tracking process.

1) Vector Field Propagation Mask: The second masking layer for the vector field is based on temporal filtering which is called *propagation* in this study. Propagation of the vector filed in time step k is obtained by changing the (x, y) position of the velocity vectors in the 2D plane according to the linear velocity values in time step $(k-1)$. The propagation is calculated using Eq. 4.

$$\begin{aligned} \tilde{\mathbf{v}}^{k}_{i'j'} &= \hat{\mathbf{v}}^{k-1}_{ij}\ ; \\ \begin{cases} i' &= i + \left\lfloor \hat{v}^{k-1}_{x} dt + w/2 \right\rfloor \\ j' &= j + \left\lfloor \hat{v}^{k-1}_{y} dt + h/2 \right\rfloor \end{cases} \end{aligned} \tag{4}$$

where $\tilde{\mathbf{v}}$ is propagated vector field, and dt is time increment. The value inside $\lfloor \cdot \rfloor$ is rounded down to the nearest integer.

As shown in Fig. 5, the propagated vector field of time step $(k-1)$ is compared with the vector field calculated in

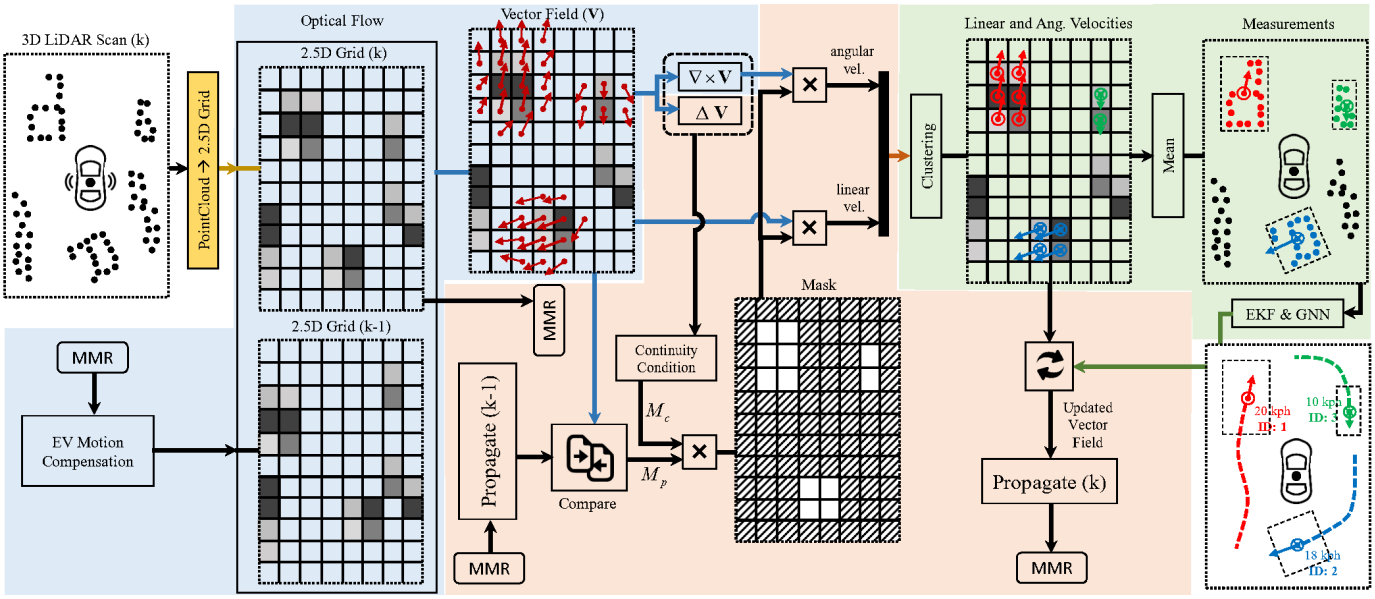

Fig. 6. Expanded schematic system diagram of optical flow based DATMO for AVs. NOTE: for reading the system diagrams used in this paper the signals are expanded (rectangles with dashed line frame) to illustrate data that is carried between processing blocks. And, **MMR** stands for memory.

the same time step and the masking matrix (Mc) is obtained using Eq. 5:

$$(Mp)_{ij}^{k} = \begin{cases} 1 & if \quad \left\|\tilde{\mathbf{v}}_{ij}^{k} - \hat{\mathbf{v}}_{ij}^{k}\right\| \leq \alpha_p \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In this equation, α_p is a constant threshold close to zero. The final masking boolean matrix is calculated by multiplying two masks calculated in two layers: $Mask = Mc \times Mp$. Applying this filtering mask to the vector field at each time step filters out undesirable false positive vectors.

2) Rigid-Body Continuity Mask: In this study, it has been assumed that the moving objects are rigid i.e different parts of a single object have zero relative motion. Therefore, linear ($\mathbf{v}$) and angular (ω) velocity vector fields should satisfy the continuity conditions of Eq. 6 [35]:

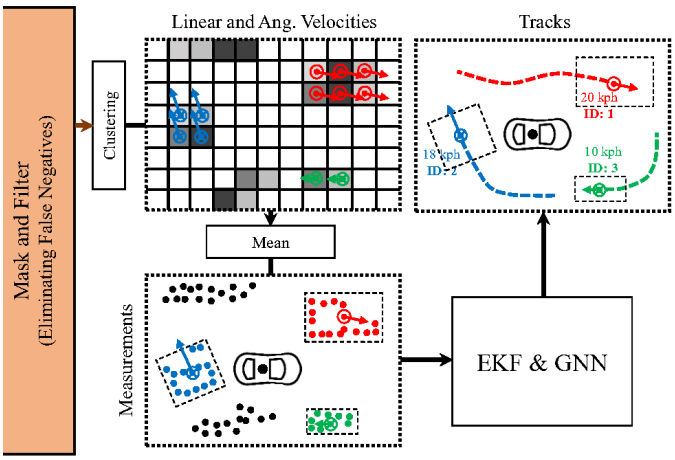

Fig. 7. Tracking process system diagram.

$$\begin{cases} \Delta \mathbf{V} & = & 0 \\ \nabla \left(\nabla \times \mathbf{V}\right) & = & 0 \end{cases} \tag{6}$$

The first part of Eq. 6 is for continuity in the linear velocity i.e the objects cannot tear apart nor implode, while the second part refers to the fact that all points on a single object should rotate with the same angular velocity. The results of both operations are 2D matrices, so, the estimated values for the cells that do not satisfy the condition (not exactly equal to zero but below a certain threshold α_{cont}) are set to zero. The resulting mask from this procedure is referred to by (Mc) in the rest of the text.

