Analysis of multiple modality object detectors and 3D multiple-object tracking

David Matos Rodriguez (ID #1352822)
MSc Automotive Technology, Signal Processing Systems
November 22, 2020

***Abstract*—3D Multiple-object-tracking (MOT) is at the core of the operation of autonomous vehicles. The approach that has dominated recently is the tracking-by-detection approach, where detection of objects is followed by temporal association over multiple frames. In this paper, we analyze the role that different object detectors have on the tracking module's performance for different sensor modalities, such as Camera, LiDAR or Camera-Lidar sensor fusion. This paper shows that consistently LiDAR based detectors outperform camera or sensor fusion based detectors across every metric on the KITTI MOT benchmark. Particularly, LiDAR based, PointRCNN outperforms sensor-fusion based, FrustumPointNet by 12.18 and 8.81 percentage points when used in the baseline tracker in the AMOTA and MOTA metrics respectively for the car class. Additionally, both FrustumPointNet and Pseudo-LiDAR FrustumPointNet are unable to detect any ground truth object in the 56-84m range of the KITTI dataset. Furthermore, due to the key role that data association plays in tracking, two different affinity measures are compared: 3D-IoU and Mahalanobis distance. Their role in tracking is evaluated on the three classes of the KITTI MOT benchmark. By changing the affinity measure in the data association module of tracking to mahalanobis distance, the cyclist MOTA increases by 21 percentage points, while maintaining similar performance in the pedestrian class. Finally, PointRCNN detections with 3D-IoU affinity measure MOT tracking algorithm is implemented with ROS operating system in the TU/e - AIIM autonomous vehicle, and its qualitative results are visualized on both the KITTI dataset and on data generated by the LiDAR sensors on the TU/e - AIIM autonomous vehicle.**

Fig. 1: Sensor setup of the TU/e - AIIM autonomous vehicle. Setup includes an array of LiDAR sensors, and a mono, stereo camera setup. Focus is on the 360° LiDAR (on top of the vehicle) and the mono and stereo cameras (behind front windscreen). The sensor data is logged to ROS as well as the implementation of the MOT algorithm, described in this work.

I. Introduction

AUTONOMOUS vehicles have become more prominent in our daily lives. They could potentially reduce the numbers of accidents, specially due to the number of cases where humans are found to be the main source of error [1], and increase productivity by reallocating driving time [2]. However, autonomous vehicles have to operate in dynamic and unstructured environments, where the room for error is minimal. The vehicle needs to understand its surroundings, and make decisions to avoid colliding with any moving objects around the vehicle. Perceiving the environment entails understanding how objects are moving around the vehicle and understanding their future behavior using sensors such as cameras and LiDARs, where a detailed description of the environment including the detection, and tracking of multiple objects (multiple - object tracking) becomes of importance.

Ultimately, the purpose of an automated vehicle is to transport people, thus safety and redundancy are important. The EU funded project Prystine aims to research and test fail-operational safe environment perception, which is achieved by exploiting sensor redundancies in the coverage of the environment in space and time. Within Prystine project, a smaller consortium of partners is focusing on traffic prediction, in highway and urban driving scenarios. For these scenarios, TU/e is focusing on the development of LIDAR and camera sensor fusion technology, mainly on vehicle, pedestrian and cyclist detection and tracking. The current setup for this vehicle is shown in Figure 1 and is used in this work.

As shown in Table I, camera and LiDAR use different technologies to enable the autonomous vehicle to detect the objects in its surroundings. Depending on the application one sensor may be preferred over the other. For instance, even though both LiDAR and Cameras are affected by weather conditions, LiDAR is more robust to rain [3] than cameras, thus it can detect objects more reliably in this condition. Additionally, LiDAR is not affected by changes in illumination, therefore it is still able to detect objects at night which camera is not capable of. However, some of the LiDAR benefits come at a cost of extremely costly sensors compared to cameras.

Modality	Affected by weather	Affected by illumination	Depth	Range	Cost	Resolution
Mono Camera	✓✓	✓	-	-	✓	✓✓
LiDAR	✓	-	✓	✓✓	✓✓✓	✓
Stereo Camera	✓✓	✓	✓	✓	✓✓	✓✓

TABLE I: Comparison of different sensor modalities for different categories. Extra tick symbols indicating higher effect/quantity. Dash indicating incapability of sensor to perform task. Inspired by comparison in [4].

For this research, the aim is to evaluate multiple modality sensors, and examine their behavior in the context of multiple-object tracking performance in the aforementioned scenarios. Additionally, we go a step deeper and examine possible improvements to the tracking aspect of environment perception as well. For instance, it would be of interest to find out what happens when a particular sensor fails, or to find out the vehicle's ability to track other objects than cars, such as pedestrians and cyclist given the urban scenarios.

Specifically, the multiple-object tracking approach followed is tracking-by-detection. In this approach, whose methodology is shown in Figure 2, LiDAR and camera sensors in different modalities are used as input to various deep learning based object detectors [5] that output perception information about objects for every frame/image, specifically 3D bounding boxes. Consequently, perception information is passed on to a tracking module, which estimates the trajectory of objects in the next frames. Further, perception information and trajectories need to be associated to identify the relation between the two in order to determine how the bounding boxes are related through time, also known as data association. Finally, the trajectories are deleted or created based on the use of different hyperparameters in the track management submodule.

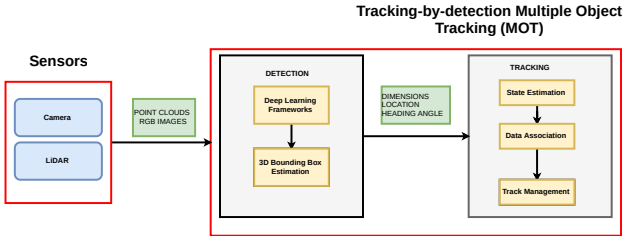


Fig. 2: Multiple Object Tracking methodology

This research focuses on the following contributions:

- Evaluate the performance of different object detectors that use different sensor modalities, specifically camera and LiDAR, in the context of its use within the MOT framework.
- Improve the tracker performance on cars, pedestrians and cyclists
- Implement the algorithm in the TU/e - AIIM autonomous vehicle.

Section II describes related work and literature on detection and tracking. Section III and Section IV describe the multiple-object tracking architecture with its individual modules explained, along with a description of the metrics and datasets used in the research, and subsequently a description for the ROS implementation. Section V evaluates the performance of the individual detectors, as well as the improvements made to the tracking module, along with qualitative results obtained with ROS. Section VI summarizes the results obtained as well as recommendations for further research.

