Received June 22, 2020, accepted August 14, 2020, date of publication August 24, 2020, date of current version September 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3019187

Visual-LiDAR Based 3D Object Detection and Tracking for Embedded Systems

MUHAMMAD SUALEH AND GON-WOO KIM
Department of Robot and Control Engineering, Chungbuk National University, Cheongju 28644, South Korea

Corresponding author: Gon-Woo Kim (gwkim@cbnu.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) under Grant 2018006154; in part by the Ministry of Trade, Industry and Energy (MOTIE) and the Korea Institute for Advancement of Technology (KIAT) through the International Cooperative Research and Development Program under Project P0004631; and in part by the Ministry of Science and ICT (MSIT), South Korea, through the Grand Information Technology Research Center Support Program, supervised by the Institute for Information and communications Technology Planning and Evaluation (IITP), under Grant IITP-2020-0-01462.

ABSTRACT In recent years, persistent news updates on autonomous vehicles and the claims of companies entering the space, brace the notion that vehicular autonomy of level 5 is just around the corner. However, the main hindrance in asserting the full autonomy still boils down to environmental perception that affects the autonomous decisions. An efficient perceptual system requires redundancy in sensor modalities capable of performing in varying environmental conditions, and providing a reliable information using limited computational resources. This work addresses the task of 3D object detection and tracking in the vehicles' environment, using camera and 3D LiDAR as primary sensors. The proposed framework is designed to operate in an embedded system that visually classifies the objects using a lightweight neural network, while tracking is performed in 3D space using LiDAR information. The main contribution of this work is 3D LiDAR point cloud classification using visual object detector, and an IMM-UKF-JPDAF based object tracker that jointly performs 3D object detection and tracking. The performance evaluation is carried out using MOT16 metrics and ground truths provided by KITTI Datasets. Furthermore, the proposed tracker is evaluated and compared with state-of-the-art approaches. The experiments suggest that the proposed framework offers a suitable solution for embedded systems to solve 3D object detection and tracking problem, with added benefits.

INDEX TERMS Kalman filter, object detection, object tracking, point cloud classification, sensor fusion.

I. INTRODUCTION

The delay in large scale commercialization of autonomous vehicles, circle around the factors pertaining to safety, feasibility and affordability. The push towards driver-less cars has been supported by the prospect of saving human lives. The current car fatality rate in the US is about 1.22 deaths per 100 million miles driven, including safety violation cases [1]. Effectively that sets the benchmark for an autonomous vehicle failure, which remains a huge challenge [2]. Furthermore, the autonomous vehicles require to take decisions while making trade-offs between safety and feasibility. Such as, avoiding lane changes, driving slow at all times, or not to drive at all; may be considered safe but remains infeasible.

The associate editor coordinating the review of this manuscript and approving it for publication was Malik Jahan Khan.

Another challenge in the autonomous vehicles' paradigm is of rising computational demands to process the raw data from sensors in real time. However, a moving datacenter in the name of autonomous vehicle is infeasible [3]. Although, edge computing can offer remote processing of computationally expensive tasks, but with a compromise on security and reliability of information.

An important potentiality of an effective environmental perception is to understand the dynamic properties of coexisting entities. This has put forth huge requirements on the associated research domains related to 3D object detection and tracking. The 3D object detection provides a faithful representation of 3D space around the vehicle, in terms of class, dimensions, and pose. Whereas, tracking enables the estimation of the dynamic parameters. Furthermore, tracking also addresses the issue of temporally missed detections due to the shortcomings of the detector. The key challenges in developing a framework for autonomous vehicles to perform 3D object detection and tracking include, real-time performance, limited computational demand, applicability in variety of weather and lighting conditions, and ease of adapting the change in number and positioning of sensors.

The autonomous vehicles are generally equipped with numerous sensors for environmental perception like ultrasonic, radar, LiDAR (light detection and ranging), cameras, and so on. Among the above sensors, many modern approaches use camera, LiDAR, or a fusion of both for 3D object detection tasks. Although, LiDAR and camera can perform the object detection independently, each sensor possesses some limitations. LiDAR based approaches are susceptible to harsh weather conditions and low resolution [4]–[6]. Whereas camera-based methods are primarily challenged by inadequate light and depth information [7]. Therefore, both sensors require a joint operation to complement the individual limitations, and to enable the applicability in a wider range of environmental conditions [8]–[11].

The Visual-LiDAR based 3D object detection methods adopt either early, late, or deep fusion schemes [12]. The modalities are combined at the beginning of process in early fusion scheme [13], with interdependent representation of data. The late fusion scheme processes the modalities independently up to the last stage where information is fused [10], [11]. The deep fusion schemes tend to mix the modalities hierarchically in neural network layers, allowing the features from different modalities to interact over layers [8], [9], [14]. In order to exploit redundant information of modalities while compensating the individual sensor limitations, late fusion schemes are most appropriate selection.

The tasks of 3D object detection and 3D object tracking are traditionally approached independently. Recent works that jointly perform 3D object detection and tracking tasks are proposed in [15]–[18], reflected on performance leaderboards across various benchmarking platforms. The approach adopted in [15], [16], utilize visual-LiDAR setup for detection, unlike monocular setups for detections in [17], and 3D LiDAR as the only sensor for detections in [18]. The methods being learning based, require laborious and expensive annotation process to prepare training datasets. Furthermore, change in the number, type or positioning of sensors require retraining of the networks. Moreover, the inference time and computational needs limit their use in real-time applications.

Currently, more and more trackers are being proposed that perform tracking in 3D space [19]–[22]. However, these trackers require reasonably accurate 3D detections. Thereby adding the computational demand on the system. The tracker proposed in [19] utilizes 3D IoU thresholding for data association under a 3D Kalman Filter paradigm. The approach proposed in [20] does perform in real-time but at the cost of GPU utilization. Furthermore, a multi-modality approach proposed in [22] focuses on fusion of detected objects information, that remains infeasible from application perspective as multiple networks run for detections. The authors of [21], distinctly addresses the problem of tracking by seeking diversity using detrimental point processes to forecast the trajectories of objects. The main drawback in the existing schemes are the parameters of networks that require training, computational needs, inapplicability on embedded systems, and reliance on 3D object detector performance.

In this work, a comprehensive framework for joint 3D object detection and tracking for an embedded system is proposed. The framework makes use of visual-LiDAR setup to exploit the information redundancy for real-time reliable results. The 3D LiDAR point cloud is represented in a cylindrical grid and possible candidates for objects are filtered. The candidates are tracked and the information of position, pose, dimensions, and class vector is maintained. In parallel, a neural network is employed for visual classification of objects for proposal generation that temporally updates the class vector of the tracked candidates. The framework is an extension of previous work [23], where only LiDAR was considered as the perceptual sensor but lacked in proper classification of objects, resulting in large number of false positives.

The advantages of the proposed approach are in many folds, as challenges pertaining to occlusions and missed visual detections are temporally addressed. Furthermore, even in poor lighting conditions the LiDAR detector continues to operate. Moreover, since no training is involved in direct classification of point clouds, the approach can seamlessly integrate into a variety of sensor arrangement. In addition, the tracker can temporally provide dynamic attributes of the detected objects, that can be directly used for autonomous decisions. The proposed framework is implemented on an embedded system and performance evaluation is carried out using well-established metrics for object detection and tracking on the ground truths provided by KITTI Datasets [24]. The main contributions of this work include a novel 3D object detector, an efficient point cloud processing for object candidate estimation, and clutter aware probabilistic tracking algorithm for an embedded system.

The perception module with efficient 3D object detection and tracking capabilities directly impacts the quality of spatial localization, mapping and motion planning. The localization can benefit from the static part of the environment in conjunction with the redundancy of the sensory setup, especially in a dense and dynamic urban environment. Similarly, regions of the environment pertaining to the dynamic objects can be ignored for mapping. In addition, motion patterns of the dynamic objects in the vicinity can be utilized for safer motion planning [25].

The remainder of the paper is organized as follows: in section II, the proposed architecture is described in terms of hardware/software and information flow. Working of the framework is explained in section III, followed by the specifications on implementation platform in section IV. A detailed description of the proposed framework at modular level is made in section V. The evaluation criteria are defined in section VI, including results and comparisons with state-of-the-art. In section VII framework implementation on the platform is demonstrated. The added features of the proposed framework are discussed in section VIII, followed by conclusions in section IX.

II. SYSTEM ARCHITECTURE

The scope of this work is 3D object detection and tracking, which is part of smart car project that involves a broad V2X based autonomous vehicle architecture. Idea is to let vehicles communicate over V2X protocol and share the environmental information. As in a dense urban situation most part of the environment is occluded by other dynamic objects. Given that all vehicles can share minimal MODT information of the environment, visibility beyond the sensors range can be attained. Furthermore, the fast-paced development of Edge computing and 5G can enhance the computation and communication capacity.