D. Tracking

The resulting vector field is used to detect moving objects and estimate their velocity. The tracking process output is the final estimated state of the objects (linear and angular velocities) augmented with a unique ID. As illustrated in Fig.1, $\bar{x}$, $\hat{x}$. and $\tilde{x}$ are masked, estimated, and propagated values of x variable, respectively, while the superscription shows the time step for the variables. An Extended Kalman Filter (EKF) is designed to use vector field data as measurements and the dynamic model of Eq.7 (constant linear/angular acceleration) as the prediction model to estimate the state (X_n) of the moving objects. Every estimated position and velocity is assigned to either an existing or new track with a unique ID via Global Nearest Neighbour (GNN) [21].

$$\dot{X}_n = f\left(X_n, U\right)$$

$$\begin{bmatrix} \dot{x}_n \\ \dot{y}_n \\ \dot{\theta}_n \\ \dot{v}_n \\ \dot{\omega}_n \end{bmatrix} = \begin{bmatrix} v_n \cos\theta_n - v + l_n\omega \\ v_n \sin\theta_n - l_n\omega \\ \omega_n - \omega \\ k_a \\ k_\alpha \end{bmatrix}; \quad U = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{7}$$

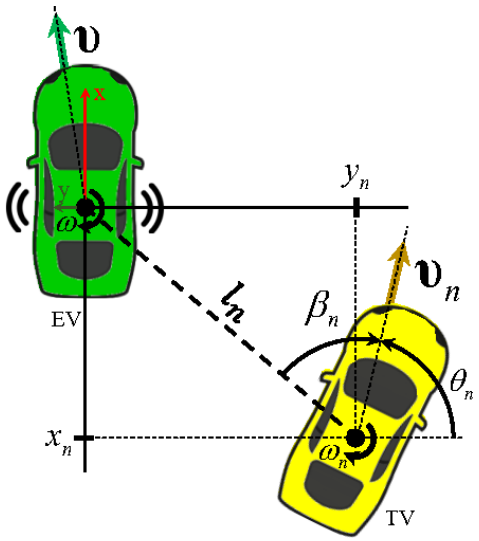

Fig. 8. Target vehicle's (TV) configuration (position, orientation, and velocity) relative to the ego vehicle (EV)

1) Measurements in EKF and Updating Tracks: In the Kalman filter structure three measurements are used for each moving object i.e two linear velocities and one angular velocity in z direction. As illustrated in Fig.7, these measurements are calculated by clustering masked velocity vector fields $\{\bar{\mathbf{v}}_\mathbf{x}, \bar{\mathbf{v}}_\mathbf{y}, \bar{\omega}\}$ provided by optical flow and taking the mean value of each cluster. In the proposed approach, Euclidean distance is utilized for clustering vectors, and mean position and velocity are fed into the EKF algorithm to estimate the state vector for each moving object using motion dynamics of Eq.7.

All clustered points should be either assigned to an existing track or initialised on a new track. Similar to [21], in our approach, the clusters are assigned to the predicted tracks via GNN. Each cluster is assigned to at most one track based on a 4D feature vector $[x_m, y_m, \lambda_1, \lambda_2]$ containing the mean position and shape of the cluster (independent of the orientation) in the motion plane. Two components in the features vector showing the shape of a cluster are eigenvalues of the covariance matrix of the points in a cluster (λ_1, λ_2). So, a cluster is assigned to a track if the Euclidean distance between their feature vectors is less than a threshold γ.

The final step in managing the tracks is confirming and/or deleting tracks. Every one of these two procedures is done by a 2D integer vector. A track is confirmed when M_1 number of measurements/detection is assigned to it in the last N_1 updates ($M_1 < N_1$). And similarly, a confirmed track is deleted if in the last N_2 ($M_2 < N_2$) consecutive updates, no measurement is assigned to it M_2 times. It should be noted that the coordinate system used in this section is attached to EV with a configuration shown in Fig.8.

The interaction between different processes is depicted in the assembled system diagram of Fig.6. This system diagram is a detailed version of Fig.1.

V. Performance Evaluation

An experimental test is designed to evaluate and verify the performance of the designed DATMO algorithm. Two main objectives are targeted in this section. Comparing the proposed DATMO with state-of-the-art (SOTA) methods, and obtaining an error model for estimation accuracy.

First and foremost we statistically compare the performance of the DATMO method with SOTA geometric model-free approach (GMFA) developed in [21] which has been proven to be more efficient than the geometric model-based tracking (MBT) method proposed in [37]. However, the proposed method is further compared against SOTA model free [14] and model-based [18] direct tracking methods as well. The GMFA algorithm is regenerated and evaluated, while the quantitative performance of the other methods are obtained from [14] and [18]. Therefore, the later experiments are consistent with those specified in these studies.

In addition, we further investigate how the state estimation error changes as a function of the deriving environment i.e configuration of EV and TV. Regarding these objectives, the experimental evaluation is conducted in two different steps. Initially, synthetic data is generated to evaluate the algorithm in various custom situations, and in the next step, the algorithm is tested on a real-world dataset of KITTI.

A. Datasets

1) Synthetic Data Generation and Simulation: In order to evaluate the proposed method for estimating the state of the target vehicles in diverse possible configurations, generating a synthetic dataset is essential. In addition, the estimation error is calculated more accurately in the simulation compared with real-world datasets such as KITTI for which the ground truth of objects' velocity has not been provided directly. In this study, the TV's configuration space is defined by three variables (Fig. 8): distance to EV (l_n), relative orientation (β_n), and relative velocity ($\Delta\upsilon_n = \upsilon_n - \upsilon$). The aim is to design scenarios covering all possible configurations for investigating the meaningful relations between the estimation error and these three variables, in addition to assessing the estimation accuracy. Therefore, the flexibility in changing different configurations provided by synthetic datasets is another reason that justifies utilising this type of dataset.