II. Related Works

A. Detection

2D detection was initially performed using hand-crafted machine learning features such as Haar-like features and AdaBoost. More recently, detection in vision has been performed by deep learning based methods, which use Convolutional Neural Networks to learn object features. Most prominently, 2D detectors that rely on camera, include Faster-RCNN [6], Single Shot Detector [7] and YoloV3 [8]. Extending to 3D object detection, PointNet [9], proposed one of the first methods for working with unordered set of point clouds. VoxelNet [10] applies PointNet to voxels. FrustumPointNet [11] fuses LiDAR with camera to generate RGB-D data. Additionally, AVOD [12] fuses LiDAR bird's eye view with camera. PointPillars [13] and PointRCNN [5] use LiDAR to generate 3D bounding boxes.

Additionally, 3D object detection can be obtained by replacing LiDAR with Pseudo - LiDAR [14, 15]. In [14], stereo depth estimation is obtained via Pyramid Stereo Matching Network(PSMNet) [16] and DORN [17] for monocular depth estimation.

B. Multiple Object Tracking

Earlier in computer vision, tracking involved following points of interest through space and time [18], such as the motion of pixels from frame to frame in order to determine their displacement. Even though some algorithms could be simple and robust, they were quite limited in the absence of strong level cues. With the recent evolution of deep learning, a different approach has surfaced, which is tracking-by-detection [19–21], which relies on object detectors to identify potential objects, and matching them through time on a different stage.

Traditionally, tracking has been performed using Multiple Hypothesis tracking (MHT) [22], yet MHT requires high computational cost which makes it unsuitable for real time applications. Additionally, tracking has also been performed with Joint Probabilistic Data Association Filter (JPDA) [23] and Global Nearest Neighbor (GNN). Recently, Simple Online Realtime Tracking (SORT) [24] proposes a tracking-by-detection tracker with high frame rate with the focus of associating objects efficiently for online and realtime applications with the use of the Hungarian Algorithm. Further, DEEPSORT [21] expands on SORT by integrating appearance information via a deep association metric. Moreover, AB3DMOT [20] expands on SORT by extending the state space of the filter defining bounding boxes from 2D to the 3D space. More recently, [25] expands on AB3DMOT by using a Greedy Algorithm rather than Hungarian algorithm. Tracking-by-detection, or online trackers tend to be the most popular as opposed to batch tracking, due to the delay of assigning a detection to a track until further information is available. Regarding motion

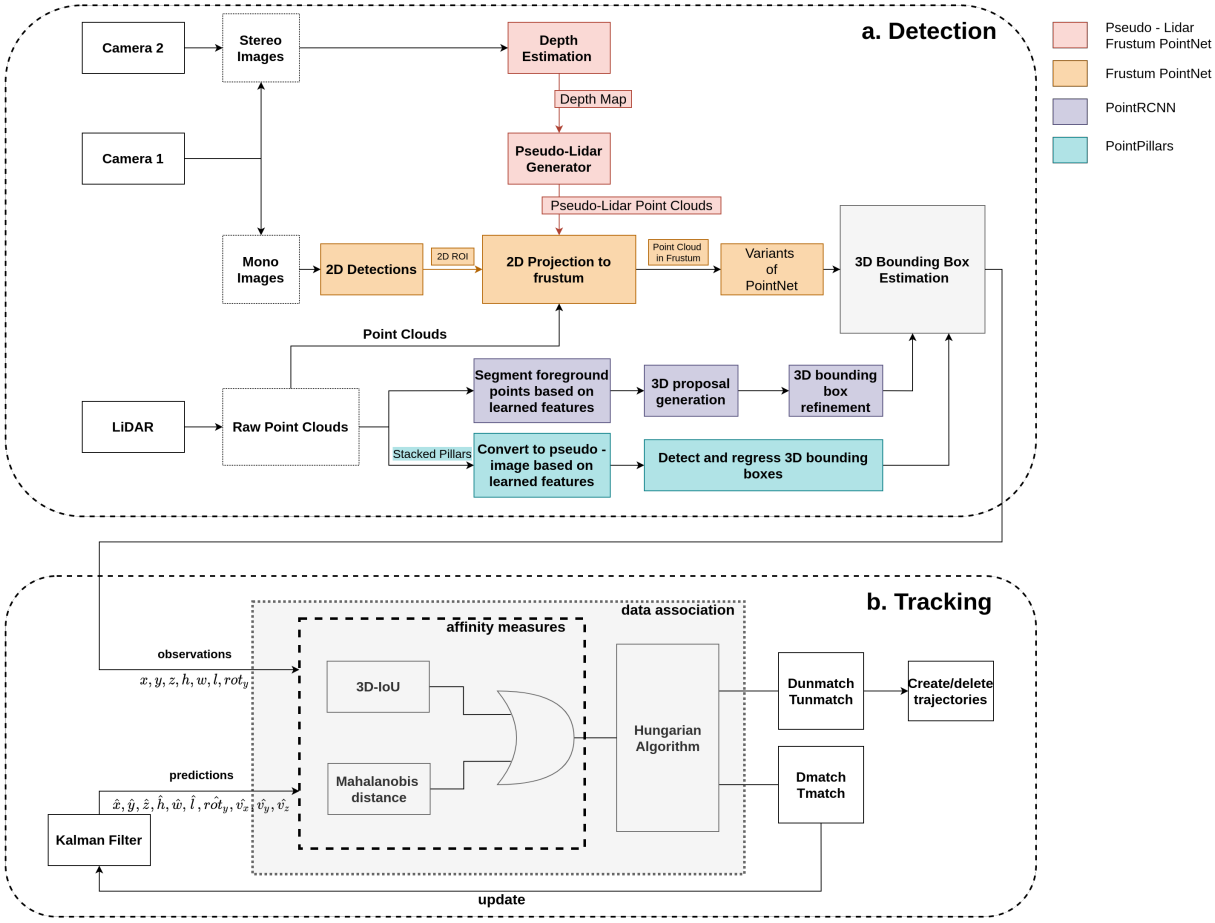


Fig. 3: Possible architecture for sensor fusion detection and tracking algorithms. In this research we evaluate the detectors separately and propose an improvement for the tracker.

prediction, aside from Kalman Filter, [26], uses an optical flow estimation network to update joint locations in human poses.

Regarding affinity measures for data association, SORT [24] uses 2D-IoU as affinity measure for data association, and AB3DMOT [20] extends it to 3D-IoU. Further, [25] and DeepSORT [21] use mahalanobis distance as affinity measure instead.

III. Methods

In this section, we describe the multiple-object tracking method followed, where a high-level architecture of the different sensor modality object detectors, and tracking architecture used are described.