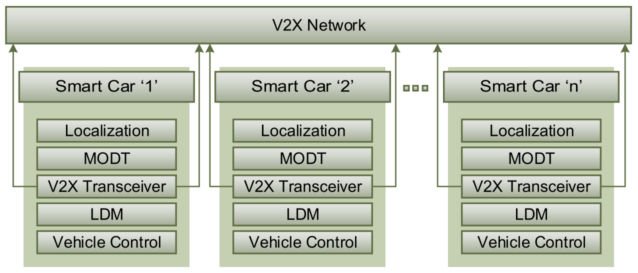


FIGURE 1. MODT in a V2X based architecture.

The proposed MODT scheme is implemented in a Vehicle-to-everything (V2X) based autonomous vehicle architecture shown in Fig. 1, in basic form. Idea is to populate Local Dynamic Map (LDM) to aid controls of individual 'n' number of smart cars in the network and share safety messages. Once the smart car is localized in the map, the local MODT information is fused with the LDM information via V2X transceiver. This provides an environmental perception ranging beyond the sensing capability of single vehicle sensors. This article is focused on MODT framework for a single vehicle, therefore localization, transmission protocols, safety messages and control mechanism will not be discussed further.

III. PROPOSED FRAMEWORK

The objective of 3D MODT is to detect objects by class, dimensions and orientation, and to maintain unique IDs along with the parameters pertaining to the position and kinematics. However, researchers approach 3D object detection and tracking problems separately, evident from the well-established evaluation metrics and leaderboard rankings. The motivation of this work is derived from the notion that the consecutive visual frames in the application areas of object detection in most cases are temporal. That is, scene does not change abruptly, rather appearance of an object in the scene remains visible for several frames. Furthermore, application areas such as autonomous vehicles largely benefit from the dynamic information of the detected objects. This usually is addressed by a tracker that maintains a unique ID for a detected object and predicts the motion patterns of the detected objects. Thus, accurate 3D object detection without tracking provides no information regarding object motion.

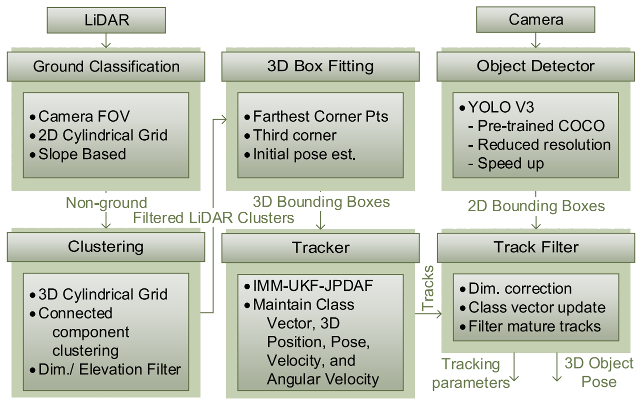


FIGURE 2. 3D MODT framework.

The proposed framework addresses the 3D object detection and tracking problem jointly in a temporal fashion, information flow is shown in Fig. 2. The framework runs on two threads, associated with LiDAR and Camera inputs respectively. The LiDAR point cloud is treated with ground removal and clustering to predict initial pose and dimensions of potentially trackable objects. The centroids of the objects are considered as measurements for the IMM-UKF-JPDAF based tracker. The second thread in parallel predicts visual detections in the image, providing localized bounding boxes and class information. Instead of assigning a fixed class and dimensions to an object in a single frame, a tracked object is assigned a class, whereas parameters pertaining to dimensions, pose, and velocity are updated temporally across multiple time frames. The tracking information is merged with visual detections to provide 3D object poses along with associated tracking parameters.

IV. PLATFORM FOR 3D MODT IMPLEMENTATION

The proposed framework is implemented and tested on Hyundai i30 (Hyundai Motor Company, Seoul, South Korea), shown in Fig. 3. Platform is equipped with OS1-64 Ouster LiDAR (Ouster, San Francisco, CA, USA), mounted on the center top of platform. For visual perception, ZED camera (Stereo Labs, San Francisco, CA, USA) is mounted beside LiDAR inside a custom-made casing. The sensors provide raw measurements to Jetson AGX Xavier unit by Nvidia (Nvidia Corporation, Santa Clara, CA, USA), that performs the computations associated to the proposed framework. Furthermore, vehicle CAN is interfaced along with V2X Modem to perform V2X communication. The framework is developed to operate on ROS (robot operating system) "Melodic Morenia" middleware on top of Ubuntu Linux 18.04.1. The GPU of Xavier is utilized through CUDA 10.0 libraries for visual detections. Whereas, LiDAR preprocessing and tracking tasks are handled by NVIDIA Carmel ARM CPU processors.

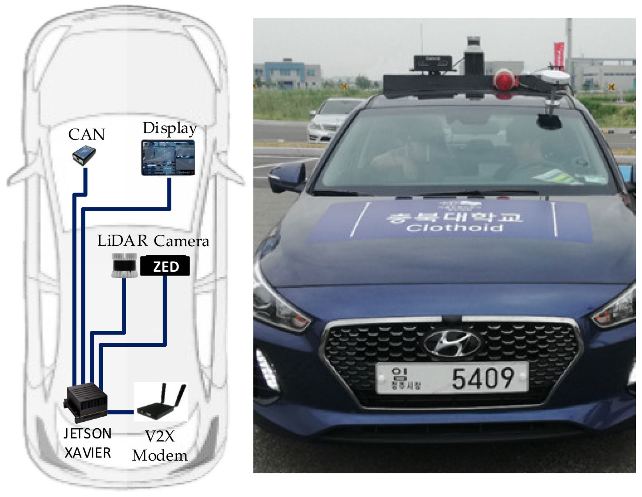


FIGURE 3. Sensor setup and implementation platform.

V. 3D MODT

The proposed 3D MODT framework comprises of two threads pertaining to the processing of 3D LiDAR and camera respectively. The thread for processing of 3D LiDAR point cloud is composed of sub modules, ground segmentation, clustering, box-fitting, and tracking. Whereas, the thread responsible for treating images from camera is composed of YOLO v3 [42] implemented as a ROS package to provide object class information. The operation and structure of each sub module is explained in the subsequent subsections.

A. GROUND CLASSIFICATION

The ground classification is an essential pre-processing task in which LiDAR point cloud is partitioned into ground and non-ground measurements. The portion of point cloud classified as ground can be further processed for road markings, curb detection, traversable area, and path planning tasks. Whereas, part of point cloud corresponding to non-ground LiDAR measurements is effectively used for the tasks pertaining to 3D object detection. Several approaches for ground classification exist in the literature that largely vary in terms of sensor setups and assumptions made for the environment.

The prominent strategies for ground classification utilize scan-rings [26], voxels [27], height threshold [28], or feature learning [29]. The scan-ring based approaches are generally applicable on single LiDAR setups, in which distance between consecutive scan lines are studied for ground classification. Whereas, voxelization of point cloud into 2D or 3D space is also a common practice to scale down the number of measurements for estimation. Similarly, with the assumption of planner ground environment, setting a height threshold is enough for ground classification. On the other hand, some approaches utilize neural networks to address the classification of the sparse LiDAR point cloud. In this work, possibility of ground to be non-planner is considered and point cloud is assumed to be a merger of multiple calibrated LiDARs that are arbitrarily positioned. This assumption rules out the approaches that rely on height threshold and scan-rings. Furthermore, the variability in number and positioning of LiDAR sensors, and constraints of embedded computing limits the use of learning-based approaches.

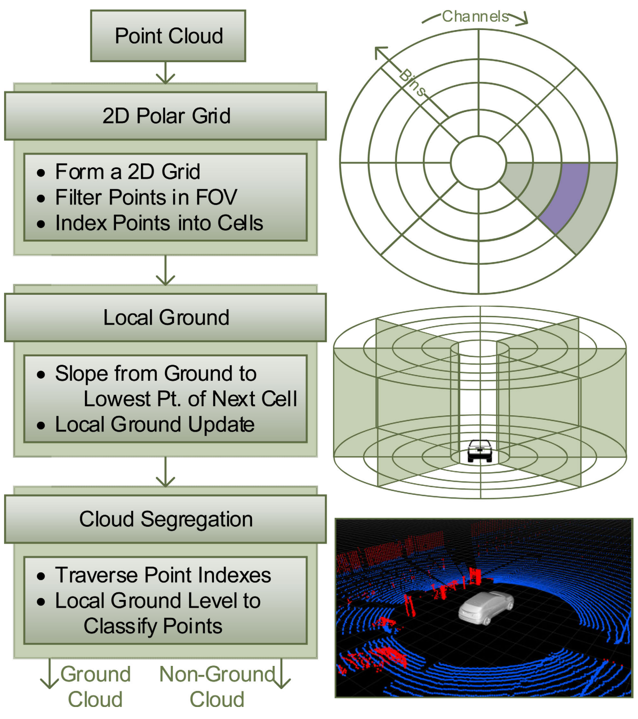


FIGURE 4. 2D polar grid for ground classification.