The driving scenario designer toolbox in MATLAB is used to generate synthetic scenarios and add a LiDAR sensor to collect point cloud data. As illustrated in Fig. 9-(b), three different types of TVs are simulated in synthetic scenarios including sedan, van, and cyclist. Moreover, for considering EV motion effect completely, a nonzero curvature is considered to avoid zero yaw rate for EV. The LiDAR sensor parameters are adjusted according to what is used in the KITTI dataset collection sensor. The point cloud data from a simulated scene has been plotted in Fig. 9-(c).

In order to cover all possible cases for the n-th target vehicle configurations ($\{l_n, \beta_n, \Delta\upsilon_n\}$), each scenario contains a target vehicle (sedan, van, or cyclist) moving in the same multi-lane road in which EV moves in one of the lanes (with a speed of 20 m/s). TVs move with 10 different speeds (10 to 40 with a

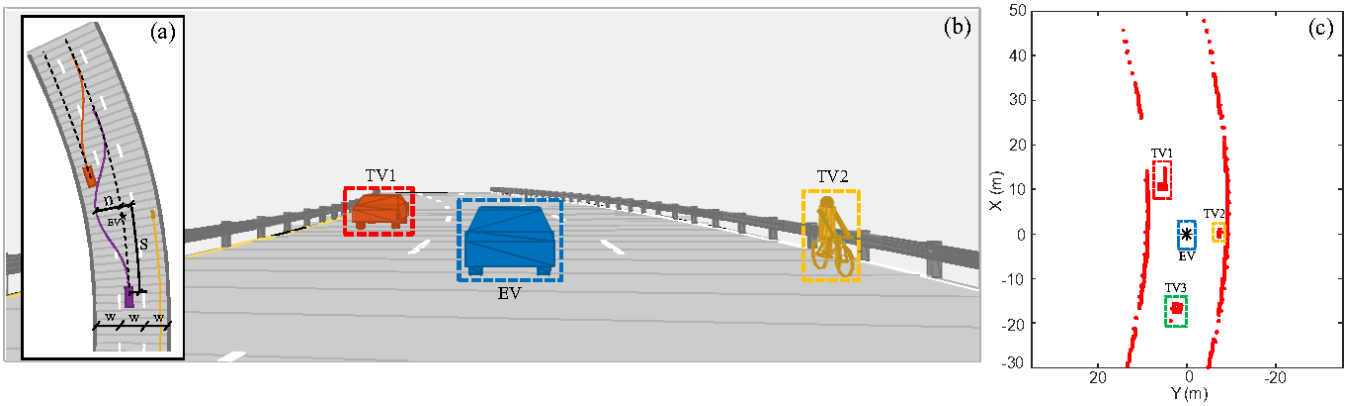

Fig. 9. Synthetic primary scenario generated in Matlab scenario designer: trajectories design parameters for different TVs (a), a 3D meshed object in scene (b), and bird's eye view scanned point cloud excluding ground points in EV coordinate system (c)

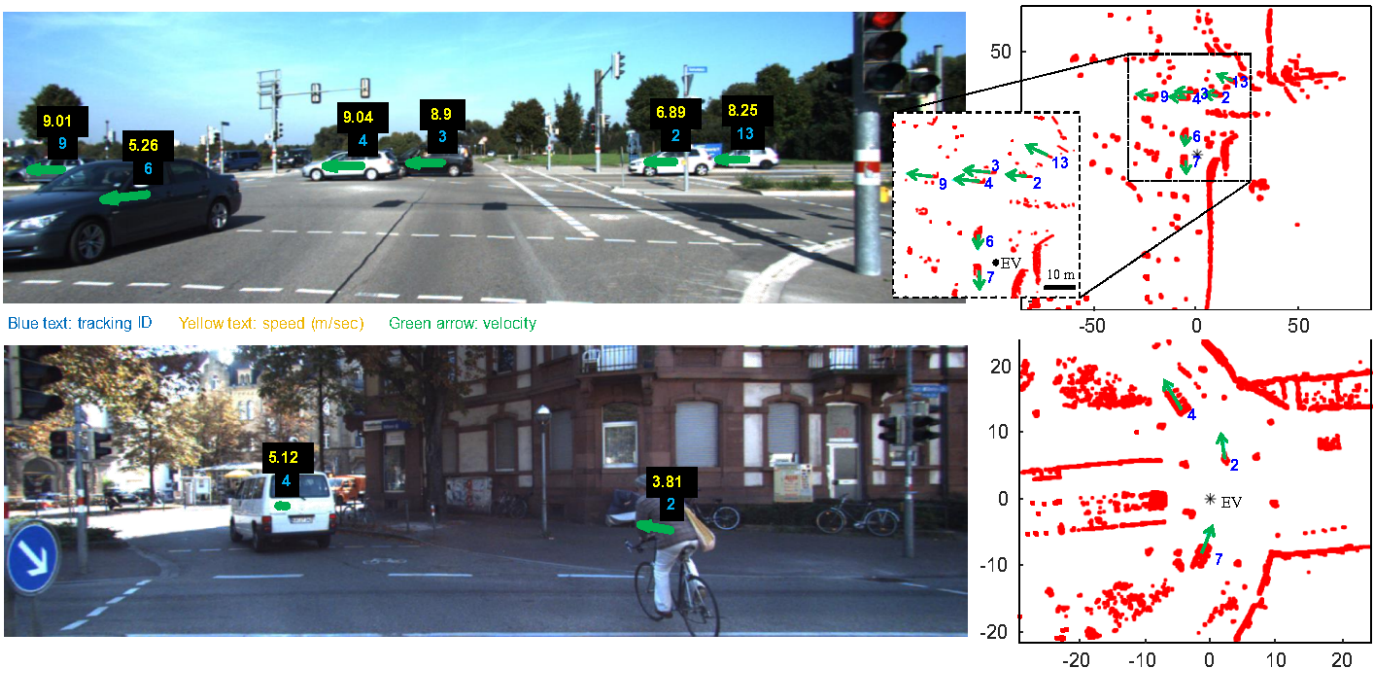

Fig. 10. Proposed DATMO algorithm result for KITTI dataset: tracking IDs and velocity vectors plotted on the front camera image (left), and bird's eye view scanned point cloud excluding ground points (right)

step of 2 m/s) in a lane and drive in two modes: first, keep the same lane, and second, overtake back and forth between two lanes with trajectories defined by two parameters of s and $n = w/2$ shown in Fig.9-(a). In the case of changing lanes/overtaking, two values of 2 and 4 seconds are used for s (assuming constant speed). And finally, the lateral offset of the TV start lane from EV's lane varies from -80m to 80m (with a step of 1m), The cyclists' trajectories include only lane-keeping i.e without any lateral motion. There is only one TV in each generated scenario to prevent occlusion, although, in Fig.9 three TVs are depicted which is a combination of three scenarios to be more informative.