Thanks to the nature of the tracking-by-detection approach, its possible to make the multiple-object-tracking task modular, in which detection and tracking act as separate modules, and tweaked independently. Keeping this in mind, multiple-modality detectors and different affinity measures for data association methods are investigated in order to further understand how deeply they are interconnected.

A. Detection module

As part of the tracking-by-detection approach, object detectors are used as input to the tracking module. Each individual detector is evaluated so that the performance of the tracker based on the different sensor modalities can be assessed. Each individual detector is described below.

As shown in Figure 3, the detection module is the first processing step in the MOT algorithm. Detectors based on different sensor modalities are used in order to understand how each input modality affects the tracking performance.

1) PointRCNN: Performs 3D Object detection from raw LiDAR point clouds. It obtains the raw point clouds and processes them in two stages: a first stage for proposing 3D proposals and a second stage for refining the 3D proposals, which are composed of 3D bounding boxes (see Figure 4). The proposed 3D bounding boxes are obtained by learning point-wise features with PointNet++[27] which is then used to segment the foreground point clouds. These segmented foreground point clouds are used to extract further information and obtain the refined 3D bounding box estimation.

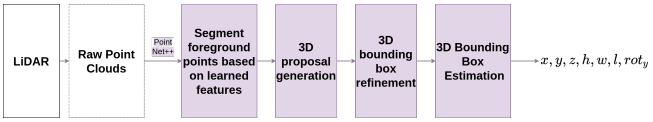


Fig. 4: PointRCNN architecture

2) PointPillars: Performs 3D Object detection from raw LiDAR point clouds but instead of encoding raw point clouds, it organizes the point clouds to vertical columns or pillars (see Figure 5). The encoded pillars are used to learn features that are then converted to a 2D pseudo-image which acts as input to a 2D convolutional neural network. Subsequently, SSD [7] is used as the detection head which regresses and detects 3D bounding boxes in a single stage.

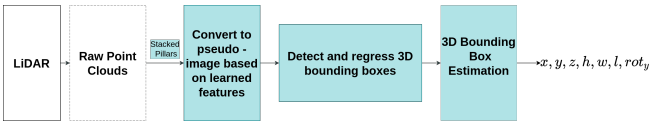


Fig. 5: PointPillars architecture

3) FrustumPointNet: Performs 3D object detection from mono RGB images + LiDAR data. It does so by obtaining robust 2D bounding boxes from a robust 2D object detector. Each of these bounding boxes are extruded to a 3D viewing frustum, in which a point cloud is obtained from depth data, obtained by a LiDAR sensor (see Figure 6). The point cloud in the frustum data is processed by a variant of PointNet[9], where the higher representation of point clouds is used to estimate a 3D bounding box.

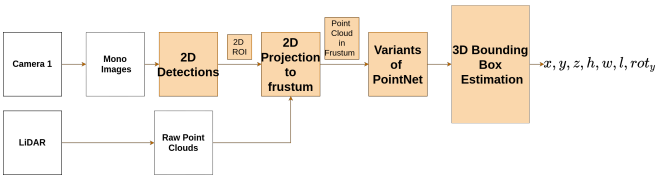


Fig. 6: Frustum PointNet architecture

4) Pseudo - LiDAR FrustumPointNet: Performs 3D object detection using the FrustumPointNet architecture, but in contrast it replaces the real LiDAR point clouds used in FrustumPointNet with a Pseudo - LiDAR point cloud (see Figure 7).

Pseudo - LiDAR [14], enables 3D object detection by converting image-based depth maps to pseudo - LiDAR representation, enabling this new data representation to be injected into any 3D object detector that uses LiDAR, such as AVOD [12] or FrustumPointNet [11]. In this work, FrustumPointNet is selected to be used with Pseudo - LiDAR so that it can enable a comparison of two identical architectures but with different sensor data. Please refer to Appendix A for extra information about equations used to generate the pseudo - LiDAR and its architecture.

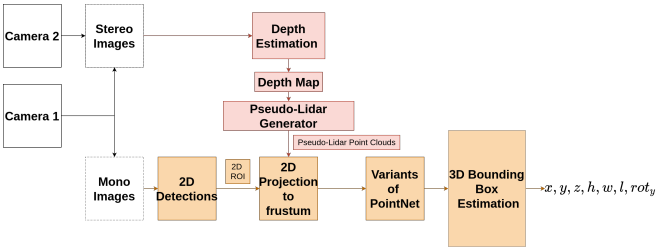


Fig. 7: Pseudo - Lidar FrustumPointNet architecture

Each of the individual detectors output 3D bounding boxes for objects found for every frame in the dataset. In this way, each detector can be used as input to the tracking module and evaluated accordingly. Additionally, each detector can be evaluated independently of the tracker and establish a detector performance based on a precision-recall curve, which will be shown in Section V-A.

B. Tracking module

The tracking module takes as input the output of the detection module, which consists of the objects' 3D bounding boxes, as shown in Figure 2. As this is a purely an online method, or tracking-by-detection method, only the detections at the current frame and the associated trajectories from the previous frame are needed. Online tracking methods only have available present and past data, but no future data. For current frame t, the tracking module receives detections $D_t = \{D_t^1, D_t^2, D_t^{n_t}\}$, where n_t indicates the total number of detections in that frame. Each detection D_t^j, with $j \in \{1, 2, ..., n_t\}$, is parametrized as set of seven parameters (x, y, z, h, w, l, roty), which indicate the 3D object center (x, y, z), the box dimension(h, w, l), orientation angle of the box (roty). Given that the detections are used as measurements or observations for the Kalman Filter [28], the total of observations is, $z = 7$.

1) Kalman Filter Prediction: For the prediction step, the displacement of the bounding boxes between frames is calculated by using a constant velocity motion model. At every frame, the state of associated trajectories from the previous frame $T_{t-1} = \{T_{t-1}^1, T_{t-1}^2, ..., T_{t-1}^{m_{t-1}}\}$, where m_{t-1} is the number of trajectories in the previous frame, are propagated to the current frame to get the estimated trajectories T_{est} for the current frame. Consequently, each trajectory T_{t-1}^i in the previous frame, with $i \in \{1, 2, ..., m_{t-1}\}$, ends up with trajectories for the current frame as a 10-dimensional vector $T_{est}^i = (\hat{x}, \hat{y}, \hat{z}, \hat{h}, \hat{w}, \hat{l}, \hat{rot}_y, \hat{v_x}, \hat{v_y}, \hat{v_z})$, which is sent to the data association module, as shown in Figure 3.