The approach adopted in this work involves indexing of point cloud into a 2D array that processes the classification task efficiently. Each cell of an array contains indexes of the point cloud measurements that belong to a section of vertically sliced cylinder into channels and bins, as shown in Fig. 4. Each channel is traversed independently, directed outward from the vehicle, to estimate the ground level in each cell of the grid. The sensor height from the ground is considered as the initial ground level, and slope to the lowest measurement of consecutive cell is computed. The slope exceeding a threshold related to a cell containing non-ground measurements and previous ground level is maintained. Whereas, slope within a threshold limit updates the ground level for subsequent cells. With all cells of the grid getting the ground level, point cloud is segregated with a tolerance parameter to remove the edge noise.

The proposed ground classification module in comparison to the module developed in prior work [23], is optimized further to reduce the process time to about one half. The approach in former work for ground classification was similar, however the point cloud was traversed multiple times for data representation into cylindrical grid, labeling of grid cells, and labeling of LiDAR points respectively. The ground classification module evaluated on the similar datasets processed the data in an average time of 7.8ms, that former to optimization consumed 15.7ms. Similarly, the process time on embedded system reduced to an average of 39.1ms from 64.9ms. The execution times are further expressed in the evaluation section. The key modifications for optimization are,

- The lowest and highest point, of each cell is found when indexes of the point cloud are being distributed into grid.
- Instead of traversing each channel twice, estimation of slope and local ground level along the bins are carried out in a single traversal.
- The iterators for the point cloud indexes are efficiently utilized to form point clouds for ground and non-ground points, skipping the step of labeling the bins.

In addition to the processing speed of computational platform, and optimized programming approach, the parameters for data representation contribute in the overall processing time. This include the range of measurements from sensor R_{range}, number of LiDAR measurements in a time step, LiDAR field of view to be considered FOV, and resolution of grid-based representation, expressed in terms of grid cell area,

$$G_{Acell} = \frac{FOV \cdot \pi}{channels \cdot 180}(R_i^2 - R_{i-1}^2), \tag{1}$$

$$R_i = \frac{R_{range}}{Bins} \cdot i, \quad where\ i = 0, 1, 2, \ldots Bins \tag{2}$$

The number of bins and channels determine the area and number of grid cells that need to be traversed for slope test and local ground estimation. Higher resolution demands additional processing, whereas lower resolution provides compact representation. Once the ground is classified, the point cloud pertaining to non-ground LiDAR measurements is presented to the clustering module.

B. CLUSTERING

The concept of clustering is to group the entities based on some similarity [30]. Clustering a LiDAR point cloud such that each cluster corresponds to a unique object is a challenging task, due to sparsity and lack of textured information. The clustering approaches generally utilize connectivity [28], centroid [30], density [31], distribution [32], or learned features of the LiDAR measurements. Connectivity or hierarchy-based approaches rely on proximity of neighboring measurements and expand iteratively.

The centroid-based approaches require prior knowledge of the number of clusters to divide data into, such as K-means, Gaussian Mixture Models, and Fuzzy c-mean. While, density-based approaches identify high density regions for clustering, but density of LiDAR measurements radially decrease as a function of distance from the sensor. Moreover, occluded measurements further affect the densities, thus density-based clustering approaches are not effectively applicable in 3D object detection paradigm.

The distribution-based clustering methods utilize distribution models to fit potential clusters of objects, providing more information compared to density-based methods, but at the cost of complexity. However, absence of distribution model and measurements under partial occlusion suffers in proper clustering. Similarly, learning based approaches trains a neural network for a set of optimization functions/criteria to cluster or per point classification such as [14], [33]. The approaches prove to be effective for 3D object detection tasks but require excessive computational resource, beyond the constraints of embedded platforms.

In this work, the LiDAR point cloud is clustered using connectivity-based approach. To reduce the complexity, 3D cylindrical grid is used instead of point wise clustering. The advantage of 3D grid over 2D grid is to cater the measurements pertaining to elevated structures, like traffic lights and bridges. Furthermore, cylindrical grid can address the sparsity of measurements that are far from the sensor. The point cloud for clustering is represented in a 3D array, where each cell contains the corresponding indexes of points.

The 3D array is processed through a 3D connected component clustering approach to group the grid cells in proximity. The formulation of clustering traverses all the cells of 3D array and examines the immediate neighbors for minimum number of cells to include in a cluster. The clusters of point cloud are filtered based on dimensions, large clusters generally correspond to buildings while very small clusters either belong to noise, insignificant obstacles, or over segmentation. Furthermore, the clusters elevated from the ground are also filtered, as intention is to track the moving objects on the ground. The remaining clusters are treated with the box fitting task to estimate the pose of object and the centroid, further explained in the subsequent subsection.

Like ground segmentation, clustering module developed in previous work [23], is optimized and process time is substantially reduced. The clustering module developed in the former work used a rectangular grid-based representation of point cloud, requiring relatively higher resolution. Furthermore, to cluster the occupied grid cells, all 26 neighbors of each cell were traversed for occupancy check. The average process time of clustering together with the box fitting task on similar datasets is reduced to 3.66ms from 14.3ms, and on embedded system the process time is reduced to 17.3ms from 31.18ms. The execution times at modular level are further explained in the evaluation section. The key modifications in clustering module are listed below,

- Rectangular grid is replaced with a 3D cylindrical grid, to exploit the point cloud of single LiDAR instead of merged point cloud of three LiDARs.
- Instead of searching 26 neighbors of grid cell for clustering, 6 immediate neighbors are traversed.

In the clustering process, unlike 2D representation of LiDAR data for ground classification, 3D or volumetric representation is adopted. In addition to the bins and channels, the vertical range V_{range} of LiDAR cloud is divided into layers. The number of LiDAR measurements however are reduced to only non-ground measurements within FOV and range R_{range}. The volume of grid cell is then represented as,

$$G_{Vcell} = \frac{FOV \cdot \pi}{channels \cdot 180} \cdot \frac{V_{range}}{levels}\left(R_i^2 - R_{i-1}^2\right), \tag{3}$$

The clustering method adopted in this work requires that the adjacent cells of the grid pertaining to unique objects are populated with LiDAR cloud indexes. Therefore, optimal resolution parameters are desired to set G_{Vcell}, as higher resolution results in over segmentation, despite increased computational resource. Whereas, low resolution representation tends to cluster LiDAR measurements pertaining to objects in proximity as single object. Therefore, the volume G_{Vcell} provides a balance between performance and computation time.

The clustering module includes the task of box fitting that greatly affects the overall performance of tracking. Even if the tracked object undergoes a partial occlusion, the dimensions and pose history of the tracked object still contribute in recovering the accurate centroid and pose, handled by the track management module.

C. BOX FITTING

Box fitting of clustered LiDAR point cloud data is an essential and challenging task, as the measurements are always occluded because of obstructions in the sensor line-of-sight. An efficient box fitting technique estimates the correct object pose and centroid, considering the partially measurements. Several approaches tend to address this problem by either model-based [34] or feature-based methods [35]. The model-based methods match the raw point cloud with a known geometric model, whereas features-based approaches exploit the edge features to estimate the pose. Lack of generality and excessive computational requirement bars the use of model-based approaches in MODT applications. Similarly, the selection of features that best describe the object pose is a difficult task. Currently, neural networks are also trained for feature selection process, as utilized in [10]. However, change in sensor setups often require labeling of datasets and retraining of networks.

In this work, considering the computational constraints a feature-based method is utilized, that performs the L-shape point cloud fitting within a minimum rectangle area. Initially, the indexes of points with coordinates that define the minimum box fitting are traversed to identify the corners of clustered point cloud on the horizontal axis. The farthest corners based on dimensions and location of object cluster are used to formulate a line, and all points of the cluster are traversed to find the farthest point from the line as a third corner. Using the three corners, dimension of the bounding box and centroid is updated. Lastly, the pose of clustered object is calculated about the updated centroid. Since the presence of occlusions affect the correct pose and centroid estimation, the tracker module maintains the history of tracked object and heuristically adjusts the dimensions and pose of object temporally. The information flow is expressed in Fig. 6, where point a and b are the farthest points of the cluster identified by the maximum and minimum coordinates of the cluster respectively.

The box fitting task is performed within clustering module and finding the farthest points in the clusters contribute in overall process time. The key factors modified to acquire optimized process time are as follows,

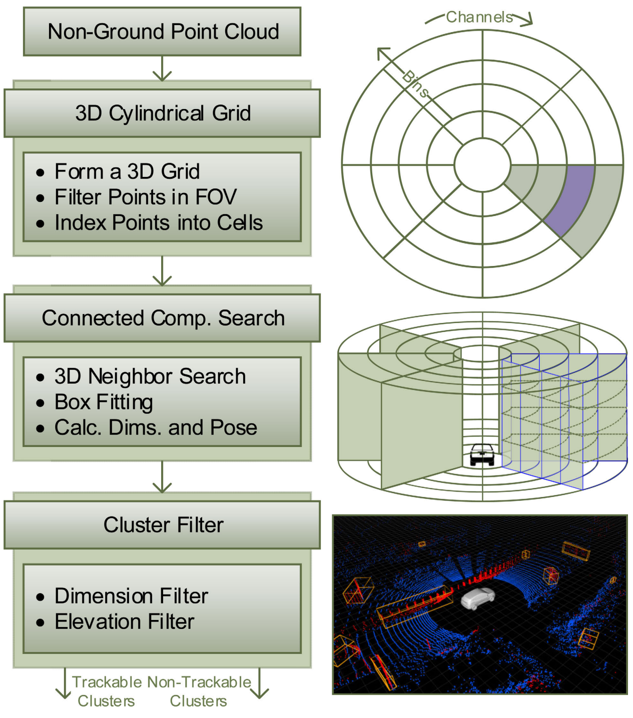


FIGURE 5. Cylindrical grid for clustering.