We refer to all synthetic scenarios described above as *primary scenarios*. To further compare the performance metrics against model-free [14] and model-based [18] indirect tracking methods, simulation scenarios designed in [14] are replicated (*secondary scenarios*). In these scenarios the TV moves with the speed of 6 m/sec along i) straight right-angled, ii) right turn, and iii) circular paths (see [14] for details).

2) KITTI Dataset: The final evaluation is conducted using the real-world KITTI tracking dataset for multi-object tracking task. Besides the ground truth labels, only LiDAR data from this dataset is used in the current study for the estimation task, however, the colour images of each frame are also used to plot velocity vectors in image coordinate (Fig.10-left) using transformation matrix (velo-to-cam). Moreover, since there is no ground truth label for the velocity of objects in the driving environment, it is obtained by tracking the centre of the 3D bounding boxes. In the KITTI dataset, the bounding box coordinates are provided in the camera frame whereas the estimated velocity values are obtained in the LiDAR coordinate system. Therefore, the calculated velocities

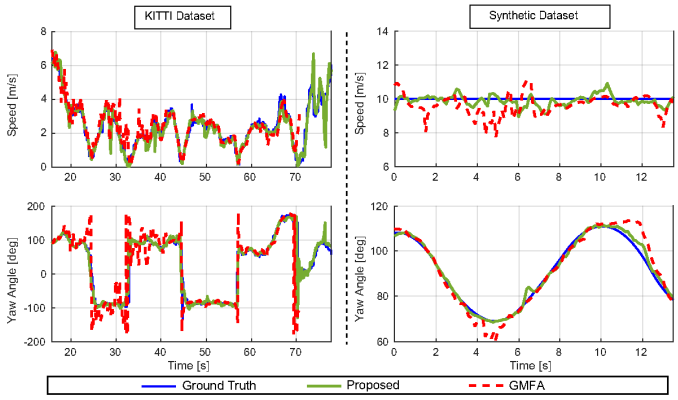

Fig. 11. Comparative results of the proposed and GMFA algorithms for the synthetic and KITTI datasets

are transformed to the camera coordinate ($T_{velo-to-cam}$) to calculate estimation error.

Since we need to compare the estimation results with the ground truth labels, the training sequences are used for velocity and yaw angle error calculations, however, both training and testing sequences are used to obtain detection performance. Adopted from [21], the point cloud data closer than 25m in the lateral direction (left and right), 80m and 15m in longitudinal front and rear directions, respectively are considered, and the rest of the data and labels out of this range are discarded in the evaluation process.

B. Evaluation Metrics

Following the previous studies [21], [22], the velocity vector estimation accuracy is evaluated by speed and angle (θ) errors with respect to the ground truth (GT) values. The estimation errors for o^{th} target object at time t are calculated in Eq. 8.

$$\begin{aligned} \delta v_o(t) &= \left| \left\|\mathbf{v}_o^{GT}\right\| - \left\|\hat{\mathbf{v}}_o\right\| \right|_t \\ \delta\theta_o(t) &= \left| \theta_o^{GT} - \hat{\theta}_o \right|_t \end{aligned} \tag{8}$$

In order to compare DATMO performance throughout all data points, the standard deviation (σ) of the error distribution of all timesteps and target moving objects is used in Table.II. Moreover, same as [21], the detection performance is also evaluated by *Precision* and *Recall* defined in Eq. 9.

$$\begin{aligned} Pr &= (TP)/(TP+FP) \\ Re &= (TP)/(TP+FN) \end{aligned} \tag{9}$$

The last metric to quantify the estimation performance is the time each algorithm takes to process an instance of the LiDAR scan to detect and estimate the state of the moving objects.

C. Results

The evaluation results are divided into three sections. Firstly, a stochastic comparative analysis is conducted with model-free direct tracking methods in [21] (GMFA) and [27]. Secondly, simulation results compare the proposed method with model-free and model-based indirect tracking methods developed in [14] and [18]. Lastly, the effects of the continuity filters (Section. IV-C) are presented in an ablation study.

The grid size in the proposed algorithms is 0.17×0.17 m and the Farneback optical flow algorithm used in the proposed method is taken from OpenCV computer vision library with the following setting: *NumPyramidLevels = 3*, *PyramidScale = 0.5*, *NumIterations = 3*, *NeighborhoodSize = 3*, and *FilterSize = 11*.

1) Comparison with Direct Tracking Methods: The detection and state estimation results for 21 sequences of the KITTI training labelled dataset and more than 800 synthetic driving scenarios (primary) are presented in this section. In both sets of these datasets, the detection and state estimation performance evaluated for cyclists, sedans, vans (or bigger vehicles such as trucks or buses in the KITTI dataset) and pedestrians are ignored in this study. As a sample, the tracking results of moving object tracking in both KITTI and synthetic datasets have been illustrated in Fig.11 left and right column, respectively, for GMFA and the proposed approaches (dashed red and solid green, respectively). Discontinuation of the dashed red diagram in the left column plot shows that the GMFA couldn't track the object from approximately 71 sec onward. The top row in this figure shows the speed estimations whereas the bottom row depicts the yaw angle estimation results for a moving object in one sequence. The estimated values are reported as relative values i.e measured in the EV coordinate system.