2) Data Association: In order to determine how each D_t and T_{est} are related, the Hungarian algorithm is used. The Hungarian algorithm uses as input an assignment affinity matrix which is used to determine which detection to match with a predicted object state. The way in which the affinity matrix is built, in turn affects how the Hungarian algorithm is able to optimally solve the assignment problem. As such, the affinity matrix represents an important role in the data association module. For our baseline, the affinity matrix is computed as the 3D-IoU between each detection and target, with dimensions

$n_t \times m_{t-1}$, indicating how much the prediction and observation are overlapping in 3D space.

IoU indicates how much two objects are related. 3D-IoU is defined in equation (2), where V_i is the intersection volume between the objects and V_1 and V_2 indicate the volume for each object individually. V_i is defined in equation (1), where A_i indicates the area of intersection between the two objects, and y indicates the height difference between the objects.

$$V_i = A_i * max(0, y_{max} - y_{min}) \tag{1}$$

$$IoU_{3D} = \frac{V_i}{V_1 + V_2 - V_i} \tag{2}$$

If the matching distance is below a certain threshold, IOU_{min}, then the match is rejected. The output of the data association module is the matched and unmatched detections D_{match}, $D_{unmatch}$ and trajectories T_{match}, $D_{unmatch}$ as shown in Figure 3.

3) Kalman Filter Update: Following the output of the data association module, the state of the matched trajectories T_{match} are updated based on the corresponding pair in D_{match}. Consequently, the trajectories associated for the frame t, are defined as $T_t = (T_t^1, T_t^2, ..., T_t^{w_t}\}$, where w_t indicates the number of matches. Further, the updated state of each trajectory $T_t^k = (x', y', z', h', w', l', rot_y', v_x', v_y', v_z')$ with $k \in \{1, 2, ..., w_t\}$ is the weighted average (gain) of T_{match} and D_{match} per Kalman Filter procedure.

C. Mahalanobis Distance implementation

For the tracking module, there are key high level sub modules such as state estimation, data association and track management, as shown in Figure 2. For this research, the focus is on data association module as its considered one of the most challenging stages of vehicle detection and tracking due to the numbers of ambiguities that arise [29]. Furthermore, given the simplicity of the tracking algorithms, the aim is for the different modalities of input detections to handle the edge cases in its different ways and the tracker to focus on frame-frame associations. Hence, the Kalman Filter [28] is kept along with the Hungarian [30] algorithm as the baseline components. Therefore, as shown in Figure 3, besides the 3D-IoU affinity measure used as the baseline, the mahalanobis distance affinity measure is added to determine the possible effects it has on the data association module and the overall tracker performance. The mahalanobis distance is calculated with equation (5).

Given a 10-dimensional vector ($x = 10$) for the state trajectories, and a 7-dimensional vector parametrizing the bounding boxes observations from the detector ($z = 7$), the mahalanobis distance calculates the probabilistic distance between the predicted detections from the Kalman Filter T_{est} and the input detections from the detector D_t. As opposed to a 3D-IoU, the mahalanobis distance takes into account the uncertainties of the prediction, through the innovation covariance in (4), which projects the system uncertainty into measurement space, where H denotes the measurement function, $H_{z \times x}$ = [I 0], where I is the identity matrix. Additionally, P indicates the state covariance matrix and is calculated with equation (3), where Q, R are the process and observation noise respectively. Furthermore, the selection of P, Q, R values affect the convergence of the Kalman Filter. The mahalanobis distance indicates roughly how many standard deviations a certain value is away from the mean of a distribution. Finally, in the same way the 3D-IoU baseline uses an IoU_{min} threshold, a mahalanobis threshold of 0.80 is also used to reject matches. To further understand how mahalanobis distance is used, please refer to Appendix B for an example.

$$P_t = FP_{t-1}F^\top + Q \tag{3}$$

$$S_t = HP_tH^\top + R \tag{4}$$

$$d_m = \sqrt{(D_t^j - T_{est}^i)^\top \times S_t^{-1} \times (D_t^j - T_{est}^i)} \tag{5}$$

In order to understand the algorithm's ability to detect objects, and consequently the sensor modalities effect on the tracker, a False Negative analysis is performed on the tracking results obtained from the MOT algorithm. Specifically, an analysis on the missed distance by the sensor to an object from the ego-vehicle, as it will be shown in Section V-B. This extracted information can give further indication on the distance limitations of an specific sensor or algorithm, in which case another sensor can be used as backup.

D. ROS implementation

On top of performing quantitative analysis for detections in different modalities, and performing analysis in the data association module of tracking, the 3D-IoU affinity measure with PointRCNN is selected to perform a qualitative analysis. The MOT tracking algorithm implemented in ROS is divided in two main nodes: a detection node which publishes the 3D bounding boxes, and a tracking node which subscribes to the bounding boxes, and publishes the objects' trajectories. KITTI raw data, along with data generated from the sensors on the TU/e - AIIM autonomous vehicle are used to visualize the MOT algorithm implementation running at 10Hz.

A ROS bag is used as the tool to play the incoming data from either the KITTI raw data or the vehicle's data. A ROS bag enables the sensor data to be stored as incoming messages, allowing us to read and visualize a variety of topics.

IV. Experiments

A. Dataset

The KITTI MOT [31] dataset consists of 21 training and 29 test sequences, totaling 6 hours of different tracking scenarios, ranging from freeways over rural areas to innercity scenes around Karlsruhe, Germany. The KITTI dataset includes camera, LiDAR, and GPS/IMU inertial navigation system. The color camera is a PointGray Flea2, 1.4M pixels, 1/2" Sony ICX267 CCD, and the LiDAR is a Velodyne HDL-64E rotating laser scanner, 10hz, 64 beams, collecting approximately 1.3 million points per second. It consists of 8 different classes including car, pedestrian, cyclist, tram, misc, van, truck, person sitting). However, only the classes Car, Pedestrian and Cyclist(optional) are used for evaluation with the KITTI

MOT benchmark due to the amount of available instances. Additionally, the KITTI dataset already provides calibration, which include projection matrices for each sequence to go from one set of coordinates to another. Additionally, the KITTI object detection dataset consists of 7481 training images and 7518 testing images, comprising a total of 80,256 labeled objects.

Finally, the KITTI dataset is used for this research given the fact that majority of literature and object detectors are benchmarked and available only for KITTI dataset as the moment of this writing. KITTI is therefore at the moment the standard for automotive applications. Other datasets were found as well in a later stage of the research (such as NuScenes), but it was decided to keep KITTI for the aforementioned reasons.