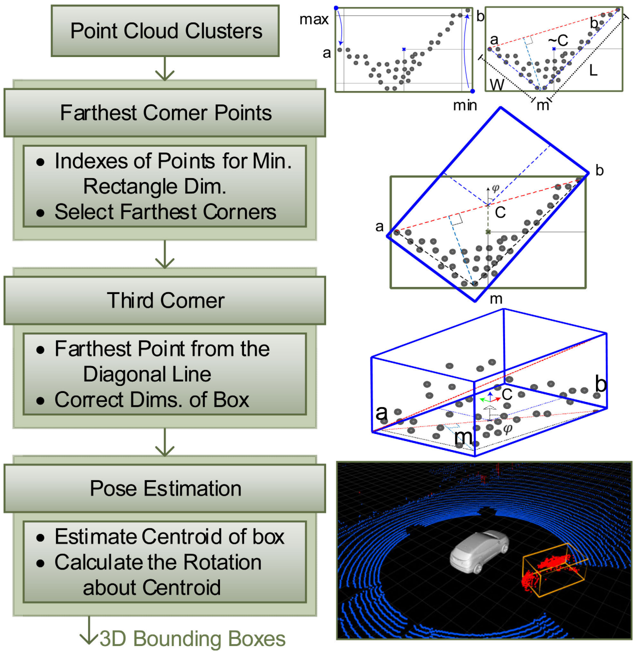

Point Cloud Clusters
Farthest Corner Points
• Indexes of Points for Min. Rectangle Dim.
• Select Farthest Corners
Third Corner
• Farthest Point from the Diagonal Line
• Correct Dims. of Box
Pose Estimation
• Estimate Centroid of box
• Calculate the Rotation about Centroid
3D Bounding Boxes
max
min
a
b
C
~C
L
W
m
φ

FIGURE 6. L-shape box fitting.

- Instead of traversing all the points, the points corresponding to the minimum and maximum coordinates of the cluster are exploited.
- The farthest points of the cluster are heuristically found by making use of dimensions and cluster position with respect to the sensor.

D. TRACKING

The multi-object tracking is an essential component in the perception pipeline of autonomous vehicles. The object tracking capabilities can enable the system to make better decisions of actions to perform in cluttered environments. An extensive literature on 2D MOT algorithms exist that focuses on object tracking in image plane [36], where the objective is to maintain unique identities and to provide temporally consistent location of detected objects. However, tracking of objects in 3D space is becoming popular in the research community [19], as more and more 3D MOT schemes are being proposed. The 3D MOT systems in general share the similar components of 2D MOT systems, a part form the distinction of object detections in 3D space instead of image plane. This potentially allows the design of motion models, data association, occlusion handling, and trajectory maintenance directly in 3D space without perspective distortion.

An autonomous vehicle acts like a dynamic system that is required to track objects in the environment. In this scenario, the tracked objects tend not to follow a regular motion pattern, giving rise to motion uncertainties. Similarly, the cluttered environments and sensor limitations impose partial or complete occlusion of objects that adds up the uncertainty in position and pose of the tracked objects. To perform tracking in the presence of uncertainties, Bayesian filtration strategies are usually deployed. Where the state estimation is carried out with either the assumption of Gaussian Mixture for density or Gaussian distribution for transition. The assumption leads to the use of Gaussian mixture Probability Hypothesis Density Filter (PHDF) [37] or Joint Probabilistic Data Association Filter (JPDAF) [38] for state estimation, respectively. Whereas, for non-Gaussian assumption Particle Filter (PF) methods are used [39]. The states of the tracked objects are updated after association between tracks and detection information is established.

In this work, like the former implementation in [23], the uncertainties due to clutter are addressed with an assumption of Gaussian distribution, and JPDAF is applied for data association. Similarly, the uncertainties due to motion is handled by an Interacting Multiple Model (IMM), to perform non-linear prediction of states for tracked objects. To cater the non-linearities of motion models for Gaussian process, an Unscented Kalman Filter (UKF) is utilized. The implementation of IMM-UKF-JPDAF is an approach that efficiently addresses the problem of recursively estimating states and mode probabilities of targets, described by a jump Markov non-linear system, in the presence of clutter.

The trackable objects are assumed to follow r motion models $M = \{M_j\}_{j=1}^{r}$, represented as a non-linear stochastic state space model,

$$x_{k+1} = f_j(x_k, u_k) + w_{j,k}, \tag{4}$$

$$z_k = h_j(x_k, u_k) + v_{j,k}. \tag{5}$$

That operates the system function f_j and measurement function h_j, with the input vector $u_k \in \mathbb{R}^p$, state vector $x_k \in \mathbb{R}^n$, and measurement vector $z_k \in \mathbb{R}^{\mathrm{q}}$ at each time step k. Where the zero-mean Gaussian noise sequences $w_{\mathrm{j,k}}$ and $v_{\mathrm{j,k}}$ are mutually independent covariance matrices $\mathrm{Q_{j,k}}$ and $\mathrm{R_{j,k}}$ respectively. Moreover, the progression of system among r models is considered as first order Markov chain with time invariant Markovian model transition probability matrix:

$$\Pi = \begin{pmatrix} p_{11} & \cdots & p_{r1} \\ \vdots & \ddots & \vdots \\ p_{1r} & \cdots & p_{rr} \end{pmatrix} \in \mathbb{R}^{\mathrm{r\times r}}. \tag{6}$$

The elements of the matrix p_{ij} represent mode transition probability from model i to j.

The proposed IMM-UKF-JPDAF tracker follows a five-step process: (a) interaction, (b) state prediction and measurement validation, (c) data association and model-based filtering, (d) mode probability update, and (e) combination step. A similar approach for a single target is explained in [40], that addresses data association of measurements to a single tracked object. In comparison, a JPDAF is deployed in the proposed framework to perform tracking of multiple objects. This requires computation of association probability between each track and measurement, while considering all feasible joint association events across all measurements, leading to a combinatorial explosion problem.

To mitigate the possible combinatorial explosion, clustering technique is adopted, where the association matrix is clustered into the sets of marginal $\theta_{j,q}$ and joint association events $\Theta = \bigcap_{j=1}^{N_k} \theta_{j,q_j}$. The number of clusters equal the sum of marginal and joint association events. The clustering technique helps in mitigating the combinatorial explosion of hypothesis that naturally grows in the cluttered environments. Furthermore, the covariance of a track prediction increases with unassociated measurement in consecutive time steps, consequently increasing the gate area for association. The larger gate area results in larger number of joint association events.

The hypothesis of all possible occurrences of events within every cluster, marginal association probabilities are computed, that is the probability sum of the joint association events; given that the measurement j belongs to track q:

$$\beta_{jq}^{cl} = \sum_{\Theta}^{\mathrm{cl}} \mathrm{P}\left\{\Theta^{cl}|z_k\right\} \hat{\omega}_{jq}^{cl}\left[\Theta^{cl}\right],$$

$$j = 1, \ldots, N^{cl} \quad \text{and } q = 1, \ldots, \mathrm{T}^{cl}, \tag{7}$$

$$\mathrm{P}\left\{\Theta^{cl}|z_k\right\} = \frac{1}{c} \prod_{j=1}^{N^{cl}} g_{jq} P_D \prod_{q=1}^{T^{cl}} (1 - P_D)^{\delta_q} \prod_{j=1}^{N^{cl}} \beta^{\phi_j}. \tag{8}$$

where, $\hat{\omega}_{jq}^{cl}$ represents the joint association event within the cluster cl of N measurements and T tracks. Furthermore, g_{jq} is the likelihood of measurement j being associated to track q, normalized by a factor c. Moreover, δ_q and ϕ_j represent the number of unassociated tracks and measurements, respectively, within the cluster. Subsequently, weighted measurement residual $\tilde{z}_{I,q,k}$ is computed for each corresponding model i of filter according to the associated measurement set:

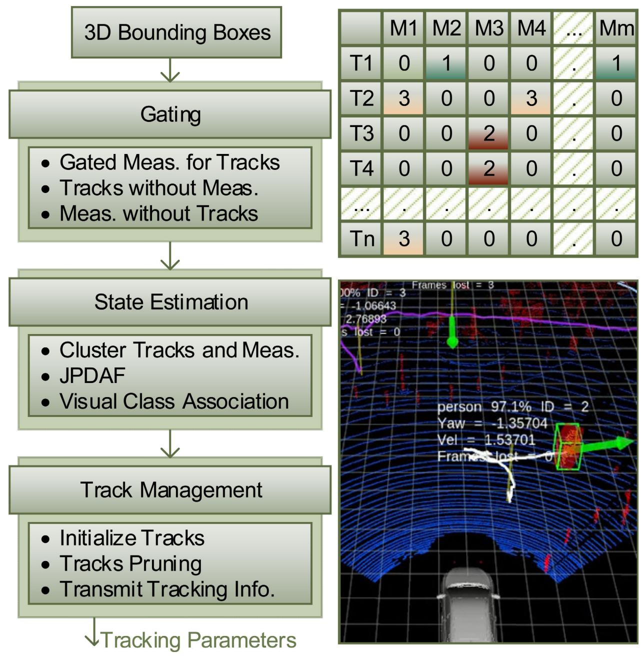
$$\tilde{z}_{i,q,k} = \sum_{j=1}^{N^{cl}} \beta_{j,q}^{cl} \tilde{z}_{i,j,q,k} \tag{9}$$

The cross-covariance matrix C_{x_k,z_k} between predicted states and measurements are used together with innovation covariance matrix S_k to calculate the optimal Kalman gain K_k. Subsequently, the states and covariances of each model for the corresponding tracks are updated.

$$K_k = C_{x_k,z_k} S_k^{-1} \tag{10}$$

$$x_{i,q,(k\,|\,k)} = x_{i,q,(k\,|\,k-1)} + K_{i,q,k}\tilde{z}_{i,q,k} \tag{11}$$

The states, covariances, and mode probabilities of IMM-UKF-JPDAF are recursively estimated with the help of individual model likelihoods. The individual filter states and covariances are combined into a single weighted output using mode probabilities of tracks. The flow of tracker module is elaborated in Fig. 7, along with the clustered association matrix, forming three association clusters as sub problems for the tracker. Furthermore, a tracked object is shown that presents the tracking parameters along with the class and confidence percentage.



	M1	M2	M3	M4	...	Mm
T1	0	1	0	0	.	1
T2	3	0	0	3	.	0
T3	0	0	2	0	.	0
T4	0	0	2	0	.	0
...	.	.	.	.	.	.
Tn	3	0	0	0	.	0

FIGURE 7. IMM-UKF-JPDAF tracker with clustered associations.

The execution times of tracker module mainly rely on the number of maintained tracks. The former implementation [23] without visual classification, pruned the tracks merely based on inconsistent measurements, resulting in large number of tracks to maintain. With an additional condition of tracked object being classified by visual object detector, reduce the number of tracks, resulting in decreased process time. The average execution time of tracker including track management and visual class association on similar datasets consumes 6.4ms and 18.5ms on desktop and embedded computing platforms, respectively. Whereas, in former implementation consumed 8.3ms on desktop and 24.6ms on embedded system respectively, without visual class association. The key factor responsible in optimizing the process time is of an efficient track management module that limits the false positive measurements for tracking.

E. OBJECT CLASSIFICATION

The paradigm of fusing multiple modalities followed in this work can be regarded as late fusion, where the tracked clusters of point clouds are classified. The classification of tracked point cloud clusters rely on two components; class association and class management.

- The class association involves the process of assigning a visually detected objects' class to the tracked point cloud cluster.
- The class management utilizes the assignment history to maintain and select a class for the tracked object.

The visual object detection is carried out using YOLO-v3 [42] that is pretrained in Microsoft COCO datasets [43] and detection classes are limited to trackable dynamic objects. YOLOv3, uses Darknet-53 (a CNN model with 53 convolutional layers) backbone, and delivers 57.9 mAP (AP50) on Microsoft's COCO dataset, using an input resolution of 608 $\times$ 608 pixels. The optimized variants of the network can perform real-time on embedded systems but at the cost of compromised accuracy.

In this work, the input image resolution is tuned to 416 $\times$ 416 pixels that maintains the process time well below 100 milliseconds on the embedded system. Although, reduction in input image resolution results in missed detections due to size, saturation, and exposure issues in image. The tracker module handles the missed detections with the class vector that probabilistically assigns the class to objects. In addition, the LiDAR range is limited to 60-80m, beyond this range point cloud is too sparse for accurate estimation of object dimensions and pose. Visual detector detecting an object beyond this range only adds an additional complexity for the class association process.

Let T^k and D^k be the sets of maintained tracks and visually detected objects respectively at time step k. The corrected centroids of tracked clusters o_i^k are projected onto the image frame of corresponding time stamp, resulting in a 2D pixel location in image $\bar{o}_i^k$. Similarly, the localization and dimensions of visually detected objects are used to calculate the centroids m_j^k. Using the 2D centroids of both sources the Euclidian distance cost matrix $E^k = \left[c_{ij}^k\right]$ is populated, where $i = \{1, 2, \ldots, T\}$ and $j = \{1, 2, \ldots, D\}$. Furthermore, constrained by the criterion that at least 30% of overlap exists among the corresponding 2D bounding boxes.

$$c_{ij}^k = \begin{cases} d(\bar{o}_i^k, m_j^k) & if iou\left(\bar{t}_i^k, d_j^k\right) > 0.3 \\ 1000 & otherwise \end{cases} \tag{12}$$

Following the Munkres association strategy for optimized minimum cost is performed and set of index pairs Υ pertaining to associated tracks t_i^k and visual detections d_j^k is obtained. Using the set Υ class association matrix $\hat{E}^k = \left[\hat{c}_{ij}^k\right]$ can be formulated such that,

$$\hat{c}_{ij}^k = \begin{cases} v_j & if \ < i, j > \subset \Upsilon \\ 0 & otherwise, \end{cases} \tag{13}$$

where, v is the number that represents the class of visually detected object and the dimension index of class association vector $A_v^i = (a_1^i, \ldots a_n^i)$. The matrix $\hat{E}^k$ is used to update the class association vector A^i with an increment in the associated class dimension,

$$\hat{c}_{ij}^k = \begin{cases} a_{\hat{c}_{ij}}^i & if \ \hat{c}_{ij}^k = 0 \\ a_{\hat{c}_{ij}}^i + 1 & otherwise, \end{cases} \tag{14}$$

The updated vector A^i, along with the age of track t_{age} is utilized to compute the class certainty P_c^i of tracked objects and the ratio P_o^i of object, that reasons the tracked object to be an outlier.

$$P_c^i = \frac{\max(a_v)}{t_{age}} \tag{15}$$

$$P_o^i = \frac{\left(t_{age} - \sum_{v=1}^{n} a_v^i\right)}{t_{age}} \tag{16}$$

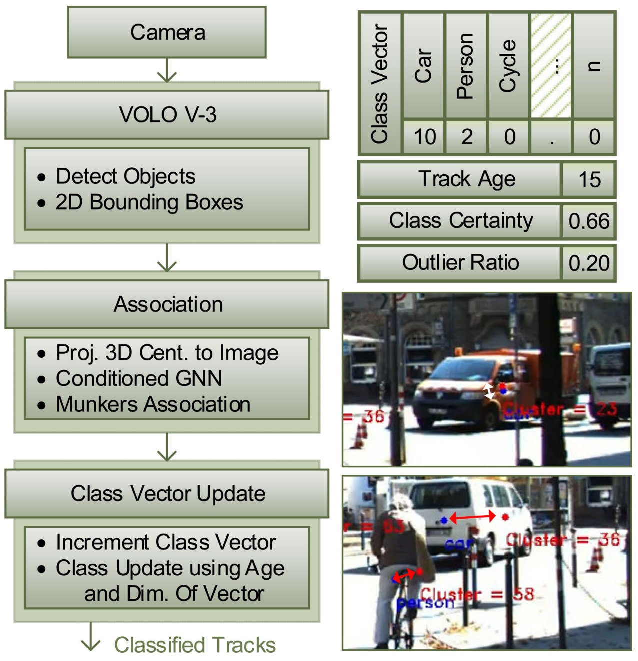


FIGURE 8. **Visual object detector to classify tracked objects using class vector.**

In Fig. 8, the flow of object classification is presented that runs in parallel with the LiDAR based detection tracking thread. Furthermore, the class vector of a mature track is shown that represents the age and counts of class associations resulting in the class certainty of 66.6%. Moreover, the red and blue dots in the image represent the projected centroid of cluster, and center of visually detected objects respectively.

The tracked objects maintain a class vector with each dimension registered to a visually detectable class. At the fusion step, after a successful association, a unit increment is made in the corresponding class dimension of the vector. The maximum count is a dimension of the class vector specifies the object class, whereas the certainty is computed against the life of the track.