In order to compare the GMFA and the proposed approaches, the estimation error distribution of all sequences for two datasets is obtained. This distribution contains estimation error of all time steps throughout all sequences. Speed and yaw angle estimation error distribution has been shown in Fig.12 top and bottom row, respectively. In this figure, the performance of both GMFA (red) and the proposed (green)

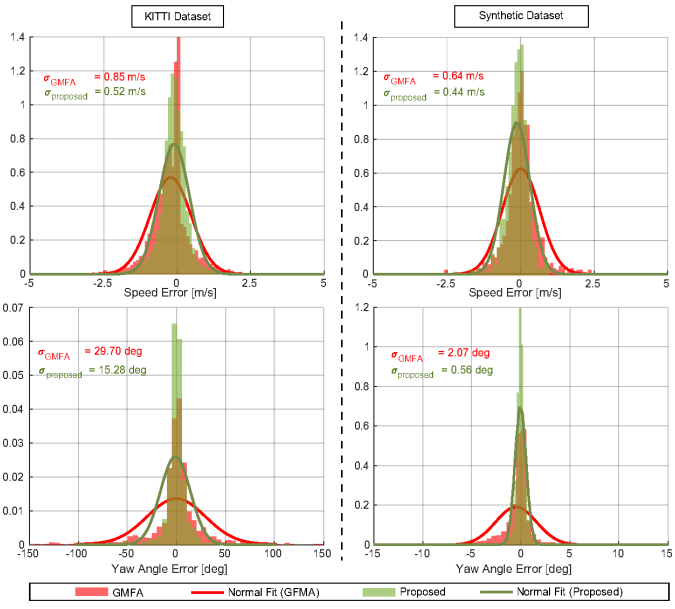

Fig. 12. Comparative error distribution of the proposed and GMFA [21] algorithms for the primary synthetic and KITTI datasets

TABLE II
EXPERIMENTAL EVALUATION RESULTS OF THE PROPOSED METHOD AND GMFA [21] FOR BOTH SYNTHETIC SIMULATION AND KITTI DATASET

Dataset	$\|\Delta v\|$ [m/s]	Method	Precision (%)	Recall (%)	σ_v [m/s]	σ_a [deg]	Time [ms]
Simulation	≤ 1	GMFA	**94.3**	**93.6**	0.28	**1.12**	147
		Proposed	94.1	93.8	**0.23**	1.37	**139**
	> 1	GMFA	88.9	87.3	2.28	3.81	155
		Proposed	**89.2**	**88.3**	**1.48**	**0.63**	**147**
KITTI	≤ 1	GMFA	**89.7**	88.7	0.48	18.12	196
		Proposed	88.8	**89.1**	**0.33**	**10.97**	**171**
	> 1	GMFA	87.5	86.8	0.78	34.21	201
		Proposed	**88.6**	**88.2**	**0.56**	**16.71**	**183**

TABLE III
COMPARING TO OTHER MODEL-FREE AND LEARNING-BASED MOTION ESTIMATION METHODS

Ref	Speed Error [m/sec]	time [ms]	Training needed
Wang et al. [38]	1.69	80	Yes
Liu et al. [39]	4.37	-	Yes
Li et al. [27]	0.42	240	Partially
Proposed	0.44	142	No

TABLE IV
COMPARING THE MEAN AND MAX ESTIMATION ERRORS AGAINST MODEL FREE [14] AND MODEL-BASED [18] INDIRECT TRACKING METHODS. RESULTS OBTAINED FROM THE SECONDARY SCENARIOS IN [14].

secondary scenarios	Method	Speed [m/sec]		Direction [deg]	
		mean	max	mean	max
i	Wang et al. [14]	0.25	0.43	0.52	1.49
	Zhang et al. [18]	0.38	0.59	1.23	1.93
	Proposed	0.09	0.31	0.18	0.51
ii	Wang et al. [14]	0.40	0.70	0.80	2.53
	Zhang et al. [18]	0.52	1.00	1.53	4.02
	Proposed	0.21	0.47	0.83	1.70
iii	Wang et al. [14]	0.29	0.40	1.65	2.50
	Zhang et al. [18]	0.43	0.84	2.25	5.09
	Proposed	0.19	0.44	0.41	1.82

methods is illustrated for KITTI and synthetic datasets separately in the left and right columns, respectively. Furthermore, for each distribution, a normal distribution function has been fitted with the standard deviation value printed in the top left for both methods using the same colour codes. Similar to [21] the standard deviation values are used to compare the accuracy of DATMO methods.

Finally, the detection and estimation results of two methods and two datasets are summarized in Table.II. Precision and recall metrics are for evaluating moving object detection while the standard deviation and time columns report the result of the state estimation accuracy and computational cost, respectively. The results are further reported for two different ranges of relative velocity ($\|\Delta v\| \leq 1$ and $\|\Delta v\| > 1$ m/s), because the GMFA method of [21] is developed for detecting and tracking of moving objects with *"low relative speed"*. Therefore, in order to check if the GMFA method is replicated properly, the estimation errors for low relative speeds ($\|\Delta v\| \leq 1$ m/s) should be less than what was reported in [21]. It should be noted that since there is no exact velocity ground truth label for the KITTI tracking dataset, the calculated error for this dataset even for low speeds is not comparable directly with values reported in [21], and we use the replicated GMFA algorithm instead to only compare the final estimation error with the proposed approach performance. Moreover, the processing time reported in this table is the average time the computing unit (Intel Core(TM) i7-7600 CPU @ 2.80GHz) needs for each cycle excluding the first step of each sequence which needs extra initialization time. The breakdown of computational complexity for different processes within the framework is given in Table. VI. Since we believe that the core process of optical flow in the proposed method could be parallelized using off-the-shelf tools, this process was implemented on both CPU and GPU (GeForce RTX 2080 Ti) for the simulation scenarios. The results indicate an 80% improvement in processing time for GPU compared to CPU.

The estimation error and computational complexity comparison with other model-free and learning-based motion estimation methods is summarised in Table. III. The performance metrics of other methods and evaluation conditions are adopted from [27]. The results are based on the KITTI tracking dataset, sequences 0000, 0005, and 0010. The objects within a radius of 50 m are considered. The results, represented in Table. III, include other learning-based motion estimation methods as additional references. The comparison suggests that although our method's performance in terms of speed accuracy is comparable to that of [27], there is a significant improvement in computational complexity. This is attributed to the iterative nature of the ICP method used in [27]. Moreover, all other methods in this table are data-driven and will need retraining the different situation or sensor configurations; otherwise, the performance may decline. While our method is deterministic and does not need training.