B. Object detector training

Following the fact that every object detector used in this research involves deep learning, there is a need to train each individual detector mentioned in Section III-A. Fortunately, the authors of the individual papers released pre-trained models which were trained on the object detection dataset, except for PointRCNN which was not yet trained on the pedestrian and cyclist classes. In order for detectors to be useful for tracking, these pre-trained models have to be inferred on the MOT dataset sequences instead, so that the detection results for the car class for each detector can be generated successfully. Finally, following convention, we follow [32] and use the sets 1,6,8,10,12,13,14,15,16,18,19 of the training set as the 11 sets used for the validation set.

Since there are no results readily available for pedestrians and cyclists, PointRCNN [5] is trained for the pedestrian and cyclist classes as well, which are needed for the data association analysis in the tracking module. For complete details about the original network, refer to the original paper [5], and to Figure 4for high level architecture.

In order to modify PointRCNN to train pedestrians and cyclists, the following changes are made to the two-stage network. The class mean size defined as height, width, length is changed from (1.5, 1.6, 3.9) used for car, to (1.73, 0.6, 0.8) used for pedestrian and cyclist. Additionally, the x,y,z scope of the point clouds is changed from [[-40, 40], [-1, 3], [0, 70.4]] meters for cars to [[-20, 20], [-0.5, 2.5], [0, 48]] meters for pedestrian and cyclist in rectified camera coordinates, as suggested by PointPillars [13].

Once the 3D proposals are generated, the training of the second network in charge of refining the 3D proposals, considers a proposal positive if its maximum 3D-IoU with ground truth boxes is above 0.6, and considers it negative if its maximum 3D-IoU is below 0.45 for cars. However, for pedestrian and cyclist the positive and negative thresholds are changed to 0.50 and 0.35 respectively.

C. Metrics

The tracking performance is evaluated using the KITTI MOT benchmark. The objective of the KITTI MOT benchmark is to estimate object tracks for the car, pedestrian and cyclist(optional) classes. The KITTI MOT benchmark follows the CLEAR [33] metrics as evaluation criterion.

The metrics used for this work are:

- AMOTA : Average Multiple Object Tracking Accuracy.
- MOTA: Multiple Object Tracking Accuracy.
- FP : number of false detections.
- FN : number of missed detections.
- IDS : number of times an ID is switched.
- FRAGS: number of times that a trajectory has been fragmented/interrupted during tracking.

The primary metric for measuring performance is the Multiple-Object-Tracking Accuracy (MOTA) defined in equation (6) and AMOTA defined in equation (7), where GT indicates the total number of ground truth objects.

As a matching criterion, the KITTI MOT benchmark uses the 2D bird's eye-view IoU of 0.50, meaning if the IOU of the prediction and the target is equal or greater than 0.50, then it is considered a true positive (match), otherwise it is considered a false positive.

By default, the KITTI MOT benchmark is only capable of evaluating 2D systems and considers every track to have the same confidence score, which means that every track output by the tracker is evaluated regardless of the confidence it has about that particular object, thus accounting in many cases for many low confidence score false positives, and more importantly only evaluating the model in a single operating point. Consequently, a common approach for submitting results in the KITTI MOT benchmark is to manually eliminate tracks iteratively based on the confidence score, trying to find the right balance between false positives and false negatives.

Besides evaluating with the convential CLEAR metrics, AB3DMOT [20] proposes a new metric in order to address the issue of single point evaluation of a model. AB3DMOT proposes AMOTA(Average Multiple Object Tracking Accuracy), which integrates MOTA over specific length of recall values. The integration is approximated with a discrete set summation over recall values. AMOTA is defined in equation (7), where L is the length of recall values, L = 40, and subscript r indicates the value at that recall.

$$MOTA = 1 - \frac{FP + FN + ID_s}{GT} \tag{6}$$

$$AMOTA = \frac{1}{L} \sum_{r \in \{\frac{1}{L}, \frac{2}{L}, \ldots, 1\}} 1 - \frac{FP_r + FN_r + ID_r}{GT} \tag{7}$$

Additionally, in order to address the issue of only evaluating in 2D space, the matching criterion is extended to 3D space by using 3D-IoU as a matching criterion rather than 2D bird's eye view. In order to be considered a match, the 3D-IoU has to be greater than 0.25. In conclusion, evaluation is performed on the CLEAR metrics extended to 3D-IoU and also evaluated on the AMOTA metric as suggested by AB3DMOT.

V. Results

A. Detector comparison

In order to first understand how each individual detector is performing, a precision-recall curve is plotted. Based on

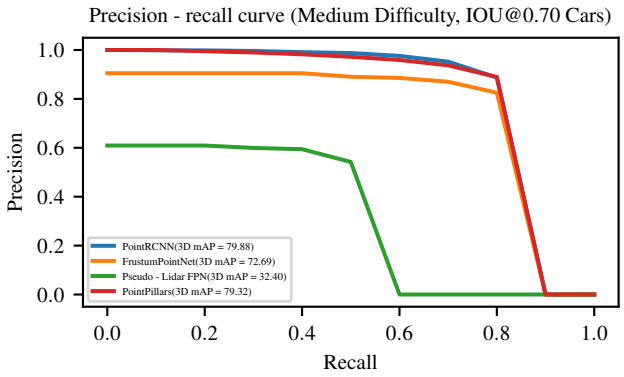
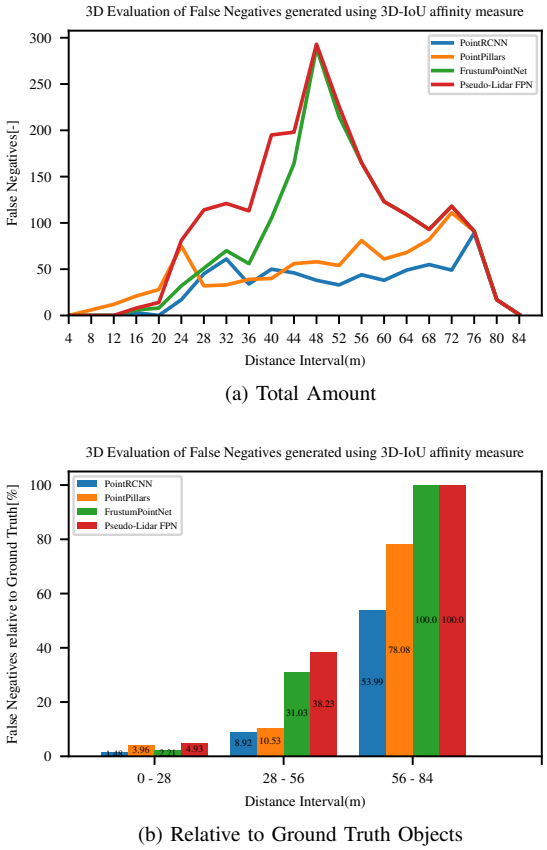


Fig. 8: Precision - recall curve for detectors on the KITTI validation set. Evaluation of the CAR class, based on the 'Moderate' difficulty at IOU of 0.70 from the KITTI benchmark.

the results from Figure 8, it can be seen that at moderate difficulty, the highest mAP are PointRCNN and PointPillars, both LiDAR based. Furthermore, the aim is to use each individual detector with the highest mAP possible as an input to the tracker, which means using detections without any confidence threshold. If the input detections are thresholded, the chances of missing an object(false negative) increase, which causes the detector to have an overall lower recall value. Depending on the application, one may favor more false negatives or positives, however in the automotive industry it is crucial that the model misses as few objects as possible. Although by using no confidence threshold, the total of false positives is higher, its left to the tracker to deal with noisier input detections.