F. TRACK MANAGEMENT

The MODT task in the presence of uncertainties pertaining to classification, clutter, and motion of objects require a robust track managing module to maintain and provide reliable information. The main purpose of track management module is to initialize and maintain track statistics, occlusion handling of the tracked object, and pruning out of tracks pertaining to false positive measurements.

The track managing module initiates new tracks for unassociated measurements with unique identity, and records track age in terms of frame count. In addition, the dimensions and pose of the tracked object are retained while considering the LiDAR properties. The measurements related to the objects moving farther from the sensor tend to experience increased occlusions, and report decreased dimensions. On the other hand, objects approaching closer to the sensor get more exposure and provide comparatively more accurate dimensions. Similar pattern is observed in the estimated pose of the object and is handled by smoothing the sudden changes in the yaw angles. The accuracy of maintained dimensions and pose aids the occlusion handling and centroid correction. As the centroid of the measurement pertaining to the object under occlusions is also shifted proportional to the change in the dimensions. By capitalizing on the sensor characteristics and maintained information the centroid C of the tracked object is corrected using change in length ΔL, width ΔW, height ΔH and yaw φ.

$$C_x' = C_x + \Delta L \cdot \frac{\cos(\varphi)}{2}, \tag{17}$$

$$C_y' = C_y + \Delta W \cdot \frac{\sin(\varphi)}{2}, \quad and \tag{18}$$

$$C_z' = C_z + \Delta H. \tag{19}$$

The tasks associated with the track management module are presented in Fig. 9, with an example of centroid correction. The mature track of an object retains the dimensions as width W and length L, compared with the measured dimensions W_m and L_m to find the difference between ΔW and ΔL. The change in dimensions is utilized together with the position of object relative to the sensor to acquire the correct centroid C'. Furthermore, the mature tracks of objects represent the measured dimensions by a wire frame bounding box, along a red box with correct dimensions and centroid. The initialized track requires measurement association for consecutive five time-steps to be regarded as a mature track. If a track misses an association measurement without getting a classification

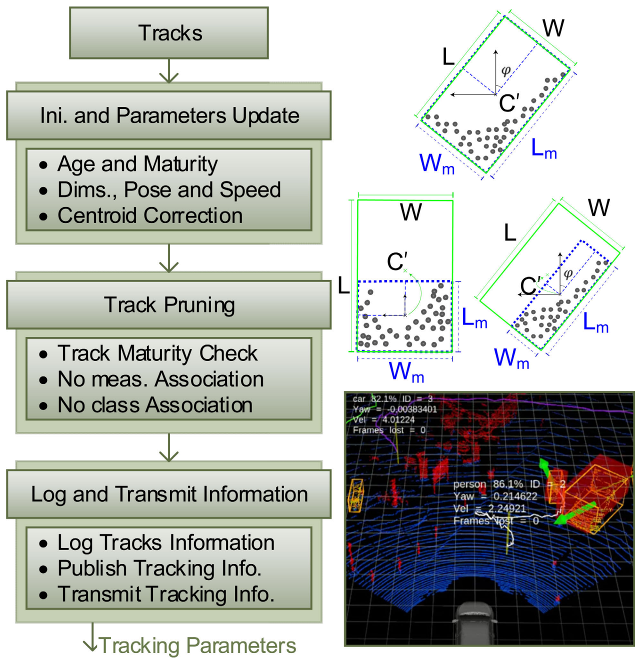


FIGURE 9. Tracks manager with pose, dimensions, and centroid corrector.

from a visual detector within the maturity period, the track is pruned out. Moreover, the ratio P_o^i for all mature tracks getting higher than 60% are filtered out as outliers. Furthermore, mature tracks that share common measurements for consecutive five time-steps results in pruning of inconsistent, or younger track. The trajectory of tracked object including the position, pose, and time is stored, and utilized to calculate relative heading direction, velocity, and angular velocity of the tracked objects at every time-step, as shown in Fig. 9. In addition, class vector A^i is updated that provides the class certainty P_c^i to the tracked objects.

VI. EVALUATION

The KITTI datasets [44] are widely accepted as a standard evaluation platform for MOT tasks. However, a tool to evaluate 3D MODT systems directly in 3D space is not provided by KITTI dataset. The convention of evaluating 3D MOT systems is to project 3D tracking results to the 2D image plane to perform evaluation. Recently, an extension to the official KITTI 2D MOT is proposed in [19], where the cost function is modified from 2D IoU to 3D IoU, and tracking evaluation is performed directly in 3D space. As the proposed framework performs MOT in 3D space, the extension of 3D MOT evaluation is more relevant. In the proposed framework however, MODT is jointly performed where an object while being tracked may require several time steps before getting an accurate class, dimensions, and orientation. To evaluate the true potential and fair comparison with state-of-the-art systems, MODT is evaluated on the KITTI datasets with ground truths [44], based on the MOT16 evaluation metrics proposed in [41]. Whereas, the MOT component is independently evaluated using off-the-shelf 3D object detector [45], [46] against the 3D MOT evaluation extension [19].

The metrics used to evaluate the proposed MODT are: (a) tracker to target assignment, (b) multi object tracking accuracy MOTA, (c) multi object tracking precision MOTP, and (d) track quality. The metric for tracker to target assignment measures the number of False Positives (FP), False Negatives (FN), and ID Switches IDSW. Where, FP and FN deals with incorrect associations of the measurements. Furthermore, the IDSW determines the number of ID switches across all the fames for an object, the metric for MOTA is computed by:

$$\mathrm{MOTA} = 1 - \frac{\sum_t (\mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDSW})}{\sum_t \mathrm{G}_t}, \tag{20}$$

where, t is the frame index and G is ground truth value. The negative MOTA indicates that the number of errors has exceeded the actual number of objects, maximized at 100. Furthermore, the metric MOTP is evaluated by the average 3D IOU of the tracked object. The metric for track quality is described by classification of a track into: Mostly Tracked (MT), Partially Tracked (PT), and Mostly Lost (ML). This measures the extent of ground truth G trajectory recovered by the tracker. A target is MT if it is successfully tracked for at least 80% of its life span. Where, IDSW number is irrelevant in this metric, as the ID needs not to remain the same throughout the track. If the recovered track is for less than 20% of its total length, it is said to be (ML). Other tracks fall under the class of (PT). A higher number of (MT) and few (ML) is desirable.

In this work, a criterion is proposed that overlaps the ground truth information pertaining to the camera and LiDAR frame as a reference. Hence, a subset of ground truth is attained such that: (a) object is in FOV of LiDAR and camera, (b) existence of object within 40 meters range from the sensor, and (c) the lifetime of a track is specified by the duration of first two conditions being true. The range criterion is set to 40 meters range, as the Velodyne HDL-64E sensor used in KITTI datasets has the effective measurement range of 20-40 meters [47]. Furthermore, instead of evaluating the detected objects and corresponding tracks framewise, the trajectories of objects are compared with the provided ground truths. As the proposed framework temporally assigns the class to the tracked objects. Since, the dimensions of objects are not provided frame-wise in the ground truth. The dimensions of tracked objects evaluated across the life of track are used for evaluations. The raw data provided by the KITTI datasets under the category of 'City', with the ground truth annotations are used for being more relevant to this implementation. However, the objects of type 'Tram', 'Misc', and 'person sitting' are excluded for evaluation, and contribute to FP if detected and tracked.

A. EVALUATION RESULTS FOR MODT

The raw KITTI Dataset is provided with the ground truth for tracking information structured in XML format. Moreover, a support to generate similar XML file directly from

TABLE 1. Tracking evaluation with datasets information.

Dataset	Frame Count	Objects	MOTA (%)	MOTP (%)	Error (m)	FP	FN	IDSW	MT (%)	PT (%)	ML (%)
1	114	8	86.1	69.2	1.7	22	24	0	63.6	9.1	27.2
2	83	2	95.5	61.1	0.3	2	0	0	100	0	0
5	160	14	83.3	81.2	0.7	98	18	0	93.3	0	6.6
9	453	60	80.5	71.7	1.8	172	331	1	49.4	20	30.6
13	150	4	90.8	61.8	0.4	21	1	0	80	0	20
17	120	2	91.1	81.2	1	2	2	0	66.6	0	33.3
18	276	12	92.5	78.8	1.2	17	13	0	78.5	7.1	14.3
48	28	5	92.7	86.4	0.73	2	8	0	50	12.5	37.5
51	444	20	80.9	64.4	2.7	115	107	1	28.9	15.6	55.6
57	367	20	74	78.73	0.77	89	222	1	57.7	19.23	23.1
	2195	**147**	**86.7**	**73.5**	**1.1**	**540**	**726**	**3**	**66.8**	**8.4**	**24.8**

tracking algorithm is provided. To perform the evaluation and benchmarking, the MODT framework is programmed to produce an XML file similar in format to that of ground truth. In addition, MATLAB wrapper is also offered by KITTI Dataset to extract tracking information from the XML files to perform the evaluations. The evaluation results of 10 dataset sequences, along with dataset recording number, frame count and the number of trackable objects that qualify the defined criterion are tabulated in Table. 1.