2) Comparison with Indirect Tracking Methods: Table IV presents secondary simulation results for comparison with in-

direct tracking methods. Like the simulation scenarios used in this comparison, the performance metrics and values for other methods in this table are sourced from [14]. As anticipated, the proposed method outperforms both indirect tracking methods, as they compute velocity based on macroscopic classified point clouds. In contrast, our framework employs the optical flow algorithm to calculate the velocity field at a microscopic level (grid-based) before the classification and EKF tracking processes take place.

3) Ablation Study: The impact of the two-layer filters applied to the velocity vector field and ablation experiment is investigated in this section. The *continuity* and *propagation* filters (see Section. IV-C) are disabled individually to quantify their effect through the changes in the performance metrics. Only the KITTI dataset is used for this experiment and the results are summarized in Table. V

Based on the findings presented in Table. V, the performance metrics show improvement with the inclusion of both filters, with notably significant enhancement attributed to the propagation filter.

D. Estimation Error Sensitivity to TV's Configuration

After validating the proposed DATMO approach and comparing it with the state-of-the-art method, the sensitivity of the speed estimation error to the changes in the TV's configuration for the proposed method is explored in this section. This would help other researchers who use this tracking method (motion planning and control) to consider an error model. Two elements of the TV configuration (β_n, l_n) are used as variables in this section. In other words, we want to investigate how the estimation error changes by changing the distance (l_n) and orientation (β_n) of the TV. The synthetic data is used to sweep these variables and the proposed algorithm is applied to measure the estimation error for each case. The result of this sensitivity analysis is presented in a 3D plot and three 2D plots (three views of the same 3D plot) in Fig.13. In this figure, the heatmap colour correlated to the absolute speed estimation error (e_v) is used to visualize the error value, particularly in the top-view plot (red and blue colours corresponding to high and low absolute error, respectively).

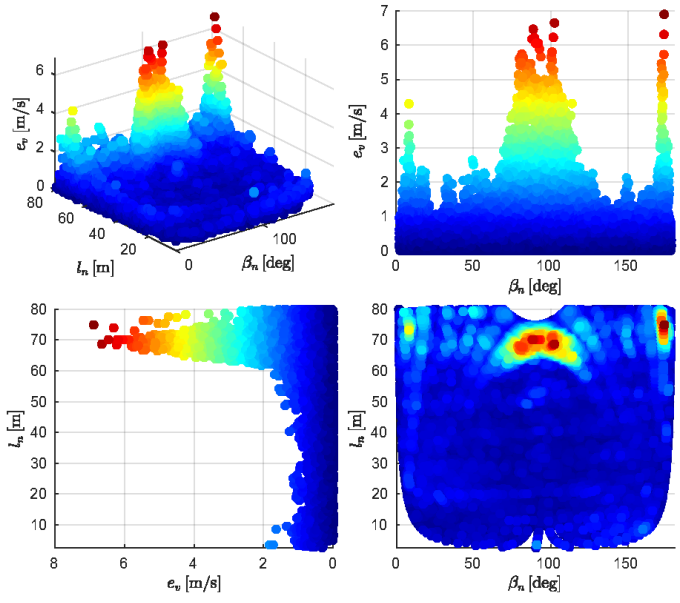

Fig. 13. Error model as a function of orientation and distance of the TV with respect to EV for $\Delta v = 10$ m/s. The heatmap colour corresponds to the absolute error value in e_v axis

TABLE V
ABLATION STUDY FOR THE TWO-LAYER FILTER APPLIED ON VELOCITY VECTOR FIELD

#	Propagation Filter	Continuity Filter	σ_a [m/sec]	σ_a [deg]
1	✗	✗	2.11	0.93
2	✓	✗	1.45	0.70
3	✓	✓	1.38	0.64

TABLE VI
PROCESSING TIME FOR DIFFERENT COMPONENTS OF THE PROPOSED METHOD

Process:	Data Pars-ing	3D to 2D Conver-sion	Optical Flow (GPU/CPU)	GNN Tracker	Total (GPU/CPU)
Time [ms]:	8.1	4.4	4.6/119	10.8	27.9/142.3

VI. DISCUSSION

The obtained results are presented in the section.V-C are further discussed in detail in this section. The comparative data reported in TableII and Fig.12 show the superior performance of the proposed method compared with the GMFA approach. But before comparing the two approaches, we need to validate the regenerated algorithm for GMFA. Since this algorithm is originally developed for low relative speed and has been validated with an autonomous vehicle platform, the performance of the regenerated algorithm for low-speed synthetic dataset should surpass the values reported in [21] (the standard deviation of the speed and yaw angle error are 0.40 m/s and 1.81 deg, respectively). According to Table.II (first row), the GMFA result for low-speed simulation outperforms these outcomes. Therefore, the regenerated GMFA algorithm is reliable to be tested as the baseline with other datasets such as KITTI or high relative speed synthetic datasets.

In detecting the moving objects, precision and recall values show almost similar performance for both comparing methods (increased only 1% in the proposed approach). However, the state estimation accuracy shows more than 34% and 50% improvement in the standard deviation of the estimated speed and yaw angle error distribution, respectively. The fitted normal distribution along with the standard deviations for both synthetic and KITTI datasets has been shown in Fig.12.

Moreover, the measured processing times show an average of $\sim 8\%$ improvement for the proposed method compared with the GMFA. The computation time for the synthetic data shows less improvement compared to the KITTI dataset (5% vs 10.5%). Since each sequence of the synthetic scenario contains

only one moving object, and the GMFA is based on point cloud registration for which the processing time depends on the number of detected moving point clusters, the computational effort is more consistent and less than that of KITTI scenarios in which the number of moving objects are more than one in most sequences. All computations in this study were done on CPU without parallel processing, whereas the proposed method which is based on an optical flow algorithm, has the potential to be implemented on GPU to accelerate the computations. This is another advantage of this approach over point cloud registration-based methods such as GMFA that use optimization and this makes it more difficult to parallelize the computations. Recently, Contemporary GPUs dedicate hardware to particularly accelerate optical flow algorithms up to 10 times faster [40]. Therefore, using parallel computation will accelerate the processing even more for the proposed approach.