(a) Total Amount

(b) Relative to Ground Truth Objects

Fig. 9: 3D False Negative Evaluation of CAR class on the KITTI validation set. Evaluation indicating the total amount of false negatives found when using the 3D-IoU affinity measure in the tracking module (a) and the percentage of false negatives found per distance interval relative to the total amount of ground truth car objects in the validation set (b).

B. Tracking results

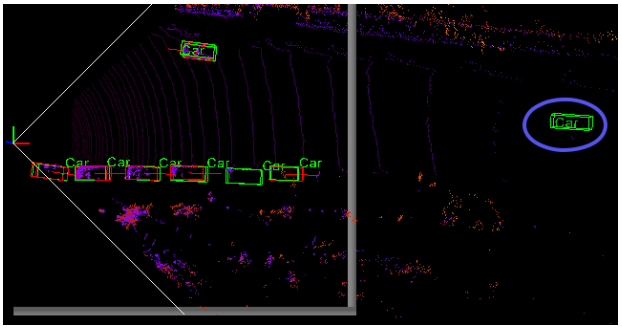
Once the tracking results are generated using different detectors, an analysis is performed indicating the ground truth distance of the ego - vehicle to a particular object which the detector misses. As seen in Figure 9b, in the range distance between 56-84m, FrustumPointNet misses every ground truth object, as well as Pseudo - LiDAR. This could be due to the fact of the FrustumPointNet architecture which relies on 2D detections in order to project point cloud in the frustum, indicating the limitation of camera for detecting far away objects. Additionally, as seen in Figure 9a, the lowest number of misses is by the LiDAR based solutions across the distance spectrum. Qualitatively, Figure 10 shows a specific sequence where FrustumPointNet is unable to detect a vehicle 64.8m from the ego-vehicle, but PointRCNN is able to do so.

As highlighted in Table II, the highest AMOTA and MOTA results are for the tracker that uses PointRCNN detections, which brings the best evaluating MOTA to 85.96 %, compared to the highest fusion-based result, 77.15 %. Further, in terms of AMOTA, LiDAR based solutions have the highest evaluation as well. Surprisingly, despite the fact that Pseudo-LiDAR is stereo based, it only trails the best LiDAR based detector by approximately 15 percentage points, which highlights also the potential that using Pseudo-LiDAR could have, if LiDAR might not be available in an effort to mitigate costs.

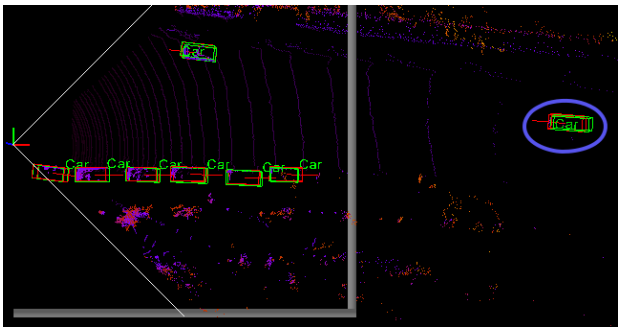
Given the promising results of PointRCNN, these input detections are selected to test the implementation of mahalanobis distance rather than the 3D-IoU for affinity measure in the data association module of tracking. Consistently, as seen in Tables III and IV, regardless of the metric used to measure performance, switching to mahalanobis distance shows a decrease in car metrics, and a considerable increase in cyclist, while maintaining similar performance for pedestrians. Due to the statistical nature of the mahalanobis distance, its ability to converge, along with Kalman Filter, is dependent upon the selection of the state covariance matrix, process and measurement noise, which vary depending on the data used. Using the 3D-IoU method, only overlapping detections get associated, which is not the case with mahalanobis distance, which potentially could explain the increase in performance for cyclists, and not cars for this particular experiment. This specific case is highlighted in Figure 11, where the mahalanobis distance is able to track cyclists from frames 35 to 44 in sequence 15 of validation set.

Detector	Method	AMOTA(↑)	MOTA(↑)	FP(↓)	FN(↓)	IDS(↓)	FRAGS(↓)
FrustumPointNet	3D-IoU	33.42	77.15	194	1713	8	49
PointRCNN	3D-IoU	45.60	85.96	358	818	0	13
Pseudo - LiDAR FPN	3D-IoU	30.11	69.48	464	2090	3	64
PointPillars	3D-IoU	43.26	84.57	252	1041	0	16

TABLE II: 3D Tracking results for the CAR class on the KITTI validation set. 3D results indicate the best evaluation point of the model.



(a) FrustumPointNet



(b) PointRCNN

Fig. 10: Qualitative results using PointRCNN and FrustumPointNet to generate tracking results. FrustumPointNet is unable to detect the furthest car to the right of the image indicated by the blue circle, but PointRCNN is (car is at 64.8m from ego-vehicle. Red Box indicates predictions, and green boxes indicate ground truth cars.

Method	Overall ↑	Car	Cyclist	Pedestrian
3D-IoU baseline	52.97	74.81	30.27	53.84
3D-IoU + covariances	53.53	74.38	32.42	53.79
Mahalanobis distance	57.25	68.77	51.26	51.72

TABLE III: 3D MOTA Tracking results for the KITTI MOT benchmark validation set using PointRCNN detections. 3D-IoU + covariances indicates that the baseline is evaluated with the same state, process and measurement noise covariance as mahalanobis distance. 3D results indicate single point evaluation of the model.

Method	Overall ↑	Car	Cyclist	Pedestrian
3D-IoU baseline	38.62	45.60	40.35	29.92
3D-IoU + covariances	38.59	45.55	40.36	29.88
Mahalanobis distance	36.36	35.42	44.02	29.64

TABLE IV: 3D AMOTA Tracking results for the KITTI MOT benchmark validation set using PointRCNN detections. 3D-IoU + covariances indicates that the baseline is evaluated with the same state, process and measurement noise covariance as mahalanobis distance.