The evaluated metrics in Table. 1 reflect that MODT algorithm performs reasonably within a 40 meters range. However, metric scores drastically degrades with an increase in range. The main contributors are the FP and FN, as IDSW notably remain low. The error metric specifies the average Euclidian distance from the object centroids of the detector and the ground truth at every time step. The range of centroid error being around a meter shows that the measurements of detector lies within proximity of object under occlusion. The metrics for the quality of tracks establish that two-thirds of the tracks fall under the category of MT. The ML metric is contributed mainly due to the sparsity of LiDAR at larger distances. The overall tracking quality is affected mainly due to the variation in the datasets with lowest quality resulting in dataset 1, 9, and 51. These datasets also contribute in higher FP and FN. The algorithm being sensitive to sudden change in speed and sharp turns that loses the track reflected in the IDSW count.

B. EXECUTION TIMES

The metric evaluations for KITTI datasets are carried out on desktop computer; however, time complexity is measured on the Jetson board. The time consumed by individual modules of the algorithm while executing in respective computational environment is presented in Table. 2. The overall performance of the algorithm in terms of metrics pertaining to accuracy, precision, and quality are comparable to state-of-the-art MODT paradigms. Furthermore, the computational

TABLE 2. MODT time consumption on desktop and jetson board.

Dataset	Objects	Ground Classifier (max. ms)	Clustering (max. ms)	Tracking (max. ms)	Visual Detector (max. ms)	Cycle Time (max. ms)
1	8	8.2	3.7	7.2	20.2	20.2
		37.7	18.2	20	80.6	80.6
2	2	7.7	3.7	2.6	19.7	19.7
		39.2	18.5	6.3	77.1	77.1
5	14	8	4.3	9.2	20.2	21.5
		37.2	19.2	28.3	83.3	84.6
9	60	8	4	12.2	24.1	**24.2**
		37.8	19.4	36.5	86.4	**93.7**
13	4	8.5	3.7	7	20	20
		41.6	17.5	15.8	80.3	80.3
17	2	7.7	3.3	1.9	19.8	19.8
		43.6	15.6	5.7	77.3	77.3
18	12	7.7	3.4	2.2	20.2	20.2
		37.7	14.8	7.7	76	76
48	5	7.6	4.1	7	19.9	19.9
		35.8	19.6	16.7	79.4	79.4
51	20	7.5	3.3	9.2	21.6	21.6
		43.4	15.6	30.4	84.9	89.4
57	20	7.7	3.3	5.5	20.7	20.7
		37.5	15	17.8	78.9	78.9

requirements adhere to demands of embedded systems, as overall execution cycle of the algorithm remains within the sampling time of LiDAR. Moreover, ample time is at disposal at every time step for communication across the platform.

The best performing parameters are used for evaluations while considering the constraints of computational resource and real-time requirements. The area of grid cell for ground classification, volume of grid-cell for clustering task, and the input image resolution are the key factors for optimization to realize real-time and resource constraint implementation of the proposed 3D object detection and tracking framework.

C. EVALUATION RESULTS FOR MOT

The extension in 2D MOT evaluation tool by KITTI for 3D MOT evaluation [19] include integral metrics to better express the performance of the frameworks. The purpose is to average the MOTA and MOTP at different threshold values for detection scores, like the existing approach of average precision for object detection [48]. Thus, AMOTA is defined as,

$$\mathrm{AMOTA} = \frac{1}{\mathrm{L}} \sum_{r\in\left\{\frac{1}{\mathrm{L}},\frac{2}{\mathrm{L}},\ldots,1\right\}} \left(1-\frac{\left[\sum_t (\mathrm{FN}_t+\mathrm{FP}_t+\mathrm{IDSW})\right]_r}{\sum_t \mathrm{G}_t}\right), \quad (21)$$

where L is the number of recall values set at 40, and all metrics are computed at r recall value. Furthermore, a scaled accuracy metric (sAMOTA) is used that provides the absolute measure of the system performance at a recall value r [19]. The tracking module in the proposed framework is tested on the validation set of KITTI tracking dataset, with an extended

TABLE 3. 3D MOT evaluation results.

Category	sAMOTA (%)	AMOTA (%)	AMOTP (%)	MOTA (%)	MOTP (%)	MT (%)	ML (%)	IDS	FRAG	FPS
Car	87.4	41.3	76.2	78.1	79.3	70.3	9.1	21	111	236
Pedestrian	53.1	15.8	41.5	46.1	67.6	30.3	38.7	57	500	242
Cyclist	85.1	41.3	77.2	70.9	77.6	71.4	14.3	3	24	229

TABLE 4. 3D MOT evaluation comparison.

Method	sAMOTA (%)	AMOTA (%)	AMOTP (%)	MOTA (%)	MOTP (%)	IDS	FRAG	FPS
FANTrack[20]	82.87	40.03	75.01	74.3	75.24	35	202	23.1
AB3DMOT [19]	**91.78**	44.26	77.41	83.35	78.43	**0**	**15**	207
mmMOT [22]	86.39	40.55	73.24	79.02	75.07	30	-	-
GTrkForecast [21]	92.37	**44.96**	**76.83**	**84.49**	78.32	3	-	-
3DT [17]	59.52	27.92	63.12	59.82	64.45	228	-	-
This work	87.42	41.31	76.15	78.12	**79.25**	21	111	**236**

evaluation tool. As KITTI dataset does not have an official train/validation split, following the approach in [19], [49] the sequences 1, 6, 8, 10, 12, 13, 14, 15, 16, 18, 19 are used for validation. Furthermore, a modification is made in the tracking module to comply with the input format of detector and tracking output readable for the evaluation tool.

The 3D MOT evaluation tabulated in Table. 3, depict better overall metric scores in car and cyclist categories compared to the pedestrian category. Furthermore, the lower scores in pedestrian category is also reflected in higher number of identity switches and fragmentation counts with comparatively lower percentage of MT percentage. This is mainly because of datasets with crowded pedestrians in proximity. The 2D MOT metrics show better weightages as tracks are evaluated on the image plane at a best performing recall threshold value. The speed of the tracker consistently remaining above 200 FPS without GPU utilization is an exceptional advantage as the framework is intended to run on embedded systems.

The recent 3D MOT methods on the KITTI leaderboards including FANTrack [20], Complexer-YOLO [15], DSM [16], and FaF [18] have not released the code yet to be evaluated on the new metrics. However, in AB3DMOT [19] reproduced results of FANTrack are evaluated and compared using the 3D MOT metrics. Similarly, mmMOT [22], GTrkForecast [21], and 3DT [17] are evaluated using the same evaluation tool, against common datasets to produce the corresponding metric results. The evaluation results of 3D MOT over validation KITTI-car dataset are tabulated in Table 4. For fair comparison all 3D MOT methods are provided with the object detections obtained by PointRCNN [45], so that only tracking performance of the methods are evaluated.

The comparison results suggest that the performance weightages lie close to state-of-the-art approaches, with a leading score of precision in 2D MOT format. Furthermore, the proposed method performs computation at the highest speed of 236 FPS. Similarly, the identity switching count ranks third among the compared approaches with a total of 21. Thus, the proposed tracker can perform on a budgeted computational resource, with performance metrics at par with the state-of-the-art approaches.

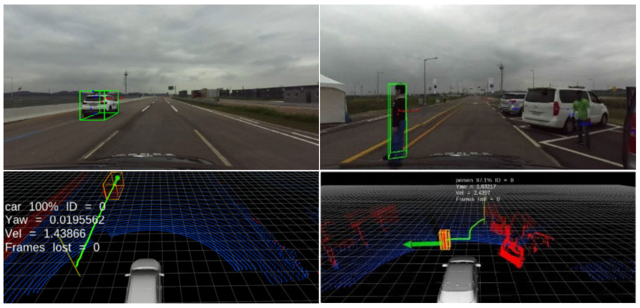


FIGURE 10. Detection and tracking of vehicle and pedestrian.

VII. EXPERIMENTS ON PLATFORM

The proposed algorithm is put to test under the scenario of fast emergency vehicle passing by and a model dummy being swiftly dragged across the front of platform as a pedestrian. Fig. 10. Shows that the framework efficiently tracks and reports the tracking parameters in both scenarios. The emergency vehicle is identified as a car with certainty of 100%, this implies that the tracked object in every time frame got assigned the same class, while maintaining a unique ID, and getting a measurement associated in the last time step.

The analysis of evaluation results against benchmarking datasets and experiments performed on the platform highlighted some failure cases and limitations of the proposed framework. The framework is set to capture and start tracking an object moving at a relative speed of less than 80 Km/h, the object entering the detection region with larger relative speed fails to establish a mature track. This results in ID switches and poor pose and tracking parameters estimation. Furthermore, the correct dimensions of a tracked object cannot be estimated that remains partially occluded throughout the track life, this reflects an error in estimated centroid.