Overall, the comparison results indicate that the proposed method's performance in state estimation and computational cost is comparable with the state-of-the-art method (GMFA), and as the last part of analysing the results error sensitivity of the proposed method is considered. The estimation error sensitivity to the configuration of the TV, illustrated in Fig.13, shows that the error magnitude is more sensitive to the orientation of the TV when the target vehicle is located at farther distances ($l_n > 45$ m). The way that the error value changes with respect to the orientation of the TV (β_n) is also interesting. The error increases at three specific orientations: $\beta_n = 0, 90, 180$ deg. regarding Fig.8, the first ($\beta_n = 0$ deg) and last ($\beta_n = 180$ deg) orientations correspond to the configuration in which TV facing or backing on to the EV, whereas the second orientation ($\beta_n = 90$ deg) is for the case in which TV's side is toward the EV i.e LiDAR sensor location. One of the possible reasons for this correlation could be the fact that in these configurations the scanned point cloud is no longer scattered in 3D space and mostly on a 2D plane. For instance, in $\beta_n = 90$ deg most of the scanned points are from the side of the vehicle. However, to elaborate more on the error model and consider all involved factors more research is required in future studies.

VII. Conclusion

In this study, a novel DATMO technique was proposed using a Farneback optical flow algorithm. This study revealed the promising potential of this approach in terms of accuracy and processing costs. Similar to traditional GMFA techniques, the optical-flow-based technique approach proposed and studied in this paper demonstrated good resilience against variations of the object sizes in driving scenes. Analysis of the error sensitivity to the configuration of the target vehicle in this study revealed meaningful correlations which could be used in future for error modelling. Our results showed that the error values increase when the TV moving in radial ($\beta_n = 0, 180$ deg) and tangential ($\beta_n = 90$ deg) directions in distances farther than 50 m. It shall be noted that Small size objects such as pedestrians were not covered in our study. Further studies could explore estimating the state of pedestrians by reducing the grid size and implementing the algorithm using parallel computing to calculate optical flow.

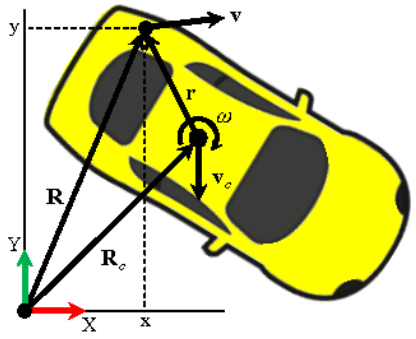

Fig. 14. Velocity of points on moving rigid body (vehicle)

Acknowledgment

This research is sponsored by the Centre for Doctoral Training to Advance the Deployment of Future Mobility Technologies (CDT FMT) at the University of Warwick.

Appendix

Deriving Angular Velocity from Velocity Vector Field

Assuming rigid body motion, the angular velocity could be obtained from the velocity vector field. If $\left\{\hat{i}, \hat{j}, \hat{k}\right\}$ are unit vectors in $\{x, y, z\}$, respectively, and considering notation used in Fig. 14, the angular velocity for planar motion is derived as follows:

$$\begin{aligned} \mathbf{v} &= \mathbf{v}_c + \omega \times \mathbf{r} \\ &= \mathbf{v}_c + \omega \times (\mathbf{R} - \mathbf{R}_c) \\ &= (\mathbf{v}_c - \omega \times \mathbf{R}_c) + \omega \times \mathbf{R} \end{aligned}$$

Rewriting this equation by substituting $\mathbf{R} = x\hat{\mathbf{i}} + y\hat{\mathbf{j}}$, and $\mathbf{V}_c = \mathbf{v}_c - \omega \times \mathbf{R}_c = V_{cx}\hat{\mathbf{i}} + V_{cy}\hat{\mathbf{j}}$:

$$\begin{aligned} \mathbf{v} &= \mathbf{V}_c + \left(-\omega y\hat{\mathbf{i}} + \omega x\hat{\mathbf{j}}\right) \\ &= (V_{cx} - \omega y)\,\hat{\mathbf{i}} + (V_{cy} + \omega x)\,\hat{\mathbf{j}} \end{aligned}$$

And by applying the curl operator to both sides, the angular velocity is obtained based on the curl of the vector field $\mathbf{v}$. It should be noted that the rigid body assumption makes the curl independent of the position and linear velocity of the centre c.

$$\begin{aligned} \nabla \times \mathbf{v} &= [\partial/(\partial x)\,(V_{cy} + \omega x) - \partial/(\partial y)\,(V_{cx} - \omega y)]\,\hat{\mathbf{k}} \\ &= 2\omega\hat{\mathbf{k}} \\ \Rightarrow \quad \omega &= 0.5\,(\nabla \times \mathbf{v}) \end{aligned}$$

References

[1] M. Sualeh and G.-W. Kim, "Dynamic multi-lidar based multiple object detection and tracking," *Sensors*, vol. 19, no. 6, p. 1474, 2019.

[2] M. Y. Abbass, K.-C. Kwon, N. Kim, S. A. Abdelwahab, F. E. A. El-Samie, and A. A. Khalaf, "A survey on online learning for visual tracking," *The Visual Computer*, vol. 37, no. 5, pp. 993–1014, 2021.

[3] C. Premachandra, S. Ueda, and Y. Suzuki, "Detection and tracking of moving objects at road intersections using a 360-degree camera for driver assistance and automated driving," *IEEE Access*, vol. 8, pp. 135 652–135 660, 2020.

[4] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE transactions on intelligent transportation systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[5] M. Kusenbach, M. Himmelsbach, and H.-J. Wuensche, "A new geometric 3d lidar feature for model creation and classification of moving objects," in *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016, pp. 272–278.

[6] A. Börcs, B. Nagy, and C. Benedek, "Instant object detection in lidar point clouds," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 7, pp. 992–996, 2017.