C. ROS Results

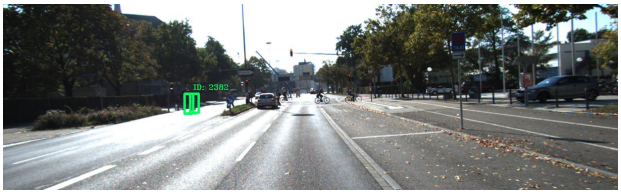
Given the pre-trained model for PointRCNN for car class, and our own training for pedestrian and cyclist (see Section IV-B for more details) , the results for detections along with its own tracks for KITTI raw data is shown in Figure 12.

The MOT algorithm is ported to the TU/e - AIIM autonomous vehicle, but with a ROS bag from the vehicle's own LiDAR sensor. As shown in Figure 13, cars are still able to be detected and tracked, but qualitatively, tracks are a bit less smooth overall as compared to the KITTI raw data. This could be explained by the fact that the pre-trained models are trained on the KITTI dataset, therefore the model struggles to generalize on completely different data. Further, no calibration is available for the generated data from the TU/e - AIIM vehicle, making the results overall less smooth. For this sample bag, no pedestrians are detected given the fact that there are almost no instances of pedestrians.

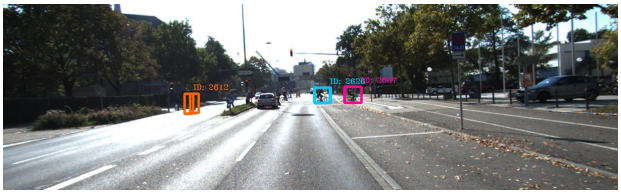
VI. Conclusion

In this paper, we presented an analysis of the effect of using different sensor modalities, such as camera, LiDAR, camera-LiDAR input detectors have on a tracking module's performance, which consists of a simple Kalman Filter for state estimation and Hungarian algorithm for data association with 3D-IoU for affinity measure. The analysis emphasized on 3D evaluation, and an analysis on the amount of false negatives generated for each input detector. It was shown that LiDAR based input detections outperform sensor fusion based input detections on the CAR class of the KITTI validation set, specifically in the MOTA, AMOTA metrics. In addition, it was shown that camera or sensor fusion based input detections used in this experiment are unable to detect objects in the 56-84m range.

Additionally, LiDAR based PointRCNN input detections, which resulted in the highest MOTA and AMOTA in the KITTI

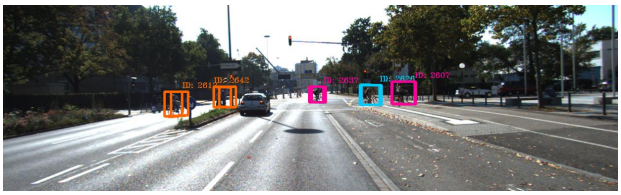


(a) 3D-IoU method sequence 15, frame 35



(b) Mahalanobis Distance method sequence 15, frame 35

(c) 3D-IoU method sequence 15, frame 44



(d) Mahalanobis Distance method sequence 15, frame 44

Fig. 11: Qualitative result of tracker evaluation using PointRCNN input detections showing the tracking results for both 3D-IoU method (a) and (c) and Mahalanobis distance method (b) and (d). The IDs on top of the bounding boxes indicate that in every case the cyclist are tracked correctly through the frames.

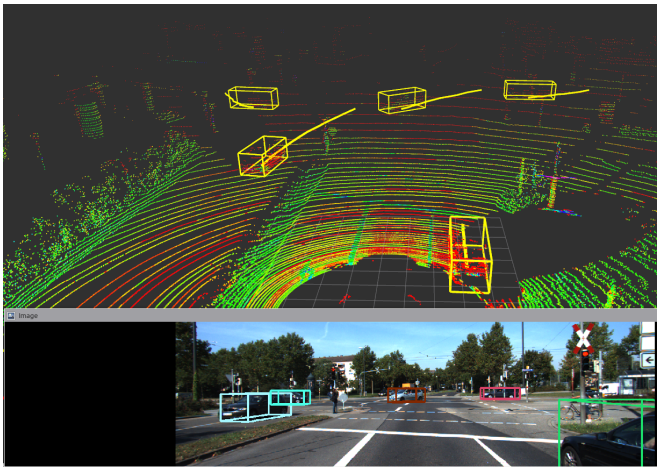


Fig. 12: PointRCNN detections with 3D-IoU affinity measure for MOT algorithm visualized on the KITTI raw data using ROS. The yellow 3D bounding boxes on LiDAR space represent cars. The yellow lines trailing the bounding boxes show the car's trajectories. The 3D bounding boxes on the image space below validate the cars found.

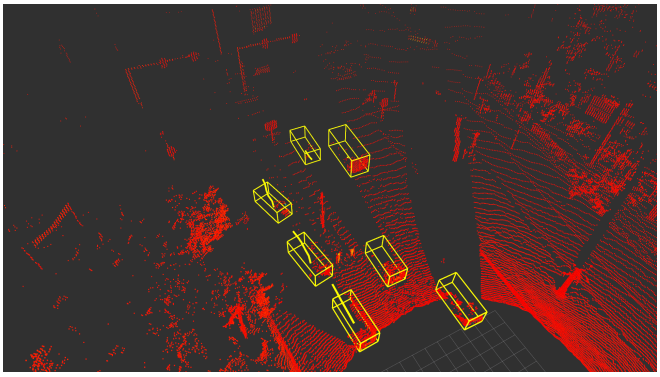


Fig. 13: PointRCNN detections with 3D-IoU affinity measure for MOT algorithm visualized on the generated data by the LiDAR sensors of the TU/e - AIIM autonomous vehicle using ROS. The yellow 3D bounding boxes on LiDAR space represent cars. The yellow lines trailing the bounding boxes show the car's trajectories.

validation set, was selected for an analysis of the tracking module, specifically the data association submodule, where the mahalanobis distance is used as the affinity measure along with the Kalman Filter. Using the mahalanobis distance showed an increase in MOTA, AMOTA for the cyclist class, yet a decrease in the car class while maintaining similar performance for the pedestrian class. Finally, the MOT tracking framework is implemented in ROS operating system, where raw data from KITTI dataset is visualized, as well as scenes generated from the LiDAR sensors in the TU/e - AIIM autonomous vehicle, in which it can be seen that the MOT tracking algorithm implemented is able to detect cars reliably at 10Hz.