The parameters of yaw angle (in radians) and speed of the emergency vehicle (in m/sec) are reported relative to the heading and speed of platform. Similarly, the trail of pedestrian track relative to platform reports the tracking parameters, as the platform stops at a distance to let the pedestrian vacate the path. Here, the class certainty is lower than 100% either because of missed visual detections or an incorrect class assignment in the track history.

VIII. ADDED POSSIBLE APPLICATIONS

The prime focus of this work is to perform multiple object detection and tracking. However, the proposed approach offers additional features that can benefit from the overall autonomous vehicle architecture. Such as, classification of static and dynamic regions in the scene can realize an informed selection of visual features for visual odometry. Similarly, the motion patterns of dynamic objects can benefit from the path planning. Furthermore, the tracking of static objects can aid the odometry, that is often required in dense urban environment where conventional localization methods become less reliable. Furthrmore, instance aware semantic segmentation of visual scene can also be realized in a cost-effective way, requiring the up sampling of tracked LiDAR clusters projected on image, as demonstrated in Fig. 11.

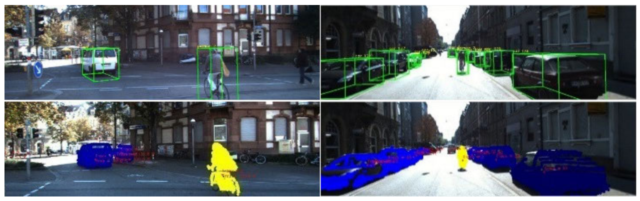


FIGURE 11. MODT based instance aware semantic segmentation.

IX. CONCLUSION

In this work an efficient MODT framework is proposed for embedded systems that operate on visual-LiDAR setups. The framework takes advantage of spatial LiDAR data and 2D scene understanding by performing late fusion of modalities temporally. The framework is tested on well-established performance metrics against publicly available synthetic KITTI datasets. Whereas, the tracking component is also independently tested for a 3D MOT for a fair comparison with state-of-the-art methods. It is intended to further extend this work in future to improve the object detection by early classification of objects and to realize MODT based semantic annotations of images. Moreover, MODT can be exploited to aid visual odometry in dense environmental conditions.

REFERENCES

[1] Car Crash Deaths and Rates—Injury Facts. *Car Crash Deaths and Rates*. Accessed: Apr. 29, 2020. [Online]. Available: https://injuryfacts.nsc.org/motor-vehicle/historical-fatality-trends/deaths-and-rates/

[2] M. Cunneen, M. Mullins, F. Murphy, D. Shannon, I. Furxhi, and C. Ryan, "Autonomous vehicles and avoiding the trolley (Dilemma): Vehicle perception, classification, and the challenges of framing decision ethics," *Cybern. Syst.*, vol. 51, no. 1, pp. 59–80, Jan. 2020.

[3] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. 23rd Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, Mar. 2018, pp. 751–766.

[4] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An efficient probabilistic 3D object detector for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12677–12686.

[5] J. Zhou, X. Tan, Z. Shao, and L. Ma, "FVNet: 3D front-view proposal generation for real-time object detection from point clouds," 2019, *arXiv:1903.10750*. [Online]. Available: http://arxiv.org/abs/1903.10750

[6] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018.

[7] G. Brazil and X. Liu, "M3D-RPN: Monocular 3D region proposal network for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9287–9296.

[8] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3D object detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 641–656.

[9] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7345–7353.

[10] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.

[11] Z. Wang and K. Jia, "Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 1742–1749.

[12] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3D object detection methods for autonomous driving applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019.

[13] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–8.

[14] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6526–6534.

[15] M. Simon, K. Amende, A. Kraus, J. Honer, T. Samann, H. Kaulbersch, S. Milz, and H. M. Gross, "Complexer-YOLO: Real-time 3D object detection and tracking on semantic point clouds," in *Proc. CVPRW*, Jun. 2019, pp. 1–10.

[16] D. Frossard and R. Urtasun, "End-to-end learning of multi-sensor 3D tracking by detection," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018, pp. 635–642.

[17] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Kraehenbuehl, T. Darrell, and F. Yu, "Joint monocular 3D vehicle detection and tracking," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5390–5399.

[18] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3569–3577.

[19] X. Weng, J. Wang, D. Held, and K. Kitani, "3D multi-object tracking: A baseline and new evaluation metrics," 2019, *arXiv:1907.03961*. [Online]. Available: http://arxiv.org/abs/1907.03961

[20] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki, "FANTrack: 3D multi-object tracking with feature association network," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1426–1433.

[21] X. Weng, Y. Yuan, and K. Kitani, "Joint 3D tracking and forecasting with graph neural network and diversity sampling," 2020, *arXiv:2003.07847*. [Online]. Available: http://arxiv.org/abs/2003.07847

[22] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy, "Robust multi-modality multi-object tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 2365–2374.

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

[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[25] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," 2019, *arXiv:1906.05113*. [Online]. Available: http://arxiv.org/abs/1906.05113

[26] P. Narksri, E. Takeuchi, Y. Ninomiya, Y. Morales, N. Akai, and N. Kawaguchi, "A slope-robust cascaded ground segmentation in 3D point cloud for autonomous vehicles," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 497–504.

[27] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3D point clouds for ground vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2010, pp. 560–565.

[28] Q. Li, L. Zhang, Q. Mao, Q. Zou, P. Zhang, S. Feng, and W. Ochieng, "Motion field estimation for a dynamic scene using a 3D LiDAR," *Sensors*, vol. 14, no. 9, pp. 16672–16691, 2014.

[29] M. Velas, M. Spanel, M. Hradis, and A. Herout, "CNN for very fast ground segmentation in Velodyne LiDAR data," in *Proc. IEEE Int. Conf. Auton. Robot Syst. Competitions*, Apr. 2018, pp. 97–103.

[30] S. K. Uppada, "Centroid based clustering algorithms—A clarion study," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 6, pp. 7309–7313, 2014.

[31] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, Jun. 2014.

[32] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Proc. 14th IEEE Int. Conf. Data Eng.*, Feb. 1998, pp. 324–331.

[33] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, Oct. 2018.

[34] D. D. Morris, R. Hoffman, and P. Haley, "A view-dependent adaptive matching filter for LiDAR-based vehicle tracking," in *Proc. 14th IASTED Int. Conf. Robot. Appl.*, Cambridge, MA, USA, Nov. 2009, pp. 1–9.

[35] Z. Luo, S. Habibi, and M. V. Mohrenschildt, "LiDAR based real time multiple vehicle detection and tracking," *Int. J. Comput. Electr. Autom. Control Inf. Eng.*, vol. 10, no. 6, pp. 1125–1132, 2016.

[36] Y. Wu, J. Lim, and M. H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[37] B. N. Vo and W. K. Ma, "The Gaussian mixture probability hypothesis density filter," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4091–4104, Nov. 2006.

[38] S. H. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. Dick, and I. Reid, "Joint probabilistic data association revisited," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3047–3055.

[39] A. Doucet, "On sequential simulation-based methods for Bayesian filtering," Dept. Eng., Univ. Cambridge, Cambridge, U.K., Tech. Rep. CUED-F-ENG-TR310, 1998.

[40] M. Schreier, V. Willert, and J. Adamy, "Compact representation of dynamic driving environments for ADAS by parametric free space and dynamic object maps," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 367–384, Feb. 2016.

[41] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," 2016, *arXiv:1603.00831*. [Online]. Available: http://arxiv.org/abs/1603.00831

[42] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," Apr. 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[43] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common objects in context," May 2014, *arXiv:1405.0312*. [Online]. Available: http://arxiv.org/abs/1405.0312

[44] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[45] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," Dec. 2018, *arXiv:1812.04244*. [Online]. Available: http://arxiv.org/abs/1812.04244

[46] X. Weng and K. Kitani, "Monocular 3D object detection with pseudo-LiDAR point cloud," 2019, *arXiv:1903.09847*. [Online]. Available: http://arxiv.org/abs/1903.09847

[47] J. Leonard *et al.*, "A perception-driven autonomous urban vehicle," *J. Field Robot.*, vol. 25, no. 10, pp. 727–774, Oct. 2008.

[48] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2009.

[49] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granstrom, "Mono-camera 3D multi-object tracking using deep learning detections and PMBM filtering," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Changshu, China, Jun. 2018, pp. 433–440.

MUHAMMAD SUALEH received the B.S. degree in electronics engineering from COMSATS University Islamabad, Abbottabad Campus, Pakistan, in 2009, and the M.S. degree in systems, control, and mechatronics from the Chalmers University of Technology, Sweden, in 2011. He is currently pursuing the Ph.D. degree with the Department of Control and Robot Engineering, Chungbuk National University, South Korea.

His research interests include robotics, semantic SLAM, object detection and tracking, and control systems.

GON-WOO KIM received the M.S. and Ph.D. degrees from Seoul National University, South Korea, in 2002 and 2006, respectively.

He is currently a Professor with the School of Electronics Engineering, Chungbuk National University, South Korea. His research interests include navigation, localization, and SLAM for mobile robots and autonomous vehicles.