[7] S. Steyer, G. Tanzmeister, and D. Wollherr, "Grid-based environment estimation using evidential mapping and particle tracking," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 384–396, 2018.

[8] M. C. Hutchison, J. A. Pautler, and M. A. Smith, "Traffic light signal system using radar-based target detection and tracking," Oct. 26 2010, uS Patent 7,821,422.

[9] Y. Ye, L. Fu, and B. Li, "Object detection and tracking using multi-layer laser for autonomous urban driving," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 259–264.

[10] L. Spinello, R. Triebel, and R. Siegwart, "Multiclass multimodal detection and tracking in urban environments," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1498–1515, 2010.

[11] B. Douillard, D. Fox, F. Ramos *et al.*, "Laser and vision based outdoor object mapping." in *Robotics: Science and Systems*, vol. 8, 2008.

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

[13] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 529–10 538.

[14] H. Wang and B. Liu, "Detection and tracking dynamic vehicles for autonomous driving based on 2-d point scans," *IEEE Systems Journal*, 2022.

[15] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2, pp. 123–139, 2009.

[16] J. An and E. Kim, "Novel vehicle bounding box tracking using a low-end 3d laser scanner," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3403–3419, 2020.

[17] S. Steyer, C. Lenk, D. Kellner, G. Tanzmeister, and D. Wollherr, "Grid-based object tracking with nonlinear dynamic state and shape estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 7, pp. 2874–2893, 2019.

[18] 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.

[19] A. Asvadi, P. Peixoto, and U. Nunes, "Detection and tracking of moving objects using 2.5 d motion grids," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 788–793.

[20] R. Kaestner, J. Maye, Y. Pilat, and R. Siegwart, "Generative object detection and tracking in 3d range data," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3075–3081.

[21] H. Lee, J. Yoon, Y. Jeong, and K. Yi, "Moving object detection and tracking based on interaction of static obstacle map and geometric model-free approachfor urban autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3275–3284, 2020.

[22] H. Lee, H. Lee, D. Shin, and K. Yi, "Moving objects tracking based on geometric model-free approach with particle filter using automotive lidar," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[23] F. Moosmann and C. Stiller, "Joint self-localization and tracking of generic objects in 3d range data," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1146–1152.

[24] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Motion-based detection and tracking in 3d lidar scans," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 4508–4513.

[25] J. Groß, A. Ošep, and B. Leibe, "Alignnet-3d: Fast point cloud registration of partially observed objects," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 623–632.

[26] J. Kim, H. Lee, and K. Yi, "Online static probability map and odometry estimation using automotive lidar for urban autonomous driving," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 2674–2681.

[27] J. Li, X. Huang, and J. Zhan, "High-precision motion detection and tracking based on point cloud registration and radius search," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[28] E. Arnold, S. Mozaffari, and M. Dianati, "Fast and robust registration of partially overlapping point clouds," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1502–1509, 2021.

[29] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis*. Springer, 2003, pp. 363–370.

[30] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

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

[32] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[33] D. Fortun, P. Bouthemy, and C. Kervrann, "Optical flow modeling and computation: A survey," *Computer Vision and Image Understanding*, vol. 134, pp. 1–21, 2015.

[34] J. Tanaś and A. Kotyra, "Comparison of optical flow algorithms performance on flame image sequences," in *Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2017*, vol. 10445. SPIE, 2017, pp. 243–249.

[35] K. M. Urwin, *Advanced calculus and vector field theory*. Elsevier, 2014.

[36] J. Casey, "A treatment of rigid body dynamics," *Journal of Applied Mechanics*, vol. 50, pp. 905–907, 1983.

[37] H. Cho, Y.-W. Seo, B. V. Kumar, and 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)*. IEEE, 2014, pp. 1836–1843.

[38] Q. Wang, J. Chen, J. Deng, X. Zhang, and K. Zhang, "Simultaneous pose estimation and velocity estimation of an ego vehicle and moving obstacles using lidar information only," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12 121–12 132, 2021.

[39] X. Liu, C. R. Qi, and L. J. Guibas, "Flownet3d: Learning scene flow in 3d point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 529–537.

[40] A. Medhekar, V. Chiluka, and A. Patait, "Accelerate opencv: Optical flow algorithms with nvidia turing gpus," https://developer.nvidia.com/blog/opencv-optical-flow-algorithms-with-nvidia-turing-gpus/, accessed: 2019-12-05.

Mohammadreza Alipour Sormoli received the M.Sc. degree from the Amirkabir University of Technology (Tehran Polytechnic) in 2017. worked as a research assistant at Koc University and is currently working toward the PhD degree in autonomous driving technology at the University of Warwick (WMG). His research interests include robotics, mechatronics, control and dynamics of autonomous systems.

Mehrdad Dianati (Senior Member, IEEE) is a professor of connected and cooperative autonomous vehicles at WMG, the University of Warwick and the School of EEECS at the Queen's University of Belfast. He has been involved in a number of national and international projects as the project leader and the work-package leader in recent years. Prior to academia, he worked in the industry for more than nine years as a Senior Software/Hardware Developer and the Director of Research and Development. He frequently provides voluntary services to the research community in various editorial roles; for example, he has served as an Associate Editor for the IEEE Transactions On Vehicular Technology. He is the Field Chief Editor of Frontiers in Future Transportation.

Sajjad Mozaffari received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tehran, Tehran, Iran, in 2015 and 2018, respectively. He is currently working toward the PhD degree with the Warwick Manufacturing Group, University of Warwick, Coventry, U.K. His research interests include machine learning, computer vision, and connected and autonomous vehicles.

Roger Woodman is an Assistant Professor and Human Factors research lead at WMG, University of Warwick. He received his PhD from Bristol Robotics Laboratory and has more than 20 years of experience working in industry and academia. Among his research interests, are trust and acceptance of new technology with a focus on self-driving vehicles, shared mobility, and human-machine interfaces. He has several scientific papers published in the field of connected and autonomous vehicles. He lectures on the topic of Human Factors of Future Mobility and is the Co-director of the Centre for Doctoral Training, training doctoral researchers in the areas of intelligent and electrified mobility systems.