Regarding further recommendations, selecting other object detectors with different sensor fusion architecture, could be useful in further analyzing the gap in MOTA, AMOTA compared to LiDAR. Furthermore, a more dynamic dataset such as NuScenes dataset could be used in conjunction with KITTI dataset to be able to determine if the tracking results follow a similar trend. Additionally, adding another simple affinity measure such as Euclidian distance could also be interesting to compare it to the results obtained with 3D-IoU and mahalanobis distance.

APPENDIX A
PSEUDO - LIDAR GENERATION

Pseudo - LIDAR [14] argues that the big gap in 3D object detection between stereo vision and LiDAR based solutions, does not come from the quality of the data, but rather from the representation of the data. Instead of using the depth map

directly produced by the stereo images into a convolutional neural network (CNN) as it has been previously performed, Pseudo - Lidar [14] proposes to convert a disparity map into a stereo depth map, which is then converted to pseudo - LiDAR by back-projecting pixels into a 3D point cloud hence imitating a LiDAR signal. The 3D location of the pixels in the left- camera coordinate frame is obtained as per equations (8) to (10). Here, the depth is z, from the disparity map D, with pixel coordinates (u,v). $(C_U, C_V$ are the pixel location corresponding to the camera center and (f_U, f_V) are the horizontal and vertical focal lengths respectively.

Once the 3D location of the pixels are obtained, the final point cloud is represented by $\{(x^n, y^n, z^n)\}_{n=1}^{N}$, where N is the pixel count.

$$z = D(u, v) \tag{8}$$

$$x = \frac{u - C_U \times z}{f_U} \tag{9}$$

$$y = \frac{v - C_V \times z}{f_V} \tag{10}$$

APPENDIX B
MAHALANOBIS DISTANCE

In order to further understand the mahalanobis distance, and its application within the data association, an example is used. Lets compare the number of detections and trajectories generated for a particular frame. According to Table V, the Hungarian algorithm should tell us whether prediction 1 belongs to observation 1 or not. The Hungarian algorithm requires an affinity measure as input indicating how affine the observations and the prediction are. Here, observation is the output of the object detector and prediction is output of the Kalman Filter.

	x	y	Number
Observation	-5.52	1.66	1
Observation	-9.14	1.58	2
Prediction	-5.36	1.73	1
Prediction	-9.69	1.61	2

TABLE V: Example set on a particular frame with predictions coming from KF and observations from the detector

Important to note that in Table V, x and y for the observations are values given by the detector, particularly here denoting x,y coordinates of an object's location. In the other hand, x and y for the predictions are the predicted state estimates mean from the Kalman Filter. In order to calculate the mahalanobis distance, the covariance matrix must be known for each prediction. For the sake of simplicity, lets assume $S_1 = \left[\begin{smallmatrix} 2.98 & 0 \\ 0 & 2.98 \end{smallmatrix}\right]$ and $S_2 = \left[\begin{smallmatrix} 2.83 & 0 \\ 0 & 2.83 \end{smallmatrix}\right]$. With these values, we can compute the mahalanobis distance $d_m = \left[\begin{smallmatrix} 0.25 & 16 \\ 14.05 & 0.43 \end{smallmatrix}\right]$, where rows indicate an observation and columns a prediction. According to the distance calculated, it shows that observation and prediction (1,1) and (2,2) are most likely related, indicated by the smaller distance. The Hungarian algorithm will use these values to output indices indicating which detection and prediction are a match, and which are not a match and need to be discarded. As initially hypothesized, $D_{match} = [1, 1], [2, 2]$.

REFERENCES

[1] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," Tech. Rep., 2015.

[2] W. Montgomery, R. Mudge, E. Groshen, S. Helper, J. MacDuffie, and C. Carson, "America ' s Workforce and the Self-Driving Future: Realizing productivity gains and spurring economic growth," no. june, 2018.

[3] "Lidar vs. Camera: Driving in the Rain — Ouster." [Online]. Available: https://ouster.com/blog/lidar-vs-camera-comparison-in-the-rain/

[4] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[5] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 770–779, 2019.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 6 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," 12 2015. [Online]. Available: http://arxiv.org/abs/1512.02325http://dx.doi.org/10.1007/978-3-319-46448-0_2

[8] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 77–85, 2017.

[10] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499, 2018.

[11] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 918–927, 2018.

[12] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5750–5757, 2018.

[13] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object

detection from point clouds," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 12 689–12 697, 2019.

[14] Y. Wang, W. L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 8437–8445, 2019.

[15] X. Weng and K. Kitani, "Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud," 2019. [Online]. Available: http://arxiv.org/abs/1903.09847

[16] R. Qin, X. Huang, W. Liu, and C. Xiao, "Pairwise Stereo Image Disparity and Semantics Estimation with the Combination of U-Net and Pyramid Stereo Matching Network," pp. 4971–4974, 2019.

[17] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2002–2011, 2018.

[18] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," *Technical Report CMU*, no. CS-91-132, 1991.

[19] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 3701–3710, 2017.

[20] X. Weng, J. Wang, D. Held, and K. Kitani, "3D Multi-Object Tracking: A Baseline and New Evaluation Metrics," 2019. [Online]. Available: http://arxiv.org/abs/1907.03961

[21] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2017-Septe, pp. 3645–3649, 2018.

[22] D. B. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[23] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association," *IEEE Journal of Oceanic Engineering*, vol. 8, no. 3, pp. 173–184, 1983.

[24] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2016-Augus, pp. 3464–3468, 2016.

[25] H.-k. Chiu, A. Prioletti, J. Li, and J. Bohg, "Probabilistic 3D Multi-Object Tracking for Autonomous Driving," 1 2020. [Online]. Available: http://arxiv.org/abs/2001.05673

[26] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11210 LNCS, pp. 472–487, 2018.

[27] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems*, vol. 2017-Decem, pp. 5100–5109, 2017.

[28] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, pp. 35–45, 1960.

[29] Anna Petrovskaya and Sebastian Thrun, "Model Based Vehicle Detection and Tracking for Autonomous Urban Driving," *Autonomous Robots*, vol. 26, pp. 123–139, 2009. [Online]. Available: http://cs.stanford.edu/group/manips/publications/pdfs/Petrovskaya_2009_AURO.pdf

[30] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 3 1955. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/nav.3800020109https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109https://onlinelibrary.wiley.com/doi/10.1002/nav.3800020109

[31] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[32] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granstr, "Mono-Camera 3D Multi-Object Tracking Using Deep Learning Detections and PMBM Filtering."

[33] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *Eurasip Journal on Image and Video Processing*, vol. 2008, 2